SECURITY OF CERTIFICATE-BASED PROTOCOLS: FOCUS ON SERVER AUTHENTICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SELİM BARAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

MAY 2015

Approval of the thesis:

SECURITY OF CERTIFICATE-BASED PROTOCOLS: FOCUS ON SERVER AUTHENTICATION

submitted by **SELİM BARAN** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Bülent Karasözen Director, Graduate School of **Applied Mathematics**	____________
Prof. Dr. Ferruh Özbudak Head of Department, **Cryptography**	____________
Prof. Dr. Ferruh Özbudak Supervisor, **Cryptography, METU**	____________
Prof. Dr. Ali Aydın Selçuk Co-supervisor, **Department of Computer Engineering, TOBB University**	____________

Examining Committee Members:

Prof. Dr. Ferruh Özbudak Department of Cryptography, METU	____________
Prof. Dr. Ali Aydın Selçuk Department of Computer Engineering, TOBB University	____________
Assoc. Prof. Dr. Ali Doğanaksoy Department of Mathematics, METU	____________
Assoc. Prof. Dr. Murat Cenk IAM, METU	____________
Asst. Prof. Dr. Fatih Sulak Department of Mathematics, Atılım University	____________

Date: ____________

ABSTRACT

SECURITY OF CERTIFICATE-BASED PROTOCOLS: FOCUS ON SERVER AUTHENTICATION

BARAN, Selim
M.S., Department of Cryptography
Supervisor : Prof. Dr. Ferruh Özbudak
Co-Supervisor : Prof. Dr. Ali Aydın Selçuk

May 2015, 75 pages

Today, secure communication channels are mostly set up via certificate-based protocols, such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS). Although they have been used for years and in so many areas, from e-commerce and internet banking to secure channel needs in military, there have been several attacks on their security model, which forced researchers to make studies on them. In this thesis, we will explain their security model, the vulnerabilities discovered so far, the precautions for these vulnerabilities and at the end, we will focus on SSL authentication piece and the popular solutions for improving SSL server authentication, such as Certificate Pinning, Convergence and Certificate Transparency which are all in the active research area to define the future of SSL and TLS protocols.

Keywords : SSL authentication, Certificate Transparency, Convergence, DANE, Certificate Pinning.

ÖZ

SERTİFİKA TABANLI PROTOKOLLERİN GÜVENLİĞİ: SUNUCU KİMLİK DOĞRULAMASINA DETAYLI BAKIŞ

BARAN, Selim
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Prof. Dr. Ferruh Özbudak
Ortak Tez Yöneticisi : Prof. Dr. Ali Aydın Selçuk

Mayıs 2015, 75 sayfa

Günümüzde güvenli iletişim kanalları genellikle Secure Sockets Layer (SSL) ve Transport Layer Security (TLS) protokolleri ile gerçekleştirilir. Bu protokoller uzun yıllardır e-ticaret ve internet bankacılığından, askeri güvenli iletişim kanalları tesisine kadar birçok alanda kullanılıyor olmasına rağmen, güvenlik zâfiyetlerini hedef alan saldırılar nedeniyle, birçok çalışmaya konu olmuşlardır. Bu tezde, SSL ve TLS protokollerinin güvenlik mimarileri, şu ana kadar tespit edilen zafiyetler ve bunlar için alınabilecek tedbirler kısaca açıklandıktan sonra, SSL sunucu kimlik doğrulama işlevine ve bu işlevi güçlendirmeye yönelik çoğunlukla son 5 yılda ileri sürülmüş projeler olan Certificate Pinning, Convergence ve Certificate Transparency dâhil 10 farklı projeye odaklanılmaktadır.

Anahtar Kelimeler: SSL kimlik doğrulama, Certificate Transparency, Convergence, DANE, Certificate Pinning.

To My Family

ACKNOWLEDGMENTS

Firstly, I would like to express my special thanks my co-supervisor, Prof. Dr. Ali Aydın Selçuk, for his great guidance and motivation throughout this study. He allocated lots of his valuable time for me. I surely would not be able to finish this thesis without his support.

Secondly, I thank my supervisor and also the head of cryptography department, Prof. Dr. Ferruh Özbudak. It was an honour for me to be a student of him.

Thirdly, I should express my thanks to director of IAM, Prof. Dr. Bülent Karasözen, and the personnel in the secretariat of IAM, Mrs. Nejla, Mr. Serkan and Ms. Cevher for their kindness, help and support during my education.

Finally, I would like to thank to my wife and son for their understanding and support during my study.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Starting from the 1990s, the Internet has been changing our lives. We have been using it increasingly, for many things making our lives easier and enjoyable. Today, in addition to the computers, smartphones, tablets and many other devices are connected to the Internet.

When the Internet was designed, its security was not thought as an important issue because there were not many websites requiring security. There were mostly static web pages, containing information like news. With the spread of dynamic web pages, communications dealing with millions of dollars of e-commerce and internet banking have become available online. Both the companies and the customers have liked those opportunities because they provided ease in operations.

However, when the communications started containing sensitive data such as user passwords and credit cards information, we have needed a secure way for the communication to prevent the attacks such as identity theft.

The required security measures are mainly: Preventing intrusions, protecting information in transit, ensuring availability and integrity.

In this chapter, we aim to make a brief introduction to the basics of Internet and network security, cryptography, public key infrastructure, certificate-based protocols and finally, we will give our motivation of the thesis.

1.1 Preliminaries

1.1.1 Network Security

Internet security is based on network security since the Internet is basically ‘the networks of networks’[1].

Securing the data in transit is hard since there are many attack types to deal with. There may be vulnerabilities in the networks due to their design and weaknesses in their components such as software bugs and configuration mistakes. The attackers can exploit the vulnerabilities by making attacks divided mainly into two types: Active and passive.

In the active attack scenario, the attacker changes the data while in transit and breach the integrity of the data, whereas in passive attack, he/she only reads it. Man-in-the-Middle (MitM) and SQL injection are among the examples of active attacks, and, port scanning and wire-tapping are the among examples of passive attacks.

Network security aims to provide three properties: Confidentiality, integrity and authenticity. Confidentiality is providing the secrecy of the data from the unauthorized parties, integrity is proving the accuracy and completeness of the data and lastly, authenticity is ensuring the sender and the recipient to be the intended one. In order to provide these properties, network security mostly utilizes cryptography.

1.1.2 Cryptography

Cryptography is the study of techniques to provide security in communications in presence of adversaries [1]. It is heavily based on mathematics.

Here are some basic terms used in cryptography:

- **Plaintext:** The message to be secured from people other than its intended recipients.
- **Encryption:** Transforming the message into a concealed form using a secret key.
- **Ciphertext:** Encrypted plaintext.
- **Decryption:** Transforming ciphertext back into original plaintext using a secret key.

Cryptography deals with securing the data from unauthorized eyes, whereas cryptanalysis, another field of study, tries to get the plaintext from ciphertext, partially or fully.

[1] That is the definition used in RFC 1122: Internet Standard. Internet consists of interconnected networks among host computers/devices. Each network are connected to the global network via devices called *gateways* or *routers* that deal with packet switching.

There is a term called ‘computationally hard’ used in cryptography and cryptanalysis. It means that the time needed to perform the task is too long, i.e. much more than billions of billions of years with the current technology in microprocessors. We expect cryptographic algorithms to have a design which is computationally very hard to break.

In a famous notation, sender of the message is named as Alice, receiver is Bob and the bad guy, the eavesdropper, is Eve.

New types of cryptography have emerged after the spread of computer communications. Today, cryptography is mainly divided into two categories: Secret and public key cryptography.

1.1.2.1 Secret Key Cryptography

Secret (symmetric) key cryptography was the only known type in the history and has been used for thousands of years. In its early forms, encryption is made by shifting the letters in the plaintext, such as in Caesar cipher[2]. Those kind of ciphers are called *shift ciphers*.

After cryptography progressed, another way was used: Substituting each letter with another letter. For example change ‘A’ to ‘R’ and so on. To decrypt, reverse operation is done. To achieve that, a lookup table is used for both encryption and decryption. This type is known as *substitution cipher*, and it is the most used type in the history of cryptography.

As seen, there is no key used in such ciphers. It is the algorithm that provides security. But at 19th century, very different approach came out: “A cryptosystem should be secure even if the attacker knows everything about the system, except the secret key”. This is the idea of a cryptographer named Auguste Kerckhoffs and its approach is known as *Kerckhoffs principle* [2]. Today’s modern cryptosystems mostly obey this principle.

Shannon published his famous ‘A Communications Theory of Secrecy Systems’ [3] paper in 1948. In that paper, Shannon defined the fundamentals of *perfect cipher*. He also introduced *diffusion* and *confusion* concepts, which lie in the basis of the current cryptosystems’ design.

In modern symmetric key cryptography that we use today, secure communication is done as in the following example: Alice and Bob agrees on a secret key information and an encryption algorithm beforehand. To send his message to Bob, Alice encrypts the message with the secret key and sends the ciphertext. Bob decrypts the ciphertext using the same key and gets Alice’s message. Eve, who tries to see the message only gets the ciphertext and since he does not know the secret key, he can not access the original message of Alice. That communication can continue securely if the secret key is not revealed to Eve. The process is depicted in Figure 1.1.

[2] In Caesar cipher encryption, letters in alphabet are shifted by 3. For example, A become D, B become E and so on. Its decryption is done as reverse, shift back with 3.

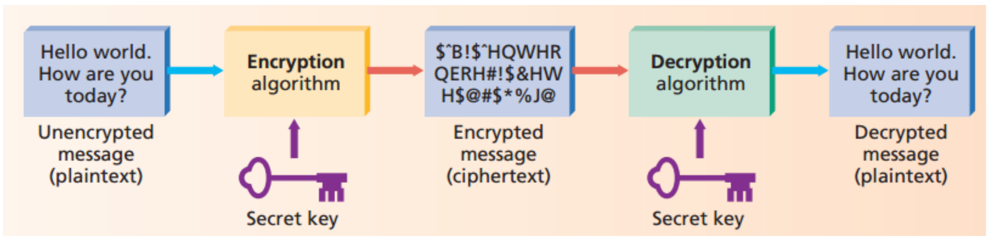

Hello world. How are you today?
Unencrypted message (plaintext)
Encryption algorithm
Secret key
$^B!$^HQWHR QERH#!$&HW H$@#$*%J@
Encrypted message (ciphertext)
Decryption algorithm
Secret key
Hello world. How are you today?
Decrypted message (plaintext)

Figure 1.1: Symmetric key cryptography: Encryption and decryption are made using the same key[3].

In 1990s, when the need of the secure cryptographic algorithm arose, National Institute of Standards and Technology of U.S. (NIST) announced a competition to choose a "publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century"[4]. Afterwards, NIST accepted candidate algorithms: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, Twofish. After comparing the algorithms according to criteria such as security, performance and feasibility, *Rijndael* was selected as the Advanced Encryption Standard (AES)[5] on October 2, 2000. The other finalists were Serpent, Twofish, RC6 and MARS [4].

Today, AES is the most used algorithm to provide data confidentiality. There are many implementations of it both in hardware modules and software. Some of its operations are even supported in recent processors' instruction sets. Its recommended key size is 128 bits but 256 bits are recommended for more sensitive data.

Main problem of symmetric key cryptography is *key distribution* since it requires secure exchange of secret keys beforehand. Moreover, when it is used for communication with several people, we need to use many secret keys. Hence, using symmetric key cryptography directly in the Internet is not possible.

1.1.2.2 Public Key Cryptography

The invention of public (asymmetric) key cryptography (PKC) can be called a revolution in the history of cryptography [5]. Until its emerge, cryptography was not widely used due to the key distribution problem.

In public key cryptography, two different keys are used: Public (revealed to everyone) and private (kept secret). Public and private keys are generated as a pair, known as *key pairs*. To send a message to Bob, Alice encrypts his message with Bob's public key. Bob, afterwards, decrypts the ciphertext using his private key. Since only Bob has the private key, Eve can not decrypt the ciphertext. This is depicted in Figure 1.2.

[3] Source: `http://csee.wvu.edu/~katerina/Teaching/CS-465-Fall-2007/SSL-Basics.pdf`

[4] For details of the announcement see: `http://csrc.nist.gov/archive/aes/pre-round1/aes_9701.txt`

[5] For standardization paper, see: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

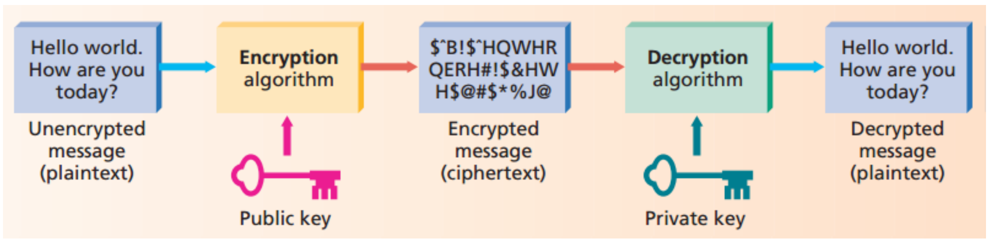
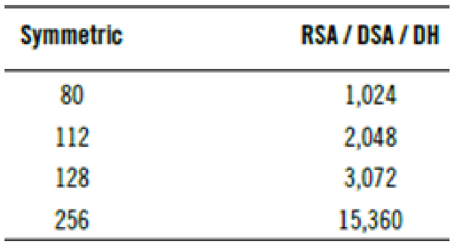

Hello world. How are you today?
Unencrypted message (plaintext)
Encryption algorithm
Public key
$ˆB!$ˆHQWHR QERH#!$&HW H$@#$*%J@
Encrypted message (ciphertext)
Decryption algorithm
Private key
Hello world. How are you today?
Decrypted message (plaintext)

Figure 1.2: Public key cryptography: Encryption and decryption keys are different[6].



Symmetric	RSA / DSA / DH
80	1,024
112	2,048
128	3,072
256	15,360

Figure 1.3: Key strength equivalence of secret and public key algorithms.

The process stated above is the typical use of the RSA[7] algorithm. To get the private key from the public key, Eve must factor hundreds digits long integer. Integer factorization is one of the well-known problems in mathematics, and currently even the fastest algorithms such as *General Number Field Sieve*[8] can factor such big integers in millions of years.

Today, RSA is the most popular algorithm[9] used in public key cryptography type. Its recommended key size is at least 3072 bits, whose security level is equivalent to approximately 128 symmetric bits. For a comparison chart, see Figure 1.3.

Public key cryptography is approximately 1000 times slower than symmetric key cryptography, so it is not suitable for encrypting huge amount of data. Thus, it is usually used for key exchange between Alice and Bob, afterwards, rest of the communication continues with fast symmetric encryption.

Another tool we gained with the public key cryptography is digital signatures. Alice signs (encrypts with his private key) a message, then, anyone can verify the signature using Alice's public key. Note that, RSA algorithm is also used for digital signatures in addition to its encryption usage.

[6] Source: `http://csee.wvu.edu/~katerina/Teaching/CS-465-Fall-2007/SSL-Basics.pdf`

[7] RSA is the concatenation of the first letters of Rivest, Shamir and Adleman, the designers of that algorithm. It is developed in 1977.

[8] For details, see: `http://www.untruth.org/~josh/math/NFS.pdf`

[9] Another algorithm for exchanging keys between two parties securely, is the one developed by Diffie and Hellman (DH). Its security is based on another well-known problem in mathematics: Discrete logarithm problem.

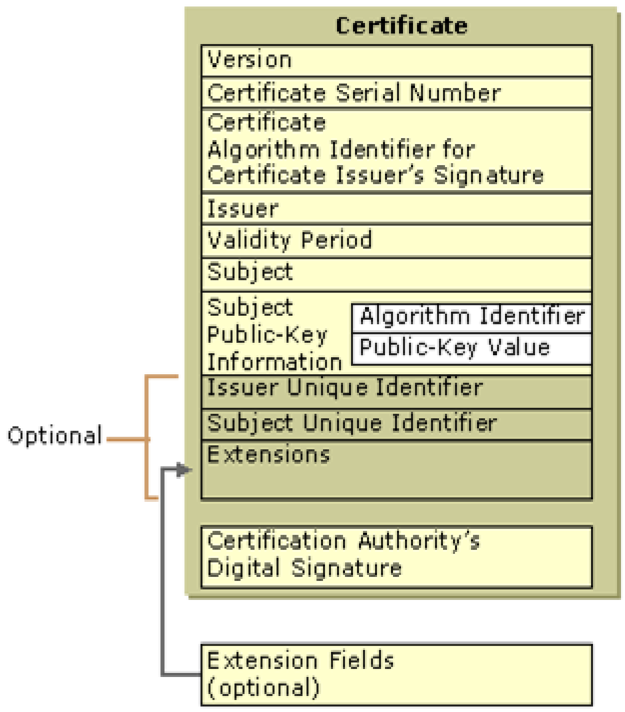

Certificate
Version
Certificate Serial Number
Certificate Algorithm Identifier for Certificate Issuer's Signature
Issuer
Validity Period
Subject
Subject Public-Key Information
Algorithm Identifier
Public-Key Value
Issuer Unique Identifier
Subject Unique Identifier
Extensions
Certification Authority's Digital Signature
Optional
Extension Fields (optional)

Figure 1.4: X.509 v3 certificate fields[10].

1.1.3 Public Key Infrastructure (PKI)

In order to utilize PKC in practice, we needed a standard for the Internet about sharing the public keys. Digital Certificates are the first step used for that purpose.

Digital Certificates are the basic building blocks of PKI. They are just container files, containing public keys along with additional information: Serial number, key size, algorithm etc. The chosen certificate format for the use in the Internet is X.509, that was initially issued by ITU Telecommunication Standardization Sector in 1988. It is currently in 3rd version (X.509 v3) and has the structure shown in Figure 1.4.

A certificate binds a public key to a domain name. In X.509 certificate type, the domain name is placed under common name (CN) that is the part of Subject attribute.

Before implementing the tools we gained via PKC, many issues were faced, including; how to handle storing and revoking of the public keys and how to scale with huge number of servers? These are the main issues that PKI was developed to deal with.

Although many other definitions[11] can be made, "Public Key Infrastructure is a framework that builds the network of trust and combines public key cryptography and digital signatures to ensure confidentiality, integrity, authentication and access control" [6] is

[10] Source: `https://technet.microsoft.com/en-us/library/cc962029.aspx`

[11] According to RFC 2828, a PKI is "a system of CAs that perform some set of certificate management, archive management, key management, and token management functions for a community of users that employ public key cryptography."

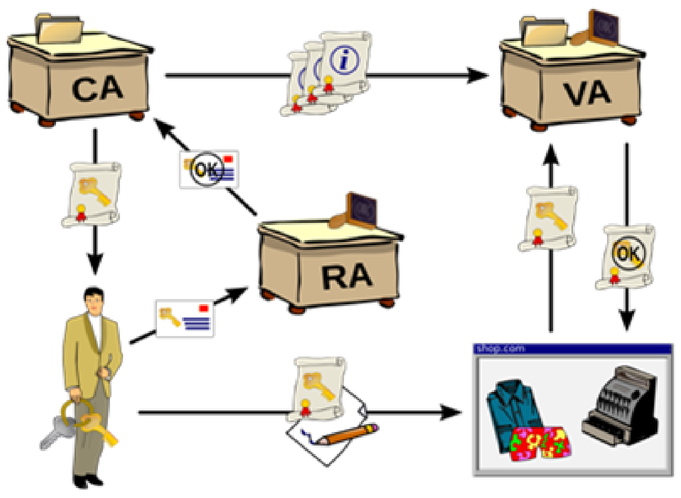


Figure 1.5: The components of PKI[13].

a good one.

The main components of PKI are Certificate Authority (CA), Registration Authority (RA) and Validation Authority (VA). CA[12] is a trusted 3rd party, trusted by certificate owners and certificate verifiers. The job of RA is to bind keys to the users. The users will register with RA to get a certificate (In fact, RA is just a company that processes requests on behalf of a CA). Lastly, VA handles the task of validating the users. The roles of the PKI components are depicted in Figure 1.5.

In PKI, root CAs can issue certificates or delegate this authority to subordinate CAs (also called sub-CAs or intermediate-CAs). As a result, chain of certificates, called *certificate chain* occurs. This is depicted in Figure 1.6.

Clients trust root-CAs and validate certificate chain up to a root-CA. The common usage of certificate verification is done in web browsers, and the certificates containing the public keys of root-CAs are usually preloaded to web browser code. The database where those root certificates are stored at clients are called *root store*.

The main organization in PKI, CAs, issue millions of certificates every year. It is known that there are nearly 100 root-CAs and also more than 1000 sub-CAs, but in fact, it is not exactly known how many sub-CAs exist because root-CAs can create sub-CAs as many as they want[15].

PKI is a general framework and some required adaptation are done for using PKI in

[12] The first and the largest CA is Verisign and it was founded in 1995. Some other big CAs are Comodo, Go Daddy, GlobalSign, Digicert, StartCom and Entrust.

[13] Source: `http://en.wikipedia.org/wiki/Public_key_infrastructure`

[14] Source: `http://www.net.in.tum.de/fileadmin/TUM/NET/NET-2013-02-1/NET-2013-02-1_07.pdf`

[15] For an animated diagram about CAs and sub-CAs, see: `http://notary.icsi.berkeley.edu/trust-tree/`.

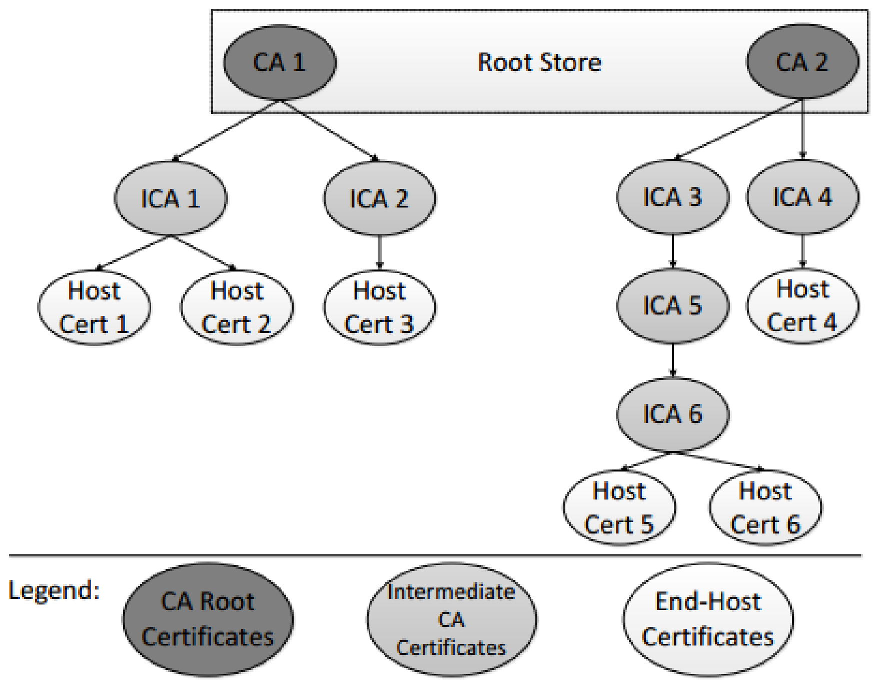


Figure 1.6: X.509 certificate chain in PKI[14].

web, creating a new standard: Web PKI[16].

A common role of Web PKI can be summarized as follows: The owner of a website that wants to provide secure communication for users to his/her website, firstly needs a digital certificate. For this, he creates a digital certificate with the free tools available both in operating systems (OS) and the Internet, then makes a certificate signing request (CSR) to a CA which he prefers. In that request, he sends the certificate along with some information that the CA requires about his identity. The CA signs the certificate with its private key then sends back to the website owner. Mostly such process ends in 1-2 hours. Finally, website owner installs the certificate to the web server of his website. The website is now ready for secure communications. Since the public key of the CA is shipped with OS or web browser code, the web browsers will authenticate the server by verifying the certificate and will start secure communication.

Today, CAs issue three types of certificates[17]:

- **Domain validation (DV):** It validates the domain ownership. The process happens usually via emails between the CA and the website owner. As seen, there is no identity verification in issuing these certificates. Since the process is fast and that type of certificates are the cheapest, most server owners prefer them.

[16] Web PKI was first introduced in 1994. For details, see RFC 5280.
[17] For a good summary, including some example pictures in the browsers refer to: `http://simonecarletti.com/blog/2013/11/ssl-certificate-types`

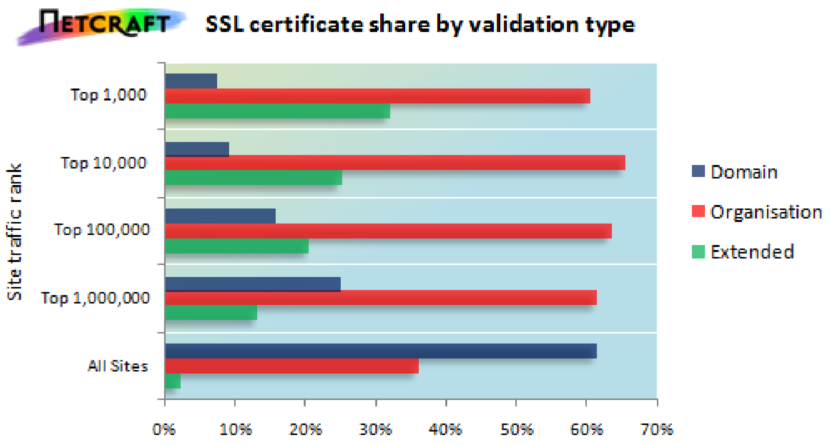


Figure 1.7: Usage of certificate types in Internet[18]. As noticed, big websites prefer mostly OV/EV, whereas normal websites prefer DV type.

- **Organization validation (OV):** It validates the domain ownership and organization information included in the certificate such as name, city and country. For that kind of certificates, CAs require additional documents to verify the company identity. Issuance process lasts several days.
- **Extended validation (EV):** For EV certificates, CAs require identity verification with very strict rules. Such rules include even face to face meeting with the owner of the websites. They are mostly preferred by companies dealing with jobs such as e-commerce and online banking. The process of issuance can take weeks. Web browsers usually show a green address bar for the websites using EV certificates.

A statistical information for their usage is shown in Figure 1.7.

Another important task in PKI is certificate revocation. Certificates may need to be revoked when they are not needed anymore or when their private key is stolen. There are two ways for revocation: Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP).

CRL is a list of serial numbers of revoked certificates and kept by CAs. The CRL location of a certificate is in 'CRL Distribution Points' field of the certificate. After time, CRLs became large, so it is slow for real-time search.

OCSP is used to get the revocation/freshness status of a queried certificate. The servers giving such service are called *OCSP servers*. The OCSP server location of a certificate is determined in the 'Authority Information Access' field of the certificate. Although OCSP is better than CRL, it suffers privacy and performance problems. OCSP servers has to respond for the check of validity of every certificate in real time. Since clients make the request to OCSP servers directly, that servers see the browsing preferences of the clients which is a violation of client privacy.

[18] Source: `https://www.entrust.com/ev-ssl-market-growing-where-it-counts`

Another method, called *OCSP stapling* is developed to solve the problems in OCSP. In this method, the target website queries the OCSP server periodically and embed an OCSP response in startup of secure communication. Since there is no 3rd party need, privacy and performance issues are solved. Although this method is better than the others, only 20 percent[19] of the websites are using this method, because OCSP stapling supports only one OCSP response at a time, which is insufficient for certificate chains with sub-CA certificates[20].

Although we gained lots of security improvement in terms of authentication via Web PKI, it has some problematic issues that caused various cyber attacks in recent years. The main reason lies behind the problems are the fact that root CAs and subordinate CAs are capable of issuing a certificate to any domain without knowledge of domain owners, and CAs are blindly trusted forever by the client software such as web browsers, there is no *trust agility*[21]. This reality reduces the Web PKI system security to the weakest CA, that is, if an attacker controls a root-CA or a sub-CA, or steal its private key, he/she can issue a forged certificate to any domain name and afterwards, can use that certificate to decrypt the communications done to that domain.

1.2 Certificate-Based Protocols

In order to provide secure communication on the Internet, we need some standards to make a key exchange securely between the client and the server. Those standards should contain methods when and how to use secret and public key cryptography to secure the connection. What we use for that need are known as protocols, and the ones we use in the Internet are Secure Sockets Layer (SSL) and Transport Layer Security (TLS), which are based on certificates.

Today, SSL and TLS are the most used protocols[22] to secure data transit in client-server communications. Mainly online banking sites, e-commerce sites and email servers use them [7].

1.2.1 Secure Sockets Layer (SSL)

The main role of SSL is to provide security for client server communications, and for that purpose, it uses cryptography, digital signatures and X.509 certificates. In typical

[19] Source: `http://www.netcraft.com/internet-data-mining/ssl-survey/`

[20] For details about OCSP stapling, see: `https://www.maxcdn.com/one/visual-glossary/ocsp-stapling/`

[21] Trust agility means the option for the clients to change their trusted sources whenever they want without facing any problems. But today, in current CA system, it is not possible because, when the client removes a CA certificate from his/her trust store, the websites, using the certificates having a certificate chain up to that CA, will be unauthenticated while connection startup

[22] Another popular protocol for securing the network connections is IPSec, but it operates on IP layer of OSI model. Hence it is required to be supported by operating systems to be used unlike the SSL/TLS which is operated at transport layer, and all operating systems and browsers support them. SSL/TLS can be used to secure any application layer protocol such as HTTP, SMTP and XMMP.

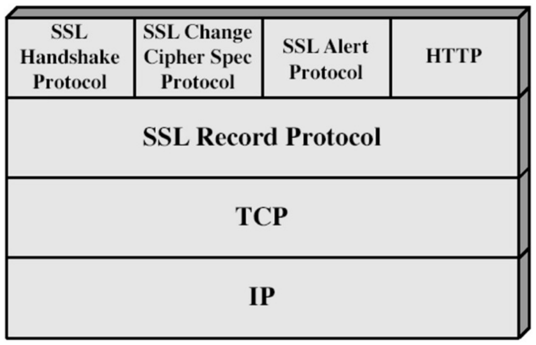


Figure 1.8: SSL protocol stack[24].

SSL usage, the client initiates the connection, authenticates the server and the server responds to the client. In web usage, client is a web browser and server is a web server. SSL-secured website names begin with ‘https’ (meaning http secure), such as ‘https://www.google.com’.

SSL protocol was developed by Netscape company in order to provide secure communication in their web browser, Netscape Navigator. Taher Elgamal, once a chief scientist at Netscape, is known as the ‘father of SSL’ [8].

The first version, SSL 1, was never released. The second version, SSL 2 come out in 1995. Since it had many problems and weaknesses, SSL 3 was released by Netscape in 1996. This version had a good design and had the core of the SSL that we use today [9].

Without SSL, the traffic between browsers and web servers is carried in plaintext, therefore it is vulnerable to eavesdropping. SSL allows sensitive information such as credit card numbers and passwords to be transmitted securely. With its emerge, e-commerce and many other lucrative businesses flourished.

The basic usage of SSL is follows: Client generates a pseudo-random key K with the help of PRNG[23], and sends K after encrypting with server’s public key, stored in server’s certificate. Server decrypts it using his private key and obtains K. Afterwards, since they agreed securely on a key K, they start secure communication, symmetrically encrypting further data with the key K.

SSL is not a single protocol, it consists of 4 subprotocols as depicted in Figure 1.8.

The client authenticates the SSL server during the handshake process. In the handshake, the cipher suites are agreed by client and server in change cipher subprotocol. After the handshake, the data is encrypted in the record protocol phase. If there occurs any alarms during these tasks, the alert is handled with the alert protocol.

The details of the subprotocols are given below[25]:

[23] For detailed information about randomness and random number generation, see: `http://www.cypherpunks.to/~peter/06_random.pdf`

[24] Source: `http://www.facweb.iitkgp.ernet.in/~sourav/SSL.pdf`

[25] For more details of the subprotocols, see: `https://technet.microsoft.com/en-us/library/`

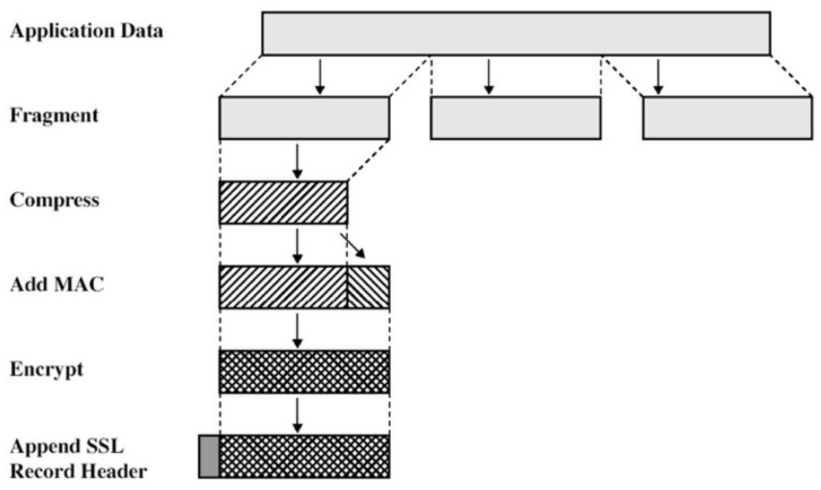


Figure 1.9: SSL record protocol operations[27].

- **The handshake protocol**, also known as *SSL handshake*, allows the client and server to authenticate each other and to negotiate cryptographic algorithm and keys.

- **The change cipher protocol** is a part of the handshake protocol and is used to agree on the cipher suites to use between the client and server.

- **The alert protocol** is another part of the handshake protocol that deals with alert messages. If an alert occurs, the session is either ended or the recipient has a choice to end the session[26].

- **The record protocol** provides two features: Confidentiality and message integrity. This protocol receives the data from upper/lower layer and afterwards;

  - Fragments/reassembles the data blocks.
  - Give numbers to the data blocks in the message to resist attacks that try to reorder data.
  - Compresses/decompresses (optionally) the data with the compression algorithm if negotiated in the handshake.
  - Encrypts/decrypts the data with the encryption algorithm and keys negotiated in the handshake.

The operations of SSL record protocol is depicted in Figure 1.9.

cc785811%28v=ws.10%29.aspx

[26] Some of the alerts, defined in the RFC 5246 document, are: handshake failure, certificate expired, unknown CA, access denied and decrypt error.

[27] Source: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html

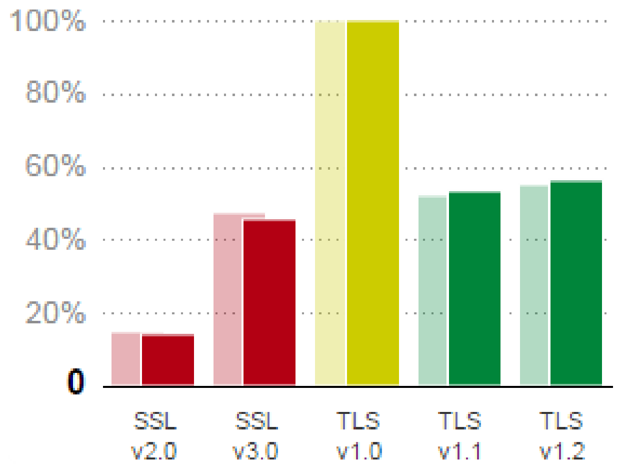


Figure 1.10: SSL/TLS protocols, supported in web servers[30].

1.2.2 Transport Layer Security (TLS)

The need of secure Internet communication was felt by vendors such as Microsoft in addition to Netscape. Hence, in order to define a standard protocol, Internet Engineering Task Force (IETF) formed a working group. That working group released a new protocol, called TLS in 1999 (RFC 2246). In fact that version (v 1.0) was nearly the same as SSL 3[28], only minor changes are done, maybe to be able to change the name of the protocol. TLS, afterwards has become the Internet standard.

TLS 1.1, containing major security fixes, was released in 2006. Finally TLS 1.2 was released in 2008 (RFC 5246). This version made the protocol more flexible, by removing hard-coded parameters. TLS 1.3 is currently a draft[29]. Day by day, more recent versions of the TLS protocol get used in web servers. The current statistics are depicted in Figure 1.10.

1.2.3 Final Remarks on SSL/TLS

SSL/TLS provides us confidentiality and integrity with their record protocol. To satisfy our security needs, they should also provide us authentication so that we would be sure that the URL address in the browser really is the web server we intended to communicate. Authentication is mostly done as client's authentication of the server in these protocols. Clients, as mentioned before, validates the server certificates which the servers presented while SSL/TLS handshake, via the public keys of the CAs, preloaded in client root store.

[28] Since TLS 1.0 was nearly the same as SSL 3.0, it was sometimes called as SSL 3.1.

[29] TLS 1.3 is based on TLS 1.2, with changes such as removing changeciphersuites/renegotiation/compression to defend against recent attack incidents on TLS.

[30] Source:`https://www.trustworthyinternet.org/ssl-pulse`. Furthermore, very good statistics about TLS primitives in use in current web world are given in that website.

If the certification validation process fails, SSL/TLS client software typically show a warning to the user. Such warnings does not deter users to close the web page in most cases. Nearly half of the users tend to continue though the warning message[31].

Although servers can also authenticate clients via SSL/TLS, that feature is used very rarely. Only some websites, giving paid services to their customers, and requiring higher security, use client authentication.

The lack of client authentication in current SSL/TLS usage, opens a hole for active attackers that can manipulate the connection between client and server, The attacker, who makes a SSL/TLS connection to the client with his/her rogue certificate, also makes another connection to the web server, pretending to be a client. Afterwards, the attacker acts as an intermediary, and gets able to decrypt the communication without being noticed. There has been many incidents in recent years, using that weakness in authentication piece of SSL/TLS. To prevent such attacks, some other measures must be deployed. Otherwise, attackers will continue to use that weakness in the future as well.

1.3 Motivation and Contribution of the Thesis

SSL was born due to the need of secure communication in the Internet. When we believed that it provided security we looked for, it flourished. We started using it for many risky situations, where huge amount of money and our privacy[32] are involved: Online banking, e-commerce, Internet voting, emails etc. Since we have trusted SSL, we increasingly continue to use it for more security-requiring tasks. However, as explained shortly in this introduction chapter, and as it will be seen in the next chapter, there are many threats for SSL, and maybe the most urgent ones to be secured is about its authentication piece.

More than ten proposals have already been put forth to solve the weakness in SSL/TLS authentication (as explained in Chapter 3), but none of them is widely deployed yet, hence, we should continue to search for a solution that will help us to solve the problem.

The contributions of this thesis are fourfold. First, we aim to present a clear picture of *SSL server authentication problem* in Chapter 2, along with mentioning some other threats in SSL. Second, we will explain some of the proposals put forth to solve the problem, in Chapter 3. Third, we will compare those proposals according to some desired properties we wanted to have in the ideal solution, in Chapter 4. Fourth, in the

[31] For a detailed study on that subject, see: `https://www.usenix.org/legacy/event/sec09/tech/full_papers/sec09_browser.pdf`.

[32] We, as users, focus on our privacy but from the governmental perspective, user privacy sometimes causes big problems. Recently, the prime minister of England said that current laws are insufficient to defend against terrorism. Afterwards, a European police chief, Mr. Wainwright stated that current laws should be reviewed to allow security agencies to monitor all areas of the online world. He also mentioned the sophisticated online communications, to be the biggest problem for security agencies tackling terrorism. For details see: `http://www.bbc.com/news/technology-32087919`

Conclusion chapter, we will give our comments and suggestions for the solution with some recommendations for the future work on the subject.

Note: Since TLS is the successor of SSL 3, and very similar to it, we will use the term ‘SSL’, meaning SSL/TLS in the following chapters.

CHAPTER 2

SECURITY OF SSL IN PRACTICE

Internet security is based on SSL security as explained in the introduction chapter. SSL has become a mature protocol after years. But it is secure if the issues about its security explained in this chapter are taken into consideration.

Firstly, SSL relies on cryptography and it is as secure as the cryptographic primitives used inside it. Cryptography is an improving subject and the cryptographic algorithms which were accepted secure and became a standard years ago, may be seen insecure now. DES block cipher and MD5 hashing algorithm are the well known examples in this manner. They are replaced by AES and SHA1 nowadays because they are faced to being cracked with the advances in computer processing power and cryptanalysis. Hence, secure cryptographic primitives should be used in SSL.

Secondly, there are SSL protocol and implementation related issues. Latest SSL (TLS 1.2) are standardized in RFC 5246. But this does not mean that, it is exactly the same as we use today in practice. There may be many implementation related wrong use issues. SSL are implemented in browsers and also in software libraries such as OpenSSL. There may be some logic errors or bugs in the code of those software. We, as users just download and use SSL clients, and trust them for our security. There is also another issue, related with sub-protocols inside SSL. For example there may be some active attacks targeting SSL handshake and dictate the cipher suites to be weak ones.

Thirdly, there may be attacks to Web PKI since it inherently has some problems due to its design. There are lots of CAs/sub-CAs, and their ability to issue certificate to any domain name without the permission of the domain owners, is the core of the problems defeating Internet security.

Finally, there may be attacks to SSL client software, and root stores keeping CA public keys. The certificate validation is done with SSL client software, therefore if it generates a wrong validation message to the user, user may choose the wrong action. Also an attacker can add a fake CA certificate to clients' root store after gaining control over client computers with a malware.

In this chapter, we will give brief information about those issues, but we will mainly focus on and give detailed information on attacks about SSL authentication piece, Web

PKI, and solution to such attacks in the rest of the thesis[1], since today, looking to the recent incidents, they are the main problems defeating Internet security.

2.1 SSL Threat Model

SSL provides us security in terms of confidentiality, integrity and authenticity in the Internet communications. The attacks for SSL target such features.

Attackers always tend to use the weakest part of the target system. For SSL, their main target is not cryptographic algorithms. The attacker wants to reach his/her aim fast, so it is not logical for him to try to break AES or RSA algorithm which will last years if secure configuration parameters of them are used. Hence, as Adi Shamir said "Cryptography is typically bypassed, not penetrated". Instead of trying to make a brute force attack for a key used in SSL, the attacker may try to breach into the server and steal the key! Similarly, exploitation of implementation bugs to bypass authentication and encryption is a better option for attackers. This leads to an important point: The cryptographic primitives we use today is secure, but there are problems with their implementation and the rest of the system.

Attacks are divided into two types: Active and passive. In active attacks, the attacker controls a device on the network between client and server such as a switch or router. He/she then inserts or modifies the packets to reach his/her goal. Active attacks are very powerful, but they are very difficult to implement since they require much more software and hardware. Usually only targeted attacks are done in this type because modifying and re-routing large amounts of packets is difficult to do. In passive attacks, the attacker only reads the packets in transit, and tries to decrypt them afterwards. It is easier and can be done secretly but less powerful than active attacks. In such attack, the option for the attacker is only collecting the packets and analyse afterwards.

Although there are many issues for SSL security, one of the best figure[2] showing them together can be seen in Figure 2.1.

Now we will continue explaining the issues about SSL security in practice, categorized under four titles: Cryptography related, protocol related, implementation related and Web PKI related.

[1] There are lots of issues in terms of SSL security, such as cryptographic, protocol related and implementation related topics, thus we can not cover them in detail here. There are many books about SSL security, for a comprehensive and up-to-date one, see: Bulletproof SSL and TLS, 2014, by Ivan RISTIC. We, as a study subject in this thesis, chose the problems about SSL authentication that were exploited many times as seen in recent years' incidents.

[2] Source: `https://www.ssllabs.com/downloads/SSL_Threat_Model.png`

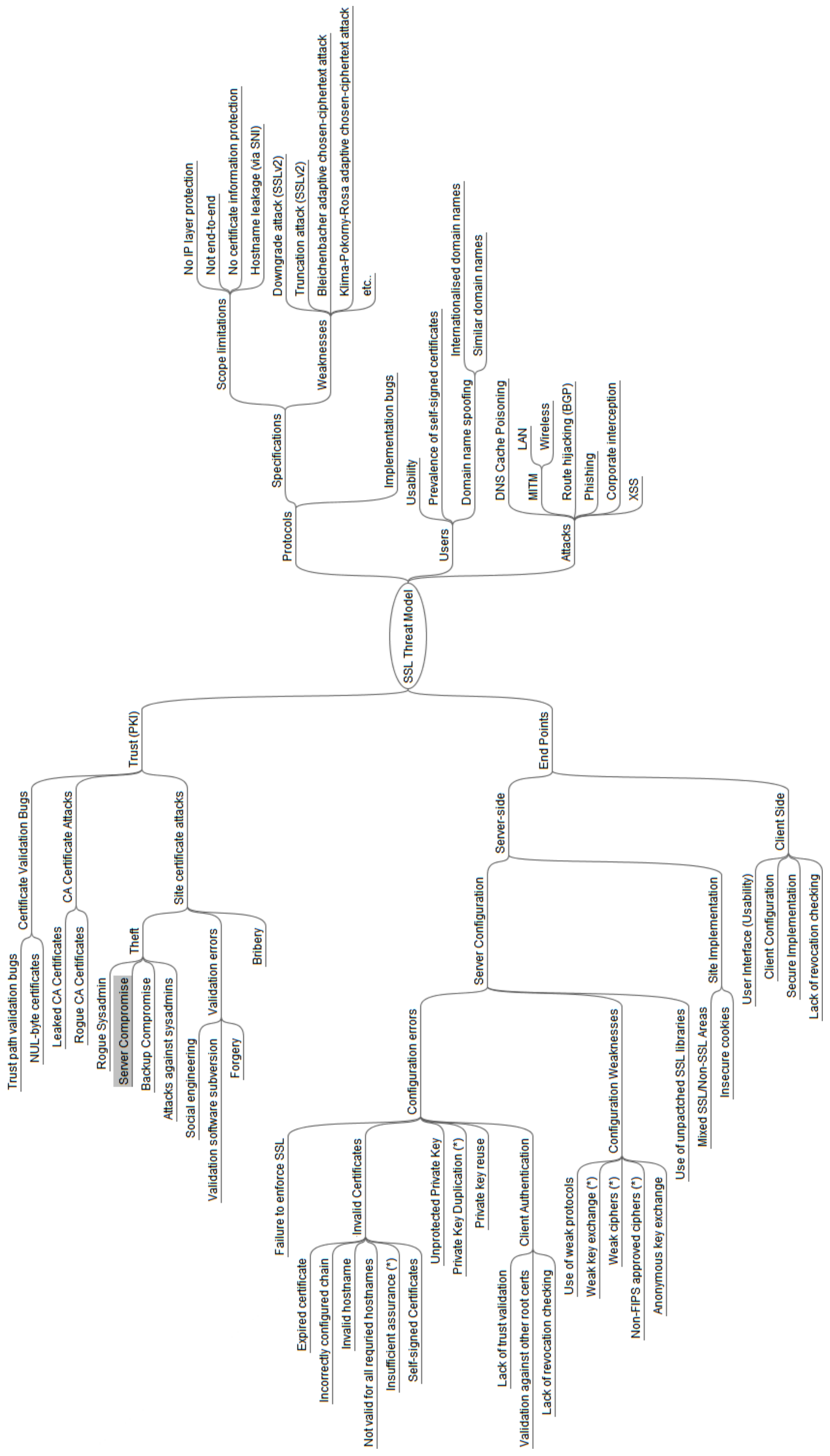


Figure 2.1: SSL Threat Model.

2.2 Cryptography Related Issues

Attackers can use the weaknesses in cryptographic primitives used by client and the SSL server. Cryptography improves day by day but so do cryptanalysis. For example, once accepted very secure, DES block cipher and MD5 hashing algorithm are now accepted insecure. Since the SSL server and SSL client software configuration parameters remain mostly unchanged, an insecure algorithm is sometimes used in SSL [10].

According to a study in 2007, 4 percent of the websites still offer RSA-512, which can be cracked nowadays in just hours with 100 dollars budget[3]. To be secure against such issues, secure algorithms must be used. Currently, the recommended cryptographic primitives are; AES-128 for secret key cryptography, RSA-3072 for public key cryptography/digital signatures and SHA-256 and above for hashing.

RC4 is a stream cipher developed by Ron Rivest in 1987. It was a popular cipher suite for SSL connections because of its speed. Like all stream ciphers, RC4 takes a short (e.g. 128-bit) key and generates a long pseudo-random keystream from that. The message is then XORed with the keystream to obtain a ciphertext. RC4 has a weakness in its design, allowing known-plaintext attacks[4]. It has been proven that RC4 biases in the first 256 bytes of a cipherstream can be used to recover encrypted text. If the same data is encrypted a very large number of times, then an attacker can apply statistical analysis to the results and recover the encrypted text. While hard to perform, this attack shows that it is time to remove RC4 from the list of trusted ciphers. The practical attack using RC4 weakness in SSL requires the web browser to transmit many HTTP requests with the same cookie. The attacker can recover whole plaintext using 2^{30} pairs[5]. In ideal situations, this requires three months. As of February 2015, the IETF explicitly prohibits the use of RC4 in RFC 7465.

An attack, using MD5 weakness, was shown in 2008 by Sotirov [11]. He was able to create a valid CA certificate for SSL after a massively parallel search for MD5 collisions. His study was an improvement on the similar study of Lenstra and Weger [12]. This shows us that MD5 must be replaced by secure alternatives such as SHA-256.

In TLS 1.0 and earlier, cipher block chaining (CBC) mode of symmetric ciphers were used. Since CBC with predictable IVs is not secure against chosen plaintext attacks, the attacker can inject plaintext into a TLS connection and after observing the ciphertext, he/she can determine some information about the rest of the plaintext. This was the weakness used in BEAST attack. BEAST was demonstrated by Duong and Rizzo at the Ekoparty security conference in Buenos Aires in 2011 using a Java applet. It exploits the aforesaid CBC issue, and requires the attacker to control the network near

[3] According to another study by Shamir and Tromer in 2013, with approximately 1 million dollars budget, a 1,024-bit key can be cracked in a year. Although such amount may be seen unaffordable, for intelligence agencies, it is not true. Those agencies spend lots of money for projects like that. This shows that, in addition to RSA-512, RSA-1024 can easily be broken by such agencies, and what is surprising that we still use RSA-1024 in many SSL servers today!

[4] Source: `http://blog.cryptographyengineering.com/2013/03/attack-of-week-rc4-is-kind-of-broken-in.html`

[5] Source: `https://cryptanalysis.eu/blog/2013/03/15/ssltls-broken-again-a-weakness-in-the-rc4-stream-cipher`

the target computer. Before TLS version 1.1, instead of using a new random IV for every TLS message, the ciphertext of the last block of the last message was used as the IV for the next message. Here's why that's bad. The IV needs to be something that an attacker cannot predict. If its known that you are going to use IV x for your next message, and attacker can trick into sending a message that starts with a plaintext block of [(NOT x) x or y], then we will be encrypting y for the attacker[6]. The issue is fixed in TLS 1.1.

On October 14, 2014, a vulnerability was discovered in the design of SSL 3, similar to padding attack for CBC mode of encryption. It is called POODLE. An attacker only needs to make 256 SSL 3 requests to reveal one byte of an encrypted message. This vulnerability exists in SSL 3 but since SSL 3 is a failback mode when handshakes with newer versions of TLS fails, it can be said that using newer versions of the TLS protocol can not save us in that case. Only way to fix the problem is disabling SSL 3. Unfortunately there is no patch for POODLE other than that.

Another issue is RSA. Although it is the most used public key algorithm in SSL, after the emerge of factoring algorithms such as *Number Field Sieve*, there is a discussion about whether to use it or use its popular alternative: *Elliptic Curve Cryptography*. Elliptic key cryptography, although not as fast as symmetric algorithms, is faster than RSA, and achieves same levels of security with RSA, with smaller key sizes. Today, support and usage of elliptic curve cryptography is getting more widespread. Choosing RSA key length as 8192-bits or more, may seem an improvement, but since the computation time in signing/verifying and encryption/decryption will increase, it is not desired.

As seen, there are not many weaknesses in terms of cryptography. If any, they can be overcome with using the well-known, globally accepted secure algorithms with suggested key sizes, and using up-to-date SSL software libraries.

2.3 SSL Protocol Related Issues

Most of the web sites use HTTP, while the ones requiring secure communication uses HTTPS, namely, SSL protocol. Hence web servers and user agents support them both. This option variety enables the attackers to make a protocol downgrade attack. In such attack, an active attacker changes the HTTPS protocol request of the user to HTTP. The attacker may also add an icon to the browser that is a sign of HTTPS traffic. Users mostly do not notice such icons and eventually the communication to the web server is done via unencrypted HTTP, although the users think it is HTTPS. Such attack is called *SSL Stripping* and it can be implemented with the popular *sslstrip* tool developed by Moxie in 2009[7].

The primary defence to such attack is to use Strict Transport Security (HSTS). It allows browsers to learn the addresses which will only be accessed via HTTPS. Such policy

[6] Source: `https://blog.torproject.org/blog/tor-and-beast-ssl-attack`
[7] For details see: `http://www.thoughtcrime.org/software/sslstrip/`

also can be defined dynamically using HTTP headers [13]. When a browser receives this header, it will stop any communications to occur in HTTP to the specified domain, and will redirect to HTTPS.

The cipher suite to be used to secure the communication is negotiated during the SSL handshake. In SSL 2.0, an attacker could downgrade the strength of the cipher suite to the weakest acceptable (for example to RSA-512) by both parties. This issue is fixed in SSL 3.0[8].

The data compression is an optional feature of SSL. The length of a compressed record is not obfuscated by SSL, hence an attacker that injects some plaintext into the SSL connection and then noting the record length, can deduce some information about the plaintext. An attack example of that issue is CRIME[9], used a similar way with BEAST to recover secret information from a cookie. Note that, the authors of the BEAST and CRIME are the same people. As a solution, compression feature is not available in current SSL.

In 2013, another attack, BREACH[10] was announced. It was built on the CRIME and can acquire sensitive data from SSL traffic in as short as 30 seconds, if the attacker can deceive the victim to visit a malicious web link. All versions of TLS and SSL are at risk. Although disabling compression in SSL help us to defend against CRIME, such way is not fruitful for BREACH since it exploits HTTP compression which cannot fully be turned off. The mitigations to BREACH are; disabling HTTP compression, randomizing secrets for each request and length hiding. As noticed, there is not a short and easy way to defend against BREACH.

2.4 Implementation Related Issues

SSL software libraries may suffer from certificate validation implementations. Such issues include;

- Verification of certificate chains,
- Verification of domain name,
- Checking revocation status of certificates, via OCSP or CRLs,
- Handling X.509 extensions.

In some versions of such libraries, its developers sometimes make mistakes in the code. Such mistakes include wrong domain name extraction in string operations of programming languages, wrong error handling, misleading alert messages and erroneous encoding and decoding.

[8] Source: `https://eprint.iacr.org/2013/049.pdf`

[9] For details, see: `https://community.qualys.com/blogs/securitylabs/2012/09/14/crime-information-leakage-attack-against-ssltls`

[10] For details, see: `http://breachattack.com`

Heartbleed[11] may be the prominent attack in this category, which occurred due to implementation mistake in OpenSSL library. It was discovered by a team of security engineers from Google. Heartbleed allows attackers to read the memory of the systems protected by the vulnerable versions of the OpenSSL, and access information such as server private keys and passwords stored in memory. OpenSSL version 1.0.1g that was released in 2014 fixed the bug.

SSL requires several random values, including the secret keys to be generated for each session. Hence, a secure random number generator (PRNG) must be used. For example, in Netscape Browser, the random number generation had relied on just few, and predictable values: Current time, process id and parent process id. But, after years, there are still 0.5 percent of SSL certificates having recoverable RSA private keys due to weak PRNG.

Moxie showed a vulnerability for certificate basic constraint validation of Internet Explorer (IE). Internet Explorer did not check 'CA:TRUE' information of the certificate. Thus, sub-CA certificates could generate a sub-CA certificate and use it to fool IE. Note that, it was an issue for a specific and very old version of IE, but shows us that such mistakes can happen.

2.5 Web PKI Related Issues

For SSL usage, firstly, we need to know the public key of the target website. If an active attacker can deceive us with his public key, we will be talking to him instead of the web server. This is possible with a MitM attack and it is easy with the tools like *sslsniff* today. Cryptography can not help us for MitM attacks because the attacker is behaving as an intermediary and encrypting/decrypting all the traffic passing over him [14]. The process is depicted in Figure 2.2.

In many cases, MitM attacks require access to the communication infrastructure. Whoever has access to the network devices such as routers can see/interfere the packets. Wireless networks without authentication are especially vulnerable because anyone can join such networks.

We solve MitM attack issues by Web PKI. One of the hundreds of trusted CAs sign the certificate of a website and while SSL handshake, our web browsers authenticate the server with the preloaded public key of the CA that signed the certificate.

The weaknesses of the Web PKI for SSL have been studied in many papers. Eckersley and Burns made similar researches [15] with Holz and Braun [16] on a huge list of SSL-enabled websites, analysing SSL primitives in use, noticing that mostly authentication errors occur. Sunshine made a study showing that users mostly ignore browser SSL warnings [17]. Soghoian and Stamm showed that some governments may force CAs to issue forged certificates for them to make MitM attacks [18]. Similar type of MitM

[11] For details, see: `http://heartbleed.com`
[12] Source: `http://securityaffairs.co/wordpress/24895/hacking/detection-mitm-forged-ssl-certificates.html`

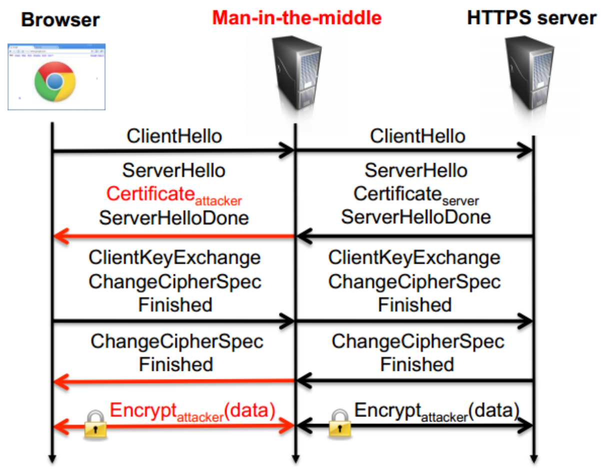


Figure 2.2: MitM scenario for SSL[12].

attacks are caught by Google, while French cyberdefense agency, better known as ANSSI, is using the certificates issued for some of Google domains, by a sub-CA linked to them [19].

Moore and Ward expressed a weakness in certificate issuance about wildcard (*) usage for IP addresses [20]. According to RFC 2818, wildcards are forbidden for IP addresses. The authors found some browsers accepting wildcard certificates issued for IP addresses.

Moxie defined an attack for certificate revocation [21]. While using OCSP, the response of the OCSP server contains a field called 'responseStatus' which is not signed. Hence, an active attacker can change the response of the server with 'trylater'[13]. Due to such response, the clients will continue without revocation check, and may use the certificates already revoked. What we learn from this, is the fact that the integrity of every sensitive area in the certificate or protocols must be protected by digital signatures.

The problems with Web PKI are follows:

- **Main problem in the design of Web PKI:** Any CA/sub-CA can issue certificate to any domain name without the permission of that domain owner. This means, for example, a sub-CA in Brasil can issue a certificate for a domain in Turkey, and that domain owner may not be aware of this. This leads to a big flaw in Web PKI: An attacker who can steal just one private key of a CA/sub-CA, gets able to make MitM attack to any SSL server. What is more, anyone with a good budget can establish a CA, and use it for malicious purposes.
- **No trust agility:** Client root stores contain hundreds of CA certificates. Clients trust a CA or distrust, no middle decision. This means, whenever you trust a CA, you trust all the certificates have a chain up to that CA! The opposite is also true. Hence, when you want to distrust a CA due to its misbehaviour, you will need

[13] Since messages communicated via OCSP are usually communicated over HTTP, it is possible.

to remove its certificate from your trust store, which will cause authentication problems for the certificates, issued by that CA or its sub-CAs.

- **Revocation problems:** Revocation methods, unfortunately does not work in most cases. Looking to the recent years' CA compromise, the revocation of the problematic certificates was delayed, and could not be spreaded to all SSL clients in needed time.
- **Insufficient CA verification for DV certificates:** Issuance of DV certificates are usually carried out using email. It is easy for an attacker to obtain a fraudulent DV certificate after he/she accessed to the mailbox of that domain owner.
- **Security of CA private keys:** The attacker can choose a CA/sub-CA, that is weak to defend itself against cyber attacks, and after accessing the private key of that CA, he/she can generate a certificate to any domain name. Therefore, CAs must keep their private keys in very secure manner, also must defend their infrastructure for any kind of cyber attacks with the use of tools such as intrusion detection/prevention systems.
- **Mishandling certificate warnings:** Certificate warnings are another big issue in current system. SSL clients and libraries do not handle such warnings in a secure manner. They sometimes even skip validation. Web browsers, once detected an invalid certificate, give warnings to the users, but users can continue the communication even such warnings exist. Furthermore, according to some studies, more than 30 percent of the users tend to continue in such cases, which completely defeats SSL security[14].

2.5.1 Recent Attacks on Web PKI

Several CAs were faced security attacks especially in recent years. The attackers tried to create forged certificates for popular domains like 'google.com' and 'yahoo.com'. Unfortunately, the attacks were detected very late, sometimes after months. Here are the publicly known attack incidents [9] on Web PKI in recent years, ordered by the most recent at first[15]:

- **National Informatics Centre (NIC) of India:** A Google blog detected unauthorized digital certificates for some of Google domains, in addition to some others such as Yahoo, on July 2014. They reported that the certificates had been issued by a sub-CA from India. That sub-CA was authorized by Indian Controller of Certifying Authorities. After that, Google revoked the sub-CA certificates of NIC on July 3. Furthermore, due to such misbehaviour, Google restricted the root-CA to issue only for few domains having a suffix '.in'.

[14] For details, see: `https://www.usenix.org/legacy/event/sec09/tech/full_papers/sec09_browser.pdf`

[15] Those are the known incidents. There may be more events that were not disclosed by the involved parties, not to degrade their commercial or organizational reputation. Another issue is stated by Moxie Marlinspike: "What happened to Diginotar is the kind of thing that happens every day. It was an accident anyone ever noticed. If the hackers hadn't been stupid, no one would have ever noticed."

- **ANSSI:** On December 2013, another Google blog detected unauthorized certificates for several Google domains. The certificates had been issued by a sub-CA whose root-CA is a French CA, ANSSI. It was noticed that the certificate was used to decrypt the traffic in a private network. Google revoked the certificate and decided to limit ANSSI to be able to issue certificates only for top-level domains.
- **Trustwave:** Trustwave stated that they had issued a subordinate CA certificate for a firm in order to inspect the SSL traffic. Mozilla said that this behaviour may violate Mozilla's policy but took no action afterwards. Trustwave also stated that they would not issue a certificate for that purpose any more.
- **DigiCert Malaysia:** They were detected to issue 22 certificates having RSA 512-bit keys. Afterwards, DigiCert Malaysia's sub-CA certificate was revoked by Entrust on 2011[16]. Entrust informed the browser vendors for the issue. Afterwards, Microsoft, Mozilla and the other vendors revoked the certificates in their browsers.
- **Turktrust:** A Turkish CA, detected by Google to have issued 2 sub-CA certificates to EGO, a corparation of the municipality of Ankara city, in 2013. The issued certificates were used to decrypt SSL traffic. The issue was detected by Chrome, using the pinning feature for Google domains. Google revoked the sub-CA certificates, and alerted the other browser vendors. Turktrust stated that they had mistakenly issued those sub-CA certificates instead of regular SSL certificates. Due to that mistake, Google decided not to show green address bar for the EV certificates issued by Turktrust. Google also said that they will continue to allow Turktrust certificates in SSL connections.
- **DigiNotar:** A Dutch CA, DigiNotar, was founded in 1998. In 2011, an attacker breached DigiNotar CA infrastructure and created a wildcard certificate for '*.google.com'. The attack was discovered by Chrome's pinning feature. DigiNotar noticed the intrusion and admitted that more than 530 certificates are created by the attacker, including '*.microsoft.com', '*.skype.com' and 'twitter.com'. All browser vendors removed the DigiNotar root certificate. Hence users could not access to many Dutch web sites. As a result, DigiNotar went bankrupt in the same year.
- **Comodo:** An American CA, detected an attacker creating 9 certificates including 'mail.google.com' and 'www.google.com', in 2011. They immediately revoked these certificates. Comodo handled the situation well, quickly detected the breach and informed the public. They are still in business.

[16] Source: `https://threatpost.com/malaysian-ca-digicert-revokes-certs-weak-keys-mozilla-moves-revoke-trust-110311/75847`

2.6 Lessons Learned

As seen in above sections, the attacks on SSL can be made by:

- Stealing the server private key and using that with the server valid certificate,
- Stealing a CAs private key and using that to issue a new valid server certificate,
- Getting a CA to issue a new valid server certificate, by fooling/hacking/cooperating the CA,
- Installing a new trusted root certificate in the client, and then creating a new server certificate without help of any CA. New trusted root certificates can be installed by controlling/hacking the client, or fooling the user,
- Using self-signed certificates and expecting the user to ignore the warning,
- Taking advantage of some implementation weakness in the SSL client software, like the OpenSSL bug.

In order to resist such attacks, precautionary measures must be taken by all the parties in Web PKI: The users, server owners, CAs, browser vendors etc. Although the attacks mostly happen on the authenticity piece, cryptographic primitives and protocol design have been attacked as well.

Summarizing the lessons learned, leads us to some basic hints:

- Theoretical attacks can turn into practice as seen in the sslstrip example, hence we should think every attack type as possible.
- It is very important to use reliable and secure cryptographic primitives.
- The software implementations of SSL specifications have to be the same as it is written in that documents, no developer addition.
- Security of all SSL features must be taken into consideration: Confidentiality, authentication, integrity.
- Flexibility in design mostly means additional risks as seen in cipher suites option example.
- CAs and SSL servers must secure their private keys, furthermore, they must defend their infrastructure to any kind of cyber attacks by using systems like Intrusion Prevention Systems (IPS).
- Clients must secure their computers/devices where they make SSL connections by using tools such as antivirus and firewall software.
- CAs/sub-CAs must be audited firmly and any misbehaviour should be punished severely.

- The trust decision in the event of certificate warning must be handled correcly by SSL client software. The user only may be able to choose the secure way.

For detailed attack types information and more lessons learned, please refer to the study of Meyer and Schwenk [22].

2.7 Summary

Providing a bullet-proof SSL is not easy due to the variety of the attacks for SSL. Attack categories for SSL are about: Its cryptographic primitives, its sub-protocol design, implementations in SSL libraries and Web PKI.

When people talk about SSL security, they mostly think that its security relies solely on the cryptography, and it is secure nowadays. But, as seen in this chapter, even the cryptographic primitives in SSL have been attacked so far.

Attackers mostly prefer the weakest part of the target system and try to reach their aim fast. Therefore their primary target is the easiest and the weakest part of SSL: Its authentication piece.

We noticed that, there is not much work for securing SSL for MitM attacks. We think that it is very important and should receive adequate attention by the researchers.

CHAPTER 3

IMPROVING SSL SERVER AUTHENTICATION

As explained in the previous chapter, current Web PKI for SSL is prone to MitM attacks, usually made by using forged certificates. Hence, researchers, standardization organizations like IETF and browser vendors like Google made some proposals to solve the problem.

Some of the proposals suggest big changes in the current Web PKI such as not using CA model any more, although most of them prefer minimizing the problem by requiring only small changes.

This chapter gives gives a brief overview about the proposals for solving the weakness in SSL server authentication. Although more than ten proposals exist, we will focus on ten of them which can be categorized under five titles:

- Pinning proposals,
- Network-perspective providing (notary) proposals,
- Transparency-providing proposal,
- DNSSEC-based proposal,
- Certificate collecting proposals.

Now we will continue with explaining the proposals by giving information about the main idea they are based on, their design and pros/cons[1].

3.1 Certificate Pinning

To recall, SSL certificates are validated by checking the certificate chain: LeafCert, signed by IntermediateCACert, signed by CACert. CACert is shipped with operating system code, which may be accepted as a secure way of delivering certificate to the

[1] The written information, of course, may not be covering all aspects of the proposals, for details please refer to project RFC documents or websites, specified in the references section.

clients. Therefore the main trust is based on the root-CA certificates stored in clients' root store.

Unlike the current certificate validation mechanism, Certificate Pinning simply stores a SSL server name information such as 'google.com' along with the certificate/public key information of that server, into the client cache, then, while starting connection to the server, checks the SSL certificate that the server presents whether it matches with the stored one. If more than one certificate is used by the SSL server, all of them can be pinned. The latter is also referred as storing the certificates in the *pinset*.

The most basic way of pinning the certificates to SSL client software such as browsers, is adding them into the software code, similar to the way that root-CA certificates are added to browser/operating system code. It is reasonably secure since attackers will not be able to modify easily that information, but on the other hand, it is not easy to change or revoke the certificates when needed. This way is used by Google Chrome browser. Public key hashes of some popular websites, mostly owned by Google, are preloaded into the Chrome.

Another way is *pinning at the first encounter* with a certificate. You assume that at the time of pinning there was no attack and the SSL certificate presented by the server was the correct one. You then pin that certificate to that server name at client. This is a bit risky way of doing a pinning, but can be considered a logical assumption, since the low probability of the attack at the time of pinning operation. This way is also known as 'trust on first use (TOFU)' method.

Pinning provides mainly key continuity, more secure authentication of the server and less attack surface. Note that key continuity is also used in SSH protocol where keys are pinned with server names when seen first time, then checked at other visits to that server. Pinning also reduces the threat of a rogue CA and CA compromise by taking the CAs out of the certificate validation process [9].

Currently, there are two popular proposals in certificate pinning category:

- Trust Assertion for Certificate Keys (TACK),
- Public Key Pinning Extension for HTTP (PKPE).

3.1.1 Trust Assertion for Certificate Keys (TACK)

The general pinning scheme is good but has a weakness: It is not flexible [23].

Thus another approach is needed to allow the SSL servers to change their certificates. At that point, TACK comes into the play. It mitigates the problem of certificate changing of a SSL server by *pinning dynamically*.

TACK is "a proposal for a dynamically activated public key pinning framework that provides a layer of indirection away from Certificate Authorities, but is fully backwards compatible with existing CA certificates, and doesn't require sites to modify

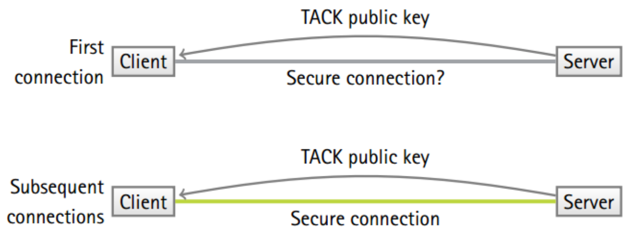


Figure 3.1: TACK pins usage.

their existing certificate chains." as stated in its website [24]. It is written by Moxie Marlinspike and Trevor Perrin in 2012.

In TACK, clients will pin a signing key (TACK signing key: TSK) which is created by SSL server operators and used to sign server's SSL key (a similar way to CA signing of their issued certificates), instead of pinning the SSL key. So server operators will easily be able to change the SSL certificates without a need for the clients to change the pins.

In short, TSK is long-lived key whereas SSL key is short-lived. Note also that, since pins are signed with TSK instead of CA keys, there will be no need to trust to CAs.

TACK fields consist of mainly: Public key of the TSK that has signed the tack, expiration time (in minutes) and signature by the tack's TSK [25].

TACK also defines a mechanism for pin activation. In the "ClientHello" phase of SSL handshake, a client may request the server to send its TSK public key and signature. The client that supports TACK, declares that support by giving an empty tack extension. If the server wants to give its tack, which is simply the pin of the server's public key signed by server's TSK, uses the same tack extension to respond. The client notes the pin in the first encounter, but activates it after the second time it is seen (Figure 3.1). It can do it for a time limit also. By time-limiting, the potential impact of a wrong pinning decision is lessened.

TACK pins are specific to application protocol. For example, a pin of HTTPS at 'example.com' implies nothing about SMTP at 'example.com'.

TACK pins can also be shared, for example, a TACK client may share the pins which he used or discovered on the internet, through some 3rd-party for the sake of other clients. But unfortunately the specification does not define the details about the infrastructure or the protocol needed for it [26].

Pros:

- Flexibility in changing/revoking the server certificates used for pinning,
- Recovering general weakness in pinning scheme,
- Can be used to secure any application layer protocol.

Cons:

- Requires change in SSL protocol,
- It is still in very early stage and not widely-accepted, although some good work is done on its website, such as software libraries.

3.1.2 Public Key Pinning Extension for HTTP (PKPE)

PKPE project [27] was started by Google, and has been developed since 2011. It focuses mainly on the pin delivery problem. It suggests to deliver the server pin values in HTTP headers (in a new HTTP header). When the client sees the pin value of the SSL server in the HTTP header, it pins that value and checks at subsequent SSL visits to that server.

Pins are created by specifying the hashing algorithm and a fingerprint computed using that algorithm.

For example:
Public-Key-Pins: max-age=2592000; // in seconds (here 1 month)
pin-sha256=fTafFhjter64sfwr/+sgdfTsdsdfgdfgsddsdsfsdfdfdsArgffff=
pin-sha256=dfdsddS5676446fsg+sdsKLIkasawwPergi16456sdsadas=
report-uri=example.com/pkp-report

The only hashing algorithm supported in PKPE is SHA256; the sha256 identifier is used as seen above example. The fingerprints are encoded using Base64. To enable pinning, you must specify max-age period in seconds and give at least two pins [9].

PKPE specifies a mechanism for user agents to report pin-validation failures. This feature is available using the *report-uri* parameter, which is simply the report's receiver server. The report is submitted using HTTP POST request.

If the subdomains of the host are needed to be added to the same pin, *includeSubdomains* parameter must be used.

If a client receives a certificate that mismatches the pinned one, the client should report about this pinning error to the server defined in report-uri parameter. But if the attacker is directing network traffic in the MitM attack, that information may not be able to reach to the server defined in report-uri parameter. To overcome this issue, the client periodically tries to send that information hoping the attack will end after some time and the information will reach to target.

The main missing feature of the proposal is that it can not be used for application layer protocols other than HTTP. But recall that, SSL is used to secure all application layer protocols, not just HTTP. Hence, it can not be an overall solution for us.

PKPE can not protect us against MitM attacks for the first connection to the server, that is, it suffers a bootstrap problem. The assumption of non-attack while pinning about a server is logical but risky. To overcome that, it may be thought to preload all pins similar to CA certificates' preloading into client software code, but estimating the number of SSL servers in use, it is definitely hard to manage this.

Not many websites publish pins via HTTP headers currently. The development of PKPE is planned to finish in very near future and then is expected to be adopted fast by other browsers in addition to Google Chrome.

Pros:

- Continuity of SSL server keys,
- Easy to implement (e.g. with browser add-ons),
- Allows using self-signed certificates.

Cons:

- *Poison pin* problem (possibility of pinning the wrong key),
- Applicable to use only for HTTP application layer protocol.

3.1.3 Final Remarks for Pinning

Pinning is still not widely used because of the issues stated above. But this is not an obstacle for us to use it in some specific cases. It may be preferred for high-risk sites where extra security is desired, as Google did for some of its propriety servers. For instance, nearly 300 pins, mostly for the websites owned by Google, are added to the Google Chrome browser within its code, starting from version 13. The whitelisted public keys for Google include Verisign, Google Internet Authority and Equifax. For example, Chrome will not accept certificates for Google properties from other CAs [28].

Also, a mobile platform, Android, has started supporting public key pinning by version 4.2 [29].

3.2 Certificate Transparency (CT)

The purpose of Certificate Transparency is making all the parties (CA, website owners, clients etc.) in Web PKI, to able to see the SSL certificate system in use, transparently, and then, in case of problems such as certificate misissuance, detect and take action in very short time (in hours instead of months). Its purpose and capability is not problem prevention. It only provides faster problem detection. It's managed by Google and started at 2012.

CT is based on the idea of adding all SSL certificates to a public log that anyone can see/monitor, and SSL clients rejecting a certificate if it does not exist in that log. If all

SSL clients obey this rule, an attacker will have to add the fake SSL certificate to log server, the action which will be easily detected by SSL server administrator by using tools in CT, explained in following paragraphs.

Its project team emphasized the number and severity of certificate-based threats' rise in recent years and decided to do something about it, with the CT project [30].

Their aim is; to recover faster in case of an incident and put domain administrators into the tasks more, since they can know their certificates better than anyone, and also to design a system that can be benefited even some parties participated.

CT has 3 goals as explained in their project website [31]:

- Preventing the cases where CAs issue certificates without the knowledge of domain owners,
- *Impostor discoverability* feature: Providing a transparent system for each party in Web PKI, including monitoring and auditing tasks, so that all the parties will be able to see the certificates in use and will detect malicious certificates faster,
- Protecting users from malicious certificates used in in SSL connections.

CT plans to reach that goals by the help of three tools: CT logs, monitors and auditors.

CT logs; are public, verifiable and in a append-only structure which makes it impossible to change/delete a certificate added before, without being noticed. Not only one log server is expected, ideally between 10-1000 log servers can suffice to provide 7/24 availability. Google is one of the log operators.

CT log servers assure being append-only by storing the added certificates in binary Merkle hash tree structure. This tree structure allows us cryptographically to be sure that it is working in append-only structure and whether a queried certificate exists in the log or not.

The root of the tree is called Merkle tree hash (MTH). When log server signs the MTH value, the generated value is called signed tree head (STH). Periodically log server writes the certificates, waiting in the queue, to the server at one attempt. For that, log server generates another tree, adding the previous tree as the left main branch, all newly added certificates as the right main branch, and calculates new tree root value, MTH, and signed value, STH, again. SHA-256 hash algorithm is used while generating the tree.

CT log servers prove us their append-only property and existence of a queried certificate by 2 different proofs: Merkle consistency proofs and Merkle audit path.

- **Merkle consistency proofs** is used for proving the append-only structure. Using the values of previous and current STH, you can calculate if current tree contains the previous one. An example is shown in Figure 3.2 (The consistency proof is the node hashes: k, l, and m).

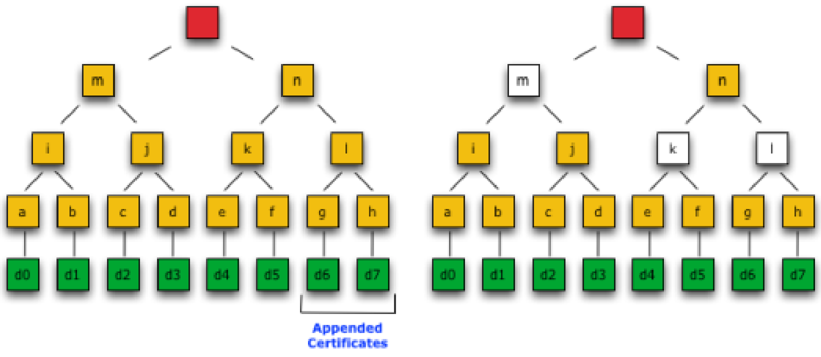


Figure 3.2: Merkle consistency proof example [2].

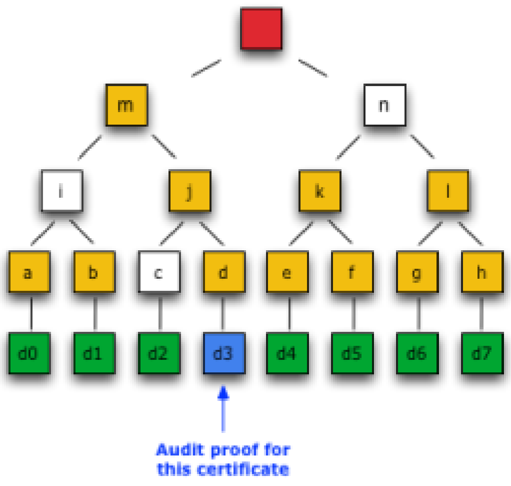


Figure 3.3: Merkle audit proof example.

- **Merkle audit path** is used for checking the existence of a certificate in the log. Log server gives a path for the queried certificate, then we can calculate the MTH value by using that path and queried certificate-hash. If we can find the same MTH value as the current tree MTH, we will be sure that the certificate exists in the log. An example is shown in Figure 3.3. (The Merkle audit proof is the node hashes: c, i, n).

Log server operators must publicly share the log information and can not declare any precondition for sharing. This is a fundamental rule of CT logs to be publicly monitored and audited.

Similar to root-CA certificates, certificates of log servers must be added to root stores of the client computers/devices, most probably via OS/browser codes.

To issue a SSL certificate to a client, CA will, additionally send the generated certificate to a log server. Log server will respond a signed timestamp (SCT) value, simply

[2] Source: `http://www.certificate-transparency.org/log-proofs-work`. Note also that, all figures used in CT section in this thesis are taken from CT project website.

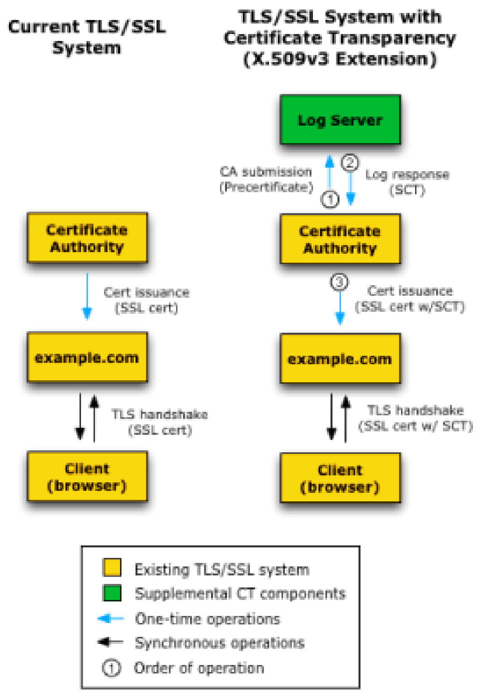


Figure 3.4: Certificate verification process in CT.

a timestamp signed by log server about nearly 100 bytes data, a proof of certificate addition to the log. CA will add SCT to the certificate, sign it, and issue to the client. A SCT is in fact a log server's promise to add the certificate to the log in a specific time known as maximum merge delay (MMD) which is usually one hour.

At the time of SSL handshake:

- SSL server will send the SCT value along with the certificate to the SSL client (At least 3 different SCT is suggested to use),
- SSL client will verify the SCT value by using the log server public key stored in client root store. If the SSL server does not issue a SCT, or if the SCT is not verified, the client software will give a warning to user and user should refuse the connection. If more than 1 SCT exists, SSL clients will require at least one of them to be trusted. A visual explanation of the process can be seen in Figure 3.4.

Sending the SCT value to the SSL client in above way (adding to the certificate) seems the most logical and easy way but, in its RFC document [32] two other methods are described: sending SCT by OCSP and sending by a TLS extension (Note that these 2 methods require additional work in all SSL servers which is not desired).

Additionally, SSL clients should, asynchronously, audit the incoming certificate whether it exists in the log server records or not. The RFC document also describes that clients should share its view of the logs to other parties, called 'gossiping' but it does not give any information about the gossiping protocol, which may probably be described in future document updates.

It may be thought that log server is similar to CA as being another trusted third party. No, because you do not have to trust a log server all the time, you can monitor its activities by CT's tools, and if you think it is misbehaving, you can remove trust and choose another log server immediately. But remember, it is not the case in the CA model. If you remove trust from a CA in your root store, any website having a certificate chain ending up to that CA will be untrusted and cause a big problem for clients visiting that websites.

Monitors and auditors are the querying and inspecting mechanisms for a log server.

Monitors are small programming libraries which could be implemented by any party: CAs, SSL server administrators and even SSL clients. CAs or server administrators should check the existence of SSL certificates they issued/owned by using monitors with the help of Merkle audit path.

Auditors software will probably be running in a dedicated server, periodically checking the append-only property of the CT log servers with the help of Merkle consistency proofs. Although any party could do auditing, CAs or browser vendors will most probably be volunteers for it since they can afford the needed resources such as a dedicated server.

In fact, the biggest actors in current Web PKI, the CAs, will be volunteers for CT since it will allow them to audit and monitor the security of their issued certificates. CAs will also be monitoring to see other CAs' certificates to inspect missteps.

CT places more responsibility on domain owners and requires that domain owners perform that duty very well. Because CT provides and aims mainly transparency of certificates to public to take action in event of suspicious cases, if the domain administrators (or CAs) do not monitor the domain's certificate in the log, this will make CT useless.

CT in Chrome is available since Chrome version 33. In addition to the basic UI indicator, it is possible to view details about each Signed Certificate Timestamp such as log server information.

To summarize the role of the parties of Web PKI, in the CT:

- CAs, SSL server operators: Adding certificates to the log, sending the SCT value to the SSL client at the time of SSL handshake.
- SSL clients: Verifying the SCT value by using the public key of log server, preloaded to client root store, then asynchronously verifying the existence of the certificate by help of Merkle audit path that log server provides and at last, gossiping their view of log (although not fully-explained in the RFC document).
- Every individual: Checking the log server's misbehaving by help of Merkle consistency proofs.

The CT project team's plans as future work are; More options in hashing and signing algorithms, better handling the issue of log server public key change.

Pros:

- Allow us to see the certificates that are generated for any domain,
- Not so much work on client/server.

Cons:

- Privacy problem: The auditor can see client's browsing history,
- It is unclear; what to do for a misbehaving log server, how to change log server keys and distribute their keys to clients.

3.3 DNS-Based Authentication of Named Entities (DANE)

DANE [33] is an extension of Domain Name System Security Extensions (DNSSEC) to store certificates of SSL servers along with the IP information in Domain Name System (DNS) database. The IETF DANE working group started developing ways to use DNSSEC to improve SSL authentication of domain names in 2009, eventually proposing DANE project.

3.3.1 Domain Name System (DNS)

DNS is used to convert domain names such as 'google.com' to IP addresses such as '100.101.102.103'. The part of addresses are named by splitting by dot, for example, in case of 'www.mail.google.com': '.' is top-level (not seen in the address), '.com' is top-level domain (TLD), 'google' is domain, 'mail' and 'www' are subdomains.

The country codes like '.tr', '.fr' are also TLDs. There are 13 top-level DNS servers all over the world, keeping list of TLD's. When a client make a DNS query for 'www.google.com', his DNS resolver software first asks the IP of '.com' DNS server to one of top-level root DNS servers. After getting its IP, it recursively continues to ask lower domains like asking IP of '.google' to the TLD and so on, as seen in the example on Figure 3.5[3]. Eventually it will learn the IP of 'www.google.com' from '.google' domain DNS server.

DNS is designed in 1983, even before HTTP protocol. Since at that times not so much security issues expected, it did not have and still not have an authentication mechanism, so a popular target for attackers because of its widespread and vital use in the current internet infrastructure. By attacks such as DNS poisoning [34], an attacker can respond

[3] Source: `http://www.uxworld.com/?p=384`

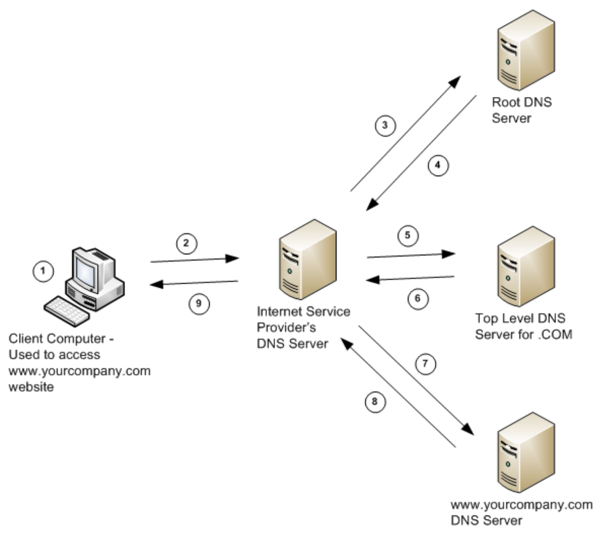

Root DNS Server
Top Level DNS Server for .COM
www.yourcompany.com DNS Server
Internet Service Provider's DNS Server
Client Computer - Used to access www.yourcompany.com website
1
2
3
4
5
6
7
8
9

Figure 3.5: A DNS query example.

to the client who is doing the DNS query, with a malicious server that attacker controls. With that, atttacker will have a chance to do phishing attacks [35] to deceive and then get the login credentials of the clients for critical services like online banking [36].

3.3.2 DNS Security Extensions (DNSSEC)

DNSSEC is developed to prevent attacks on the DNS, explained above. ICANN [37] is leading DNSSEC. It is a new set of protocols that adds DNS the missing property: Integrity checking. It uses digital signatures to assure that the response received from DNS server is the same as with the data published by the domain owner.

DNSSEC adds many digital signatures to the DNS hierarchy. In that structure, each organization along the way must sign the key of the one below it. For example; for the 'www.icann.org' address, '.org' signs 'icann.org's key, and the 'root' signs ".org's key [38]. The signing process which will probably last several years, started at 2010 by signing the root zone. Root zone signing is administered by VeriSign. TLD level signing is in progress and current data can be found here [39].

Note that DNSSEC does not provide secrecy, DNSSEC responses are authenticated only, not encrypted.

3.3.3 DANE's Design

In current Web PKI, it is not possible to know the CA that a domain administrator preferred to buy SSL certificate for his domain. DANE is an exception.

DNS databases store the domain name-certificate binding value, called TLSA DNS resource record (RR) field in DANE protocol. Each TLSA record has 3 fields: 'Usage', ‘Selector’ and ‘Certificate for Association’. These records are stored under the target domain with a prefix that shows the protocol and port.

To summarize DANE’s usage [40]:

- Domain administrators should create a TLSA RR for their domain, which can easily done by using tools, freely available [41].
- Domain administrators should make that TLSA RR information to be added to domain’s DNS zone (signed using DNSSEC).
- Clients, making a DNS query for a web address, should additionally query TLSA RR field, then, if ‘usage’ field in the TLSA RR is 0 or 1; the certificate will be validated as usual, i.e. by validating the certificate chain up to root CA. But if it is 2 or 3, the CA system is out of validation process, and validation is done only with public keys used in DNSSEC. Note that usage types 2 and 3 allow self-signed certificates to be used, which is removing the need to CAs.

For example, if you are running a secure web service at ‘example.com’, and want to tell clients to only accept certificates from Charlie’s CA, you could provision a TLSA record under ‘_443._tcp.example.com’ with the following contents: ‘Usage’ = 1, ‘Selector’ = ‘SHA1 digest’, ‘Certificate for Association’ = ‘Charlie’s certificate SHA1 digest’ [9].

There are 2 standardization documents for DANE. First one [33] is about usage of the TLSA RR and second one [42] is about use-cases of DANE. The DANE working group is regularly updating that documents, named as ‘draft-ietf-dane-protocol’ and ‘draft-ietf-dane-use-cases’.

DANE increases the role of DNS operators and domain administrators. As noted, DANE allows domain owners to specify the CA (by using the ‘Certificate for Association’ field in TLSA RR), which are allowed to issue certificates for their owned resources, and that solves the problem of CAs’ certificate issuance capability to any domain.

The main problem of DANE is obviously DNSSEC deployment. It is not DANE, causing the problem, it is because DANE’s prerequisite, DNSSEC, will be tried to be used first time in that wide manner. DANE needs many changes in current system for example in DNS servers and SSL client software. The needed changes are in terms of software as well as hardware. Noting the difficulty to change the mind of network operators and manufacturers, we can not just rely on DANE to provide the desired highly secure SSL server authentication [43].

Certificate validation will be done as usual when no DANE information exists, so an attacker who is blocking DANE traffic can bypass DANE security. The key verification process require many DNSSEC queries to be done, resulting in a latency, not desired in upper layer real-time messaging/voip protocols like XMPP and SIP [26].

If DANE has aim to replace CA model some day, we will be having a similar problem as now, in terms of 3rd party trust. Currently we have been complaining about CA model as there are lots of CAs and we blindly have trusted every CA forever, but in DANE, we trust DNS server operators that will most probably a potential target for attackers then. Do we believe that DNS server operators will guard to security attacks better than CAs in current system? Well, probably not.

In DANE, there is a single point of failure: the DNSSEC root. One having control over DNSSEC root may use a certificate that is validated and trusted by every DANE client software.

Currently, DANE is not supported by the operating systems. Since DANE is based on DNSSEC, until OS supports it, it is not easy to use them. You must implement DNSSEC yourself in the code. But none of the browsers do that currently. Only Fedora, a Linux distribution, is giving support for DNSSEC starting from 2014, December [44].

Although DANE is not fully adopted by browsers, its functionality can be added via add-ons, as in Firefox [45] and Chrome [46]. These add-ons check the existence and validity of DNSSEC signed DNS records while connecting to SSL-websites.

DANE is not available for many application layer protocols other than HTTP. But there are efforts in the DANE Working Group to extend its usage to protocols such as s-MIME and XMPP.

Pros:

- Strong domain name authentication,
- Allow to define the CA that a domain owner bought SSL certificate from, publicly to clients. So SSL clients will start validating the certificate if it is signed by that CA.
- Allow self-signed certificates.

Cons:

- Requires change in many parties in Web PKI,
- DNS server will be aware of the running server IP as well as port number that may be privacy issue.

3.4 Proposals on Using Network Perspectives (Notaries)

The proposals under this title are; Perspectives, Convergence and DetecTor project [4]. First 2 are very similar and based on taking the SSL server authentication task from

[4] There is also another project based on notary idea, from the author of DetecTor: Mutually Endorsing CA Infrastructure (MECAI) project. It aims to improve the Web PKI system with notary idea and defines extra-needed work as CAs' duty. With its usage, it will not suffice to hack just one CA. It provides mainly privacy and performance improvements. Since it did not progress so much after it was proposed, we did not include details about it in this thesis.

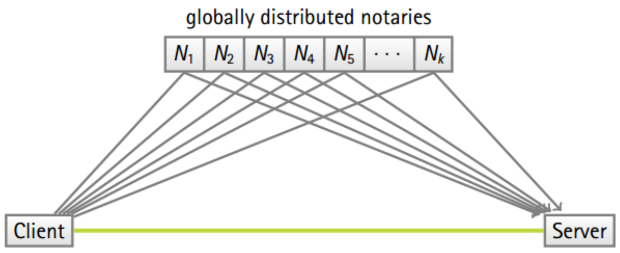


Figure 3.6: Notary idea.

browsers/CAs to another newly-created party, *notary*. The last project is also very similar but it uses TOR [47] anonymity network instead of notaries to gain a network perspective.

The MitM attacks are mostly specific user-targeted attacks, not global attacks, namely, only the victim's connections is manipulated by the attacker (Note that, making a global MitM attack is very difficult and requires having control over all the users' connections. Maybe Internet Service Providers (ISPs) are capable of doing such thing since their customers' connections pass over them).

So if we can ask some other users while SSL handshake: "Which certificate do you see for the site www.abc.com ?" then, since they would most probably not in the control of same attacker doing MitM to us, even if the attacker presents a rogue certificates of the SSL server, the asked users will respond the real certificate, which will give us a chance to understand the suspicious situation.

The users that we will ask that question are called *notaries* (see Figure 3.6) in these proposals. Notaries are just servers periodically checking the SSL servers and collecting the certificates that the SSL servers present. The trust decision of a SSL certificate will be made according to notaries' responses.

A question may come to mind at this point: If an active attacker can replace the SSL-server-presented-certificate to the client by doing a MitM attack, won't he be able to change the response of the notaries? Well the answer is no. He can not, because similar to CAs, the certificates of the notaries will be preloaded to client root store probably within the OS/browser source code.

3.4.1 Perspectives

Perspectives project is the idea of Dan Wendlandt, David G. Andersen and Adrian Perrig in Carnegie Mellon University (CMU), USA. They made the first proposal in their article, in 2008 [48].

In Perspectives, when the user gets the certificate from the SSL server while connecting to it, he does not verify it with current verification mechanism, instead he asks number of servers, called *notaries*, for that site's SSL certificate. It is suggested to ask 4-10 notaries before making a decision.

"This additional data from diverse network vantage points over a span of time gives clients the *perspective* to make a strictly better security decision" says the authors of Perspectives.

Notary servers store a record of all keys it saw from SSL servers along with a timestamp. When a request received, the notary servers look up the entry in the database and then give a response according to the key history for the requested key.

The notaries also check the consistency of a certificate by collecting certificates periodically, and giving a response to clients like; "The certificate is seen for this site for * days", which gives a valuable information (statistical and probabilistic) to the client in making a good trust decision.

Notary servers can be operated by any party; organizations, companies, even individuals.

Currently, they maintain 10 official notary servers, also there exists 6 third party servers. Perspectives is used by nearly 30,000 users worldwide and although current few notary servers suffice for now, there will be needed hundreds of them when Perspectives is used by all internet users, stated by its developers.

"Perspectives has challenges in completeness, privacy and responsiveness" thinks Moxie Marlinspike, the developer of similar project, Convergence. He says that it is just used for initial SSL connection (not for subsequent elements like scripts/images), so can not eliminate CAs completely. Also since the notaries we ask for a site certificate will be able to keep our browsing preferences, that will violate our privacy. At last he mentions the lag at SSL connection startup because you have to wait for some notaries to check the certificate and respond back to you [49].

There are also some issues about the notary infrastructure, specifications and security policies. Perspectives is described by its authors as "not bullet-proof, but provides a security trade-off suitable for many non-critical websites." Current Web PKI system is still recommended for more security requiring cases [50].

The Perspectives Project is implemented as a Firefox add-on (add-on version 4.5.2 currently). It shows a green tick icon on Firefox, if notaries, configured in the add-on, thinks the server-presented certificate is the same as notaries' records.

Pros:

- Easy to implement in browsers; just an add-on,
- No need for CAs anymore,
- Allow self-signed certificates to be used by any SSL server,
- Gives the users trust agility: Anytime you do not trust a notary, you can just remove it from your list, also can add new notaries, without any complication in connection to any SSL server.

Cons:

- Not works in closed networks (or internal notaries in the network should be operated, which is a resource-needed task),
- Having problems in *captive portals*[5], where connecting to the notary at SSL startup is blocked by the service-provider.
- *Citibank problem*[6].

3.4.2 Convergence

Convergence project [51], is very similar to Perspectives and released in August 2011 at Black Hat security conference [49], by Moxie Marlinspike. He also wrote an article in his blog [52], about that project.

Moxie enters the topic with a basic question: "Whom we have to trust? How long?". He then states that *trust agility* concept to be missing in the current Web PKI system, hence he designed Convergence especially providing that property.

Trust agility allows the users to decide whom to trust, and revise that decision anytime they want. But in the current Web PKI system it is not possible to do it in an easy way because when you decide to remove trust from a CA, for all the websites using the certificates issued by that CA, SSL clients will show warnings.

Similar to Perspectives, notary servers are set up to retrieve SSL-websites on the internet and record the certificates they present, then when the users ask for a site, respond according to their records [53].

It is suggested to ask many notaries and then decide either according to consensus or majority vote. Moxie refers this as *collective trust*. Also users can choose to remove the notaries that usually respond different than the consensus from your notary list, which will give the users trust agility in their decisions.

The difference between Perspectives is the user's active participation in selecting and removing the notaries. Users also remove trust from a notary anytime they want unlike in Perspectives.

The user-configurability and flexibility is Convergence's strongest feature, but also can be seen as a weakness, because users most probably would not change default settings, and the default added notaries in SSL clients will be receiving so many requests [50].

A security analyst in Qualys, Ivan Ristic, thinks Convergence to be a completely different proposal and should be tried [54].

[5] A captive portal is usually a login page used before connecting to internet. It is preferred for additional user authentication and payment purposes, and mostly in Wi-Fi hotspots, hotel rooms, business centers etc.

[6] Few websites like *www.citibank.com* prefer to present different certificates to SSL clients at SSL handshake, which makes the notary system fail since the notaries see the change of certificate and report as a suspicious situation to the user. Since there are very few websites like that, they will be expected to change that behaviour when proposals based on notaries used widely.

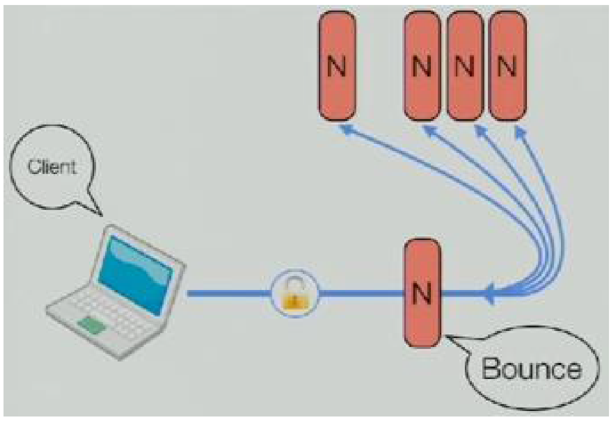


Figure 3.7: Notary bounce in Convergence.

Convergence promises to overcome the challenges of Perspectives that Moxie stated, via following:

- There is no network lag in Convergence, because, the client, after getting the presented certificate from SSL server at connection-startup, sends it to the notary, who will check with its data and then respond immediately.
- It uses *notary bounce* to provide privacy. Notary bouncing is an additional job of notaries in Convergence. The client opens a secure SSL connection to target notary by passing over a notary bounce, doing a job similar to a proxy. Since it is a SSL connection, the notary bounce will not see the queried website you are trying to connect. Also since you do not go directly to target notary, the target notary will see that the request comes from a notary bounce. This is depicted in Figure 3.7[7].
- Additionally, when the response comes from a notary for a site certificate, the client keeps the response in client cache, then when he needs to go to the same site at that day, he uses the same response, he does not ask the notary again.

Convergence can be extended to use any methods that the notary operator would like: DANE, Certificate Transparency or even CA signatures as done in current environment. This option variety for notaries in verifying certificates is mentioned as *extensible* feature of the proposal [55].

For the system to fail, all the notaries must misbehave or be under attack and that is a very low probability.

Convergence is implemented as a Firefox add-on, and can be downloaded at project website. The add-on will be showing the notaries' responses on Firefox GUI like 'Verified by Convergence' instead of in current system situations like 'Verified by Turktrust'. Its current development is done in Convergence Extra task [56].

[7] Source: `http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/hw-CertReputation.pdf`

It is a popular approach to SSL authentication, but big actors like Google has no plans to add it to Chrome. Although Google appreciates the idea as being great, they believe 99.99% of Chrome users never change the default Convergence settings which will cause the default-added notaries have a huge network traffic. They also see the issues in closed networks and captive portals as serious problems of the proposal [57]. Note also that, starting from 2013, Moxie started focusing another proposal: TACK.

There are more than 50 Convergence notaries, including Qualys company and EFF. The list of notaries can be found here [58].

Pros:

- No single point of failure (Users can trust several notaries and several notaries can give information for a specific site. So if there is not a consensus between notaries, users can reject the connection or decide according to majority (These options are configured in the browser-addon),
- Easy to implement in browsers; just an add-on,
- No extra work on SSL servers,
- No need for CAs anymore,
- Allow self-signed certificates to be used by any SSL server (In fact, there is no difference between a self-signed certificate and a CA-signed certificate in view of Convergence, they are just certificates, since CAs are taken out of the authentication task),
- Gives the users trust agility: Any time you do not trust a notary, you can just remove it from your list, also can add new notaries, without any complication in connection to any SSL server.

Cons:

- Not works in closed networks (or internal notaries should be operated in that networks, which is a resource-needed task),
- Having problems in captive portals, where connecting to the notary at SSL startup is blocked by the service-provider.
- Citibank problem.

3.4.3 DetecTor

DetecTor [59] project uses the similar notary idea but instead of using different dedicated notary servers, it makes the users be their own notary. It is proposed by Kai Engert in a "The Chaos Computer Club" conference lightning talks, Hamburg, Germany in 2013 [60].

Its author thinks the similar notary-based proposals such as Perspectives and Convergence bring another trusted 3rd party to us: notaries. And he does not like this because it is very similar to CA trust! Instead, he proposes not using a 3rd party, and prefers the clients be their own notary. He also wanted to design a tool to work even on small systems with limited resources and run without user confirmation or interaction.

As guessed from its project name, it uses anonymity-providing and distributed TOR network to have different network perspectives for its aim.

In DetecTor, when user want to connect a SSL server, the SSL client uses the proxy feature (the routing component) of the TOR network (no need of anonymity feature), and firstly start 5 simultaneous connections to target SSL server, just waiting for the certificate the server will present. When it gets that information, it immediately closes the connections. Then after making a trust decision according to the 5 responses, the SSL client will continue or reject to open a SSL connection to target SSL server.

It configures the 5 connections to use TOR exit nodes (third node) to be in different countries (the TOR client software permits only exit nodes to be in a given set of countries), so that it offers a better network perspective and the possibility of being manipulated by an adversary is reduced (In detail, it divides the world into 5 parts called *spheres*, where each country belongs to one sphere, then make 5 connections to different spheres).

DetecTor software library is capable of storing previous responses for sites, and in case of unavailability in TOR network, tries to use that responses.

The aim of the developer is to create a general-purpose software library to easily be implemented in any SSL clients and be used for any application layer protocol along the HTTP. But it is stated to be in ‘early development’ in its website.

Pros:

- No extra notary servers needed,
- No trusted 3rd party (although you rely on the nodes of TOR network),
- Doesn’t require new server software to be developed or deployed.

Cons:

- Requires change in SSL protocol,
- Performance problems: 5 connections to the TOR network, waiting the responses. Thinking the current latency in TOR network, it is a big issue.
- Doesn’t work for closed networks, since the SSL servers in that networks cannot be reached from TOR exit nodes,
- In early stages.

3.5 Proposals on Collecting the Certificates

There also exists some proposals, aiming to:

- Collect the certificates in use,
- Try to detect the forged, suspicious certificates,
- Provide valuable statistical information about issuer CAs and certificate's cryptographic properties such as used hashing algorithm and key size.

They may not be addressing all part of the problem but they aid some part of it that the other proposals did not.

3.5.1 Crossbear

Crossbear project aims to detect MitM attacks on SSL protocol by help of notary idea, and also try to find the location of the attacker by tracing the routes. It is proposed by Ralph Holz, Thomas Riedmaier, Nils Kammenhuber, George Carle in their article [7].

They start with a curiosity of MitM attacks by asking questions like; "How many attacks so far, what characteristic they have and which certificates are used?". Since the attacks mostly are not disclosed by the victim because it may be seen a sign of security weakness of the organization or the company, we are not so aware of details for MitM attacks.

They believe that such detail of a MitM attack/attacker is needed, in order to take the correct precautions for them. They also note that MitM attacks are not always done with a forged certificate, mostly a self-signed certificate is enough.

They developed a tool, called *Crossbear* for that purpose. It aims to prove a MitM attacks' existence then try to find the attacker's location later give a report for the attack.

Crossbear does 3 tasks for a SSL MitM attack: Detection, localisation and reporting.

For detection; looking from different network perspectives, namely, notary idea is used. The Crossbear server queries Convergence notaries and stores SSL server certificate information sent by them. If it notices a MitM attack by that data, it adds that information to its attacked-servers list.

For localisation; that is, finding the location of the attacker, it uses a large number of traceroutes, that are done by *hunters* from different locations. The hunters look the certificate chain of presented certificate by SSL server, record their IP route by tracerouting to that SSL server. The collected information then sent to central database for analysis. For choosing the target servers to analyse, hunters use the regularly-updated list in Crossbear server about reported MitM-attacked-servers.

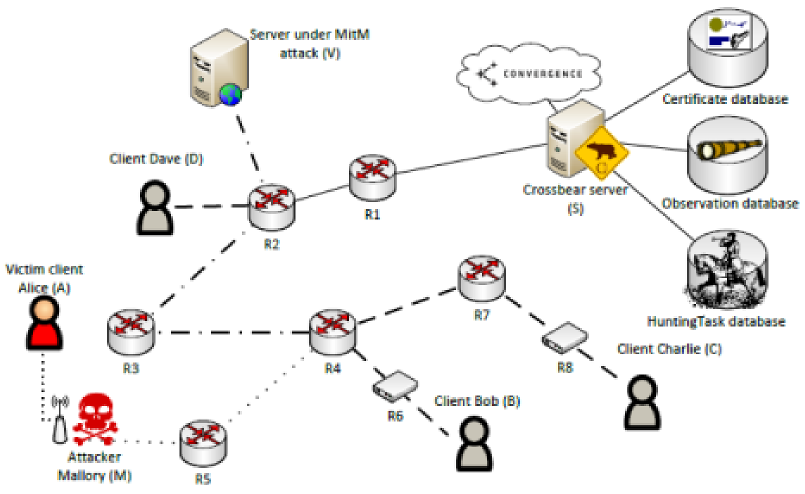


Figure 3.8: Crossbear: Attacker detection.

The attacker's location is determined by cross-bearing, namely, comparing the routes which hunters send the information about, and with the help of looking at the intersection of them, as seen in Figure 3.8[8].

Two types of hunters exist: Firefox-addon used for detection and localisation, and standalone software for localisation. For reporting attacks to the clients, the add-on is used.

Crossbear also stores information from different sources along with hunters, like:

- *Geo-IP-mapping*: To find out the countries the attack passed over, by using geo-IP databases.
- *Used CAs*: To see the CA preference of domains (Since famous domains like Google, tend to use same CAs, the change in the CA can be used to determine the suspicious situation).

Crossbear has finished its beta phase; up to now, they did not notice a MitM attack, but continue to collect information from hunters and other sources. The project members suggest using Crossbear to detect and localize the MitM attacks but express that although it is successful for some attack types, it is not a full-solution to the problem of detection/localisation of all attacks. After the studies they did so far, they state that best results can be expected against an attacker doing a non-selective MitM attack, or attacks performed centerizedly for example by ISP's, monitoring all traffic to a SSL server.

Pros:

[8] Source: `http://www.net.in.tum.de/fileadmin/bibtex/publications/papers/holz_x509forensics_esorics2012.pdf`

- Very different idea: To find to location of the attacker, which other proposals do not aim.
- No extra work SSL servers.

Cons:

- In early stages,
- Need more hunters to determine the location of the attacker precisely,
- Not aims to address all parts of the problems, although it aids some part of it.

3.5.2 SSL Observatory

SSL Observatory [61] is a project of Electronic Frontier Foundation (EFF) to collect the X.509 SSL certificates of SSL enabled-websites by simply downloading like anyone could do, and store them in a MySQL database to make further analysis. Its purpose is to search the vulnerabilities, see CAs' role better and provide data to researchers in making analysis of current Web PKI.

Its authors see the large number of CA/Intermediate CAs as the cause of the problem, and since all clients trust them by default, even one wrongdoing intermediate CA can make the Web PKI trust system insecure. Then they decide to focus on inspecting especially the CAs' behaviours in current Web PKI. They expressed their purpose as "Studying CA behaviour and detecting problems" in DEFCON-18, July, 2010.

They criticize the X.509 certificates as being *extremely flexible* and *ugly*, since it is designed in 1980s, even before HTTP protocol and define it as the other cause of the problem.

They also think EV certificates to be a great idea for making Web PKI system more secure and suggest using EV certificates as much as possible. But remembering the numbers about EV certificates in use (approximately 34.000 EV certificates used with issuers around 40 [15]), we need more way to go for that.

SSL Observatory scans all IPv4 addresses listening on the well-known SSL port,443, with a tool NMAP [62] and it started scanning task with running on 3 2GB-RAM linux machines having 100Mbps internet. With the collected data, it managed to derive valuable information on current Web PKI such as:

- N.of SSL-secured IP's,
- N.of valid/invalid certificates,
- N.of trusted CAs defined in certificates,
- N.of organizations defined in certificates,

- N.of servers using broken keys,
- N.of certificates using ... hash algorithms like MD5, SHA1 and SHA256,
- Suspicious certificates such as the ones using 512-bit RSA keys,
- Interesting behaviour of SSL clients and servers.

Currently, SSL Observatory is a Firefox add-on, collecting certificates while the clients are connecting to SSL websites, then sending them to EFF to be saved to a dedicated database.

As noticed, there is obviously a privacy problem that EFF can store also the sending user's IP and location information. For this issue, Eckersley, a project member of SSL Observatory says they are just aiming to know the certificates in use and focusing on the ones used for malicious purposes, they do not care the user sending the certificate information and he suggests using anonymity services like TOR to overcome that privacy issue [63].

The project also aims to warn users about suspicious CA signatures that the add-on decides according the information like stored SSL certificates database. It can last some seconds to give that information but it is still an improvement to system.

Pros:

- Allows to see the certificates in use and detect suspicious cases which allows us to monitor the current Web PKI system in use, better.
- Easy to use, just an add-on,
- No extra work on SSL servers or clients.

Cons:

- Privacy problem as stated above,
- Not aims to address all parts of the problems, although it aids some part of it.

3.5.3 Berkeley ICSI Notary Project

Similar to SSL Observatory, ICSI Project [64] aims to collect SSL certificates of websites (but passively), and stores them in a database to make analysis mostly on the validity of certificates, common/uncommon patterns and suspicious situations. There was also a project of Google named 'Certificate Catalogue' doing similar job, but it was halted.

It provides a DNS interface for the collected data to be queried by clients with just giving the SHA1 hash of the queried certificate

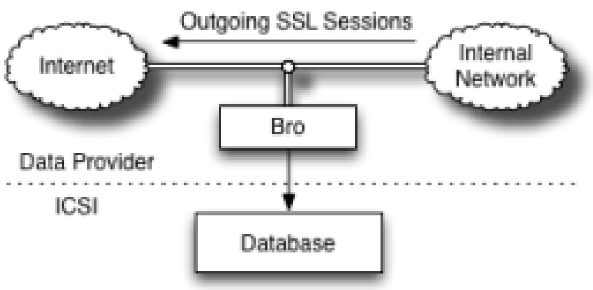


Figure 3.9: ICSI: Collecting certificates from live network.

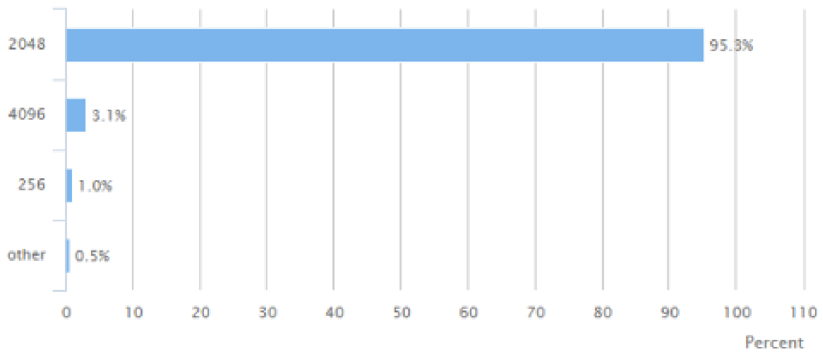


Figure 3.10: Certificate key lengths in use.

They emphasizes the problem that a compromised CA failing the trust in whole system globally, so that something must be done.

ICSI Project operates a notary service [65] like in projects Perspectives and Convergence, allowing a different network perspective for detecting the malicious certificates. The notary, additionally, collects the certificates from live network traffic and stores them in a central database which is a very different collecting operation than in similar projects like SSL Observatory. With that, they provide a nearly real time perspective for certificates in use.

The collecting task is done via adding their monitoring infrastructure to the border gateways (see Figure 3.9) of currently 10 helping organizations. When a certificate is seen in the network traffic of that gateways, it is uploaded to central ICSI database.

They use open-source 'Bro network' [66] monitor to extract certificates in the traffic. With that monitoring they also collected valuable statistical information such as N.of certificates in notary, N.of connections observed (See an example in Figure 3.10).

Pros:

- Allows to see the certificates in use and detect suspicious cases which allows us to monitor the current Web PKI system in use, better.
- No change in current system.

- No extra work on SSL servers or clients,

Cons:

- Requires some organizations to help them by allowing to add some monitoring tools to their gateways,
- Not aims to address all parts of the problems, although it aids some part of it.

3.6 Summary

Web PKI seemed to solve MitM attack problems for SSL, but it inherently had some problems: CAs. In the early stages, there were very few secure SSL sites so just one CA was enough. But since the number of certificates became millions, the situation had changed. Since it is a profitable task, many CA companies emerged. The Web PKI design was also permitting CAs to create subordinate-CAs which has the same signing authorization and are trusted by all SSL clients. The number of CAs/sub-CAs, and their ability to issue a certificate to any domain without domain owner's knowledge, caused many security-breach incidents in recent years, such as in Diginotar and Turktrust examples. These incidents showed that something must be done to overcome the authentication weakness in SSL.

Starting from 2008, and mostly after 2011 when those incidents occurred, some proposals by researchers from universities, big actors like Google and standardization organizations like EFF, put forwarded to improve SSL server authentication. The proposals were mainly about: Introducing new trusted parties, trying to look from the other users' perspectives, utilizing DNS records, collecting and analysing all certificates used by servers and offering a transparent system where o domain owner could see all certificates issued for his domain name. Since choosing the ideal solution for the problem involves knowing all aspects, pros and cons of the proposals made so far, in this chapter, such details of the proposals were given. To summarize, key features of the proposals are listed below:

- **PKPE:** Stores SSL server public key information (pin) in clients, delivers pin to clients via new HTTP header, allows clients to report pin validation errors automatically to a defined server.
- **TACK:** Stores SSL server public key information (pin) in clients, delivers pin using TACK extension (a way to be added to SSL protocol), instead of pinning the server public key, pins the server master public key (called TSK) used to sign the server public key, a way to provide key change flexibility.
- **Perspectives:** Introduces trusted 3rd parties, namely, notary servers, that are periodically collecting and storing certificates of SSL servers and when requested by clients, giving information about public key change history for requested SSL server. Users ask more than one notary and then decide according to majority or consensus votes of responses.

- **Convergence:** It is similar to Perspectives. It has improvements onto it. It provides trust agility for user, and also has solutions to Perspective's privacy and performance issues.

- **DetecTor:** Makes users their own notary and opens 5 simultaneous TOR network connections, each of whose TOR exit node computer is from different part of the world, afterwards, compares the certificates presented by SSL server and decide accordingly.

- **Certificate Transparency:** Designs an open, transparent system of SSL certificates. Stores all certificates into a public, auditable, append-only log servers, then clients will reject any certificate not added to the log. Also domain owners will be able to see the certificates issued for their domain and take action for suspicious cases.

- **DANE:** Stores certificate information of SSL servers into the DNS server records, along with IP information. To provide security of DNS queries, requires DNSSEC to be in use. Clients will obtain certificate information while querying IP of a domain name from DNS server.

- **SSL Observatory:** Collects SSL certificates of servers into a central database to make analysis. Allows to see the current Web PKI, certificates, and CAs' information statistically, furthermore, detects suspicious cases. Helps clients via a browser add-on which gives warning for suspicious certificates.

- **Crossbear:** Proves MitM attacks' existence and tries to find the location of the attacker. Uses Convergence notaries to detect the attack, also uses *hunter* browser-addons, that will send their traceroute for attacked server, to Crossbear central server for localisation of the attacker.

- **ICSI Notary Project:** A notary service, collecting certificates from SSL servers. Additionally collects from live network, via looking inside the packets. For that, they added their infrastructure to gateways of some cooperators. Eventually it gives information for a queried certificate by DNS query.

CHAPTER 4

COMPARISON OF THE PROPOSALS

The proposals for improving SSL server authentication were mostly made starting from 2011 when most of the recent attacks took place. Although they took good reviews when proposed, we still do not have a solution implemented widely and covering all parts of the problem.

Because, as seen in Chapter 3, each proposal has pros and cons. Each one has a solution to some parts of the problem. While improving the current system, most of the proposals require some additional tasks to be performed by: CAs, domain operators, clients etc.

Determining the ideal solution requires knowledge of each of the current proposals' details, pros and cons and the researches about these proposals' analysis and comparison. While our study, we read several documents about such analysis and comparison, listed in the following items:

- Clark and Oorschot made an analysis about pinning proposals [67].
- Weeks made a comparison of 8 proposals according to 17 criteria [68].
- Gielesberger made an analysis about pinning, DANE, CT and the proposals using notary idea [69].
- Brown and Jenkins made an analysis and comparison of pinning, CT and DANE proposals after specifying some desired properties for the solution [43].
- Enrique, Cochrane, Moreira-Lemus Paez-Reyes, Marsa-Maestre and Alarcos made an analysis covering most of the proposals, along with giving their proposal called 'MIDAS', using a different type of pinning [26].
- Perl, Fahl, Brenner and Smith made a presentation including a comparison of 7 proposals according to nearly 20 criteria [70].
- Dacosta, Ahamad and Traynor made a study including the property defining, required to achieve an effective and practical defense against MitM attacks [71].
- Grant made an analysis of 8 proposals and compared them according to 11 criteria [50].

- Ristic made an analysis for the current proposals in his up-to-date book about SSL [9].
- CT project website provides a chart comparing 5 proposals according to 7 criteria [72].

We will continue with defining our desired properties to exist in the ideal solution, afterwards, we will make a detailed comparison according to these desired properties.

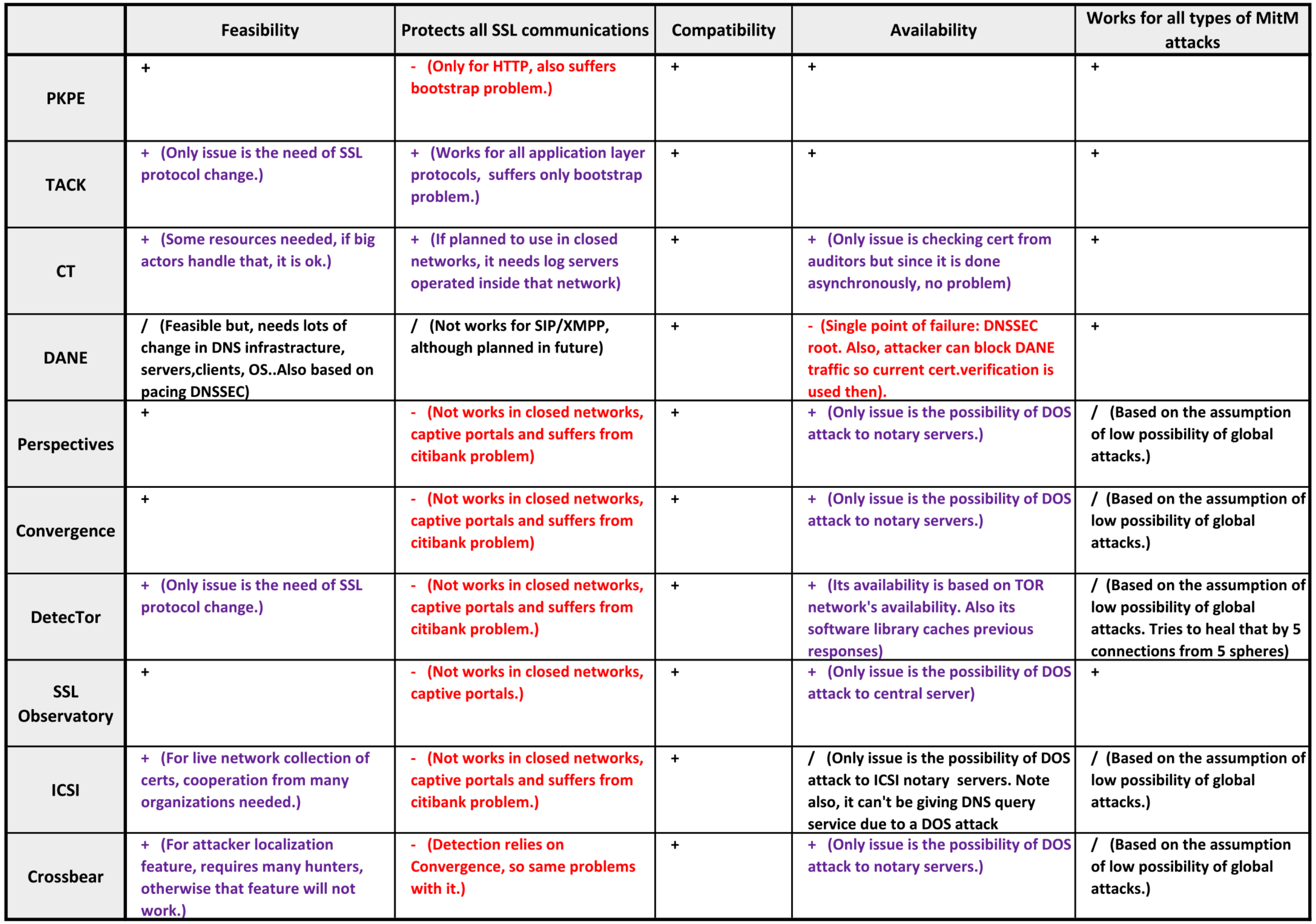
4.1 Desired Properties in the Ideal Solution

- **Feasibility:**It should be practical, not just theoretical . It should be implemented easily and in a short time, ie. in months.
- **Immediate benefit to a participant:** Once a participant, such as a user, CA or server administrator decides to use the solution, he should get the desired benefit immediately, without waiting all parties to participate.
- **No performance weakness:** It should not offer slower validation scheme than current system, certificate validation must finish in few seconds.
- **Not causing new problems:** It must not create new problems while solving the currents. We do not want any new problems.
- **Handling end-user trust decision correctly:** It must present the required information in easily-understandable form and let the user give the trust decision. The default trust choice must be the one offering most-secure way. Also the user may be able to change his trusted parties any time (trust agility), and this configuration change must be implemented immediately.
- **Impostor detection:** Domain owners must be able to see all the certificates issued for their domain.
- **User privacy:** The browsing preferences of the users must not be revealed to any party in the solution.
- **Protecting all SSL communications:** It must be usable for all application layer protocols, not just HTTP (Recall that SSL/TLS is transport layer protocol and used for securing any application layer protocol). Also it must be usable in all networks (not just internet). Lastly, it should be working for every SSL connection, regardless of the client or target server.
- **Resistance to attacker's client/server invasion:** It must not fail even client or server computer is under control of the attacker.
- **Resistance to rogue/compromised CA:** It must help us to detect the certificates issued either mistakenly or deliberately by CAs or by the one who stole the CA private key.

- **Work for all types of MitM attacks:** It must deal with both local attacks (only victim's or his neighbour networking devices' connections are under attack) and global attacks (the attacker is able to manipulate all network connections)
- **Little out-of-band check necessity:** It should not require out-of-band checks since attacker can also target these channels by MitM or DOS attacks.
- **Introducing no trusted 3rd party:** We already have trusted 3rd parties: CAs, and we have many of them. The solution should not introduce trusted 3rd parties.
- **Allowing self-signed certificates:** We do not want to pay for a certificate. Self-signed certificate creation is easy and free. The solution should allow them.
- **Little work in SSL clients:** It should not require much changes in client software/hardware.
- **Little work in SSL servers:** It should not require much changes in server software/hardware.
- **Easy key change/revoke:** It should have an easy mechanism to change and revoke all the public keys of the parties that it introduced.
- **Compatibility:** It should work well with the current certification validation scheme. The non-participants also may be able to use the current validation system even when the new solution is adopted to hardware/software.
- **Availability:** It should work 7/24 and should not have a single point-of failure.
- **Little change in current protocols:** It should not require changes in current protocols like SSL and TLS.

4.2 Comparison

Comparison of the proposals are made in the following figures in the next page. In each figure, ten proposals are compared according to 5 properties. If the comment in a cell starts with '+', it means that proposal mostly satisfies the property. If comment starts with '-', it means not satisfying, finally, if it starts with '/', it means 'in-between', that is, it does not fully satisfies the property.

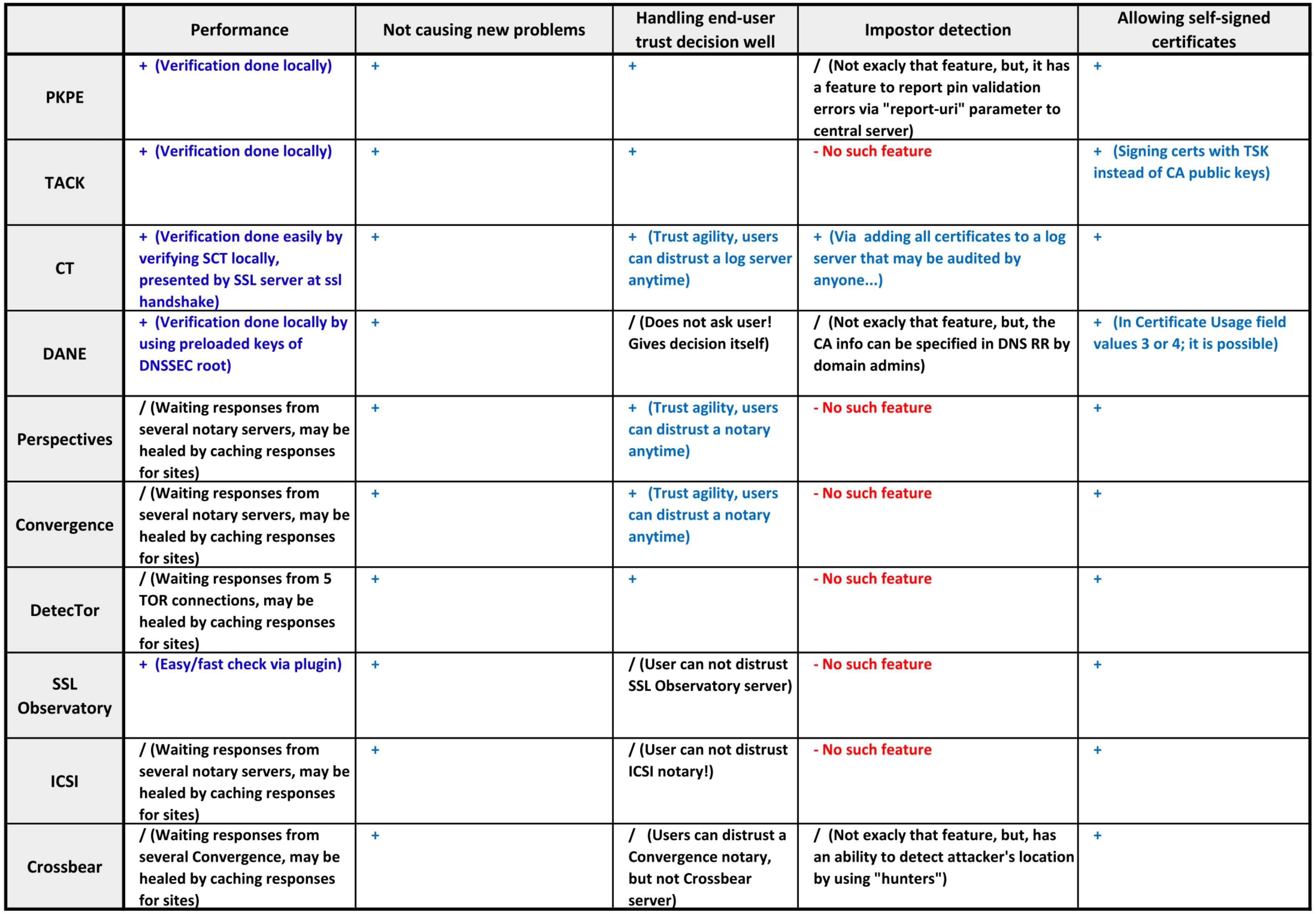


	Feasibility	Protects all SSL communications	Compatibility	Availability	Works for all types of MitM attacks
PKPE	+	- (Only for HTTP, also suffers bootstrap problem.)	+	+	+
TACK	+ (Only issue is the need of SSL protocol change.)	+ (Works for all application layer protocols, suffers only bootstrap problem.)	+	+	+
CT	+ (Some resources needed, if big actors handle that, it is ok.)	+ (If planned to use in closed networks, it needs log servers operated inside that network)	+	+ (Only issue is checking cert from auditors but since it is done asynchronously, no problem)	+
DANE	/ (Feasible but, needs lots of change in DNS infrastructure, servers,clients, OS..Also based on pacing DNSSEC)	/ (Not works for SIP/XMPP, although planned in future)	+	- (Single point of failure: DNSSEC root. Also, attacker can block DANE traffic so current cert.verification is used then).	+
Perspectives	+	- (Not works in closed networks, captive portals and suffers from citibank problem)	+	+ (Only issue is the possibility of DOS attack to notary servers.)	/ (Based on the assumption of low possibility of global attacks.)
Convergence	+	- (Not works in closed networks, captive portals and suffers from citibank problem)	+	+ (Only issue is the possibility of DOS attack to notary servers.)	/ (Based on the assumption of low possibility of global attacks.)
DetecTor	+ (Only issue is the need of SSL protocol change.)	- (Not works in closed networks, captive portals and suffers from citibank problem.)	+	+ (Its availability is based on TOR network's availability. Also its software library caches previous responses)	/ (Based on the assumption of low possibility of global attacks. Tries to heal that by 5 connections from 5 spheres)
SSL Observatory	+	- (Not works in closed networks, captive portals.)	+	+ (Only issue is the possibility of DOS attack to central server)	+
ICSI	+ (For live network collection of certs, cooperation from many organizations needed.)	- (Not works in closed networks, captive portals and suffers from citibank problem.)	+	/ (Only issue is the possibility of DOS attack to ICSI notary servers. Note also, it can't be giving DNS query service due to a DOS attack	/ (Based on the assumption of low possibility of global attacks.)
Crossbear	+ (For attacker localization feature, requires many hunters, otherwise that feature will not work.)	- (Detection relies on Convergence, so same problems with it.)	+	+ (Only issue is the possibility of DOS attack to notary servers.)	/ (Based on the assumption of low possibility of global attacks.)

Figure 4.1: Comparison Table 1.

	Easy key change/revoke	Immediate benefit to participant	No trusted 3rd party	User privacy	Resistance to attacker's invasion of server/client
PKPE	+	+	+	+	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
TACK	/ (With usage of TSK, it gains flexibility in key change. But for TSK change, it needs similar approach as CA keys)	+ (Only big issue is to wait SSL protocol change)	+	+	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
CT	- (No clear policy how to deal with log server public key change.)	+ (Easily start with an add-on)	/ (It has: CT log servers, but we can distrust easily them	- (Auditors see clients' browsing adresses	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
DANE	- (No clear policy how to deal with log server public key change.)	/ (After many changes and DNSSEC's widely usage, then it will be beneficial to a participant)	- (DNSSEC root! TLD DNS servers...)	/ (DNS servers be aware of running SSL servers and used ports for them)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
Perspectives	- (No clear/easy policy how to deal with notary public key changes.)	+ (Easily start with an add-on)	/ (It has: notaries, but we can distrust them easily.	- (Notaries see clients' browsing adresses)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
Convergence	- (No clear/easy policy how to deal with notary public key changes.)	+ (Easily start with an add-on)	/ (It has: notaries, but we can distrust them easily.	+ (Notaries see clients' browsing adresses but it overcomes that issue by utilizing notary bounce concept, similar to proxy idea)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
DetecTor	/ (TOR client software handles that key changes)	/ (Only big issue is to wait SSL protocol change)	+ (TOR network exit nodes, but they change nearly in all connections)	+ (TOR network's anonimity feature handles privacy problem)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
SSL Observatory	- (No clear/easy policy how to deal with SSL observatory central server public key changes.)	+ (Easily start with an add-on)	- (It has: SSL Observatory servers)	- (SSL Observatory servers see clients' browsing adresses)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
ICSI	- (No clear/easy policy how to deal with ICSI notary public key changes.)	+ (Easily start with an add-on)	- (It has: ICSI notary)	- (ICSI notary sees clients' browsing adresses)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)
Crossbear	- (Similar issues with Convergence.)	+ (Easily start with an add-on)	/ (It has: Convergence notaries, but we can easily distrust them.	+ (Similar issues with Convergence)	- (Attacker can modify client's root CA-store also manipulate browser and plugins)

Figure 4.2: Comparison Table 2.



	Performance	Not causing new problems	Handling end-user trust decision well	Impostor detection	Allowing self-signed certificates
PKPE	+ (Verification done locally)	+	+	/ (Not exacly that feature, but, it has a feature to report pin validation errors via "report-uri" parameter to central server)	+
TACK	+ (Verification done locally)	+	+	- No such feature	+ (Signing certs with TSK instead of CA public keys)
CT	+ (Verification done easily by verifying SCT locally, presented by SSL server at ssl handshake)	+	+ (Trust agility, users can distrust a log server anytime)	+ (Via adding all certificates to a log server that may be audited by anyone...)	+
DANE	+ (Verification done locally by using preloaded keys of DNSSEC root)	+	/ (Does not ask user! Gives decision itself)	/ (Not exacly that feature, but, the CA info can be specified in DNS RR by domain admins)	+ (In Certificate Usage field values 3 or 4; it is possible)
Perspectives	/ (Waiting responses from several notary servers, may be healed by caching responses for sites)	+	+ (Trust agility, users can distrust a notary anytime)	- No such feature	+
Convergence	/ (Waiting responses from several notary servers, may be healed by caching responses for sites)	+	+ (Trust agility, users can distrust a notary anytime)	- No such feature	+
DetecTor	/ (Waiting responses from 5 TOR connections, may be healed by caching responses for sites)	+	+	- No such feature	+
SSL Observatory	+ (Easy/fast check via plugin)	+	/ (User can not distrust SSL Observatory server)	- No such feature	+
ICSI	/ (Waiting responses from several notary servers, may be healed by caching responses for sites)	+	/ (User can not distrust ICSI notary!)	- No such feature	+
Crossbear	/ (Waiting responses from several Convergence, may be healed by caching responses for sites)	+	/ (Users can distrust a Convergence notary, but not Crossbear server)	/ (Not exacly that feature, but, has an ability to detect attacker's location by using "hunters")	+

Figure 4.3: Comparison Table 3.

	Resistance to rogue/bad CA	Little out-of-band check need	Little work in SSL clients	Little work in SSL servers	Little change in current protocols
PKPE	- (No such ability)	+	/ (Needs work in clients, to deal with pins)	/ (SSL servers must send pins via new HTTP header.)	+
TACK	- (No such ability)	+	/ (Needs work in clients, to deal with pins)	/ (SSL servers must respond TACK extension)	- (Needs change in SSL protocol, allowing a TACK extension)
CT	/ (Detects malicious CA and certificates, in fact it allows its detection to be made by auditors)	/ (Only issue is verifying asynchronously from auditors)	/ (Needs work in clients, to deal with SCT)	/ (SSL servers must present SCT to the clients while SSL handshake.)	/ (If SCT presented while handshake by server, no problem. But TLS extension method needs SSL change)
DANE	- (No such ability)	+	/ (Needs work in clients to verify certs with DNSSEC keys, also OS's must support DNSSEC)	- (Needs change in both SSL servers and DNS servers for DNSSEC adaptation)	/ (No change needed, but DNSSEC deployment is required!)
Perspectives	- (No such ability)	/ (Notary server check, but can be overcome with caching responses for sites)	/ (Needs work in clients, to deal with notary communications and caching	+	+
Convergence	- (No such ability)	/ (Notary server check, but can be overcome with caching responses for sites)	/ (Needs work in clients, to deal with notary communications and caching)	+	+
DetecTor	- (No such ability)	/ (5 TOR Network connections needed, but can be overcome with caching responses for sites)	/ (Needs work in clients, to deal with TOR communications and caching)	+	- (Needs change in SSL protocol, for opening 5 tor connections and waiting responses)
SSL Observatory	- (No such ability)	/ (SSL Obsrv.server check, but can be overcome with caching responses for sites)	/ (Needs work in clients, to deal with SSL Observ.server communications and caching)	+	+
ICSI	- (No such ability)	/ (ICSI Notary server check, but can be overcome with caching responses for sites)	/ (Needs work in clients, to deal with ICSI notary communications and caching)	+	+
Crossbear	- (No such ability)	/ (Convergence check, but can be overcome with caching responses for sites)	/ (Needs work in clients, to deal with Convergence communications and caching)	+	+

Figure 4.4: Comparison Table 4.

4.3 Summary

Some comments about the comparison are follows:

- There is not a proposal covering all our desired properties.
- Some of the proposals are really successful in terms of security, privacy and performance.
- While fulfilling the properties about security or privacy, some other properties such as ‘no trusted 3rd party’ or ‘Little out-of-band checks’ become unsatisfied in the proposals. Furthermore, to achieve flexibility in key change, SSL protocol change becomes a necessity.
- The proposals face multiple challenges because of their complexity, deployment and operational costs. They also require some additional tasks to be performed by CAs, domain administrators, clients, while improving the current system.
- The proposals about pinning are easily-deployed but faces *poison pin* problem. They also have a problem in pin change. Anyway, they seem as the first solution for us to start the improvement to the current system.
- The proposals about notary idea, are really wisely-developed. Since MitM attacks are mostly specific user-targeted attacks, looking from another users perspective, may reveal the MitM to the victim. However, they face performance and privacy problems, although they try to solve that issues by caching responses at client and offering proxy-like solutions. They are highly feasible, only obstacle for them is the required resources such as notary servers. But there exists some notary servers currently, and some plugins are developed for browsers, so that they can be evaluated.
- The proposals about collecting certificates of SSL servers, are based on a very simple idea but while helping us to see fraudulent certificates for SSL-servers, they additionally provide very valuable information about current CA system: Issuer CAs/subordinate CAs, used certificates and cryptographic primitives implemented in that certificates etc.
- CT proposal provides us a transparent Web PKI system where a domain owner can see the certificates used for his domain name. This is very vital because most of the MitM attacks for SSL are possible due to lack of such feature. Although CT requires some additional resources, we believe that it can be managed by big actors in Web PKI: CAs, browser vendors etc. Furthermore, its privacy problem is not big and can be overcome via anonymity-providing services, or using proxies.
- DANE proposal, although seems very promising, is the last solution to be selected according to us. Because, its prerequisite, DNSSEC, is under spread phase for years, but it still needs lots of tasks to be completed. But, even those tasks finishes, we will face bigger task: Client changes, such as OS, client software

and server software changes. Furthermore, we think DNS operators are not better to guard against cyber attacks than CAs. But recall that, DANE requires DNS operators to put TLSA RR value into DNS records, and to protect its integrity. Finally, since it has a single-point of failure, namely, DNSSEC root, which can be considered as the root CA of all CAs, it did not seem logical to us to rely on just one trusted 3rd party that we will never be able to distrust.

- Nearly all the proposals have a problem in public key change of the entity that provides root trust. For example, in CT, public keys of log servers, in notary-based proposals, public keys of notary servers. It seems, there is only one way to deal with that issue, similar to the way, done in current environment: Preloading such root public keys to clients (Bundled with OS or SSL client software code).

CHAPTER 5

CONCLUSION

In this thesis, we have studied SSL security, mostly with a focus on its authentication piece. Firstly, we presented a clear picture of the *SSL server authentication problem*. Secondly, we explained ten famous proposals put forward to solve the problem, by giving information about their design, key features, pros and cons. Thirdly, we specified the desired properties to exist in the ideal solution. Lastly, we made a detailed comparison of ten proposals according to that desired properties.

To conclude our work, we will now give our final remarks and suggestions with some recommendations for future work on the subject.

5.1 Final Remarks

We should deploy a solution for *SSL server authentication problem*, and we are already late for that. Some MitM attacks for SSL may be happening while these lines are written. Although the parties of the Web PKI are well-aware[1] of the problem, and are trying to find a solution, we are still suffering from the problem. The current progress of the proposals are not enough, hence, precautionary measures should be undertaken immediately.

According to us, we currently have the following course of actions:

- **Continue with the current system applying small fixes:** We may choose not to make big changes and use the existing system while applying some fixes. Because the current CA system is not so bad, it has been used for years and proved its maturity. It has just some issues to be fixed, that is it. Therefore it is a logical conclusion to continue with the current system.

  Firstly, we should focus on improving the CAs, determining more standards for them in terms of protecting their private keys and their organizational trustiness proof. With that, we can minimize the certificates issued mistakenly or maliciously by CAs. We should also find a solution to subordinate-CAs. We may

[1] The number of proposals are a good proof of the awareness of the public.

choose to eliminate them, or force them to be as reliable as root-CAs. Anyway, the number of CAs are definitely large, and must be reduced to the minimum.

Secondly, we should find a solution after making a risk analysis. In that analysis we should specify the cases where we really need secure communications. For example, although many SSL-servers exist, how many of them are really dealing with sensitive data and are possible target for attackers? A game site? A personal blog? Or an online banking site? After specifying the real needs for SSL-security, we then can apply some fixes for those cases. As a start, we can think of pinning all the certificates of online banking, highly-popular mail services, e-commerce servers, social networking (e.g. facebook, twitter) sites etc. If these websites prefer certificates whose cryptographic parameters are highly-secure (e.g. at least RSA 8192 bits), they can specify a long expiration date (2-3 years), so that, pin change may not be troublesome. Since device root stores, keeping certificates of CA public keys, are currently managed with OS security updates, these pins' management can also be added to that task. Note that, since this seems the easiest way improving the security, Google and Mozilla started using these way, and preloaded some server pins mostly for servers owned by them, into their browsers.

Thirdly, using EV certificates for high security-required sites can improve the system, since these types of certificates are issued after strict procedures, such as document declaring and face-to-face meetings between CAs and domain owners.

Finally, for the top-security requiring situations, client certificates can be another option, but it is not easily-scalable solution and puts all the management task onto server owners, such as certificate creation for each client and sending securely to the them etc. They are mostly preferred by websites giving paid-services to their customers, but can be noted as another solution.

- **Use more than one proposal to cover all our needs:** Most of the proposals, as noticed, focuses and has a solution to some part of the problem. And they are mostly orthogonal, that is, they can be used together in the solution. For example, DANE, Pinning, CT, Convergence can be used at the same time. Well, it may not be reasonable to build many systems since it will cause management problems, but we can choose the first proposal satisfying our needs at most, then use another proposal covering the insufficient parts of that proposal. So, we could reach our goal by using just 2 proposals at the same time. These proposals may be selected as TACK and CT and additionally Convergence.

- **Use the number one proposal and try to improve it:** Choosing the proposal that fulfills our most of fundamental needs, namely, TACK, and start deploying it globally to guard ourselves to the problem immediately. And to make TACK to cover all our needs, we can try to add the features it lacks, looking to the strong features that other proposals (such as CT and Convergence) offer.

- **Use the proposal focusing directly on the root trust:** As noticed while reviewing the proposals, the main trust is finally based on the keys preloaded to client side. For example;

  - **CT:** Log servers' public keys,

  - **Perspectives/Convergence/Berkeley ICSI Notary:** Notaries' public keys,
  - **DetecTor:** TOR network nodes' public keys,
  - **DANE:** DNSSEC root/TLD's public keys,
  - **Crossbear:** Crossbear server's public key,
  - **TACK:** Tack Signing key (TSK),
  - **PKPE:** Pin of HTTP servers.

  So, why don't we use a solution directly providing the main trust? Well, there is a proposal for this: Pinning. Since it does not require any additional infrastructure like servers, it is easy to implement it.

  However, pinning scheme has a problem in key change. To overcome that, we can prefer TACK since it uses TSK as a pin which provides flexibility in key change. TACK seems a good and overall solution to the problem, since it covers many items in our desired properties and can be used for any application layer protocol in addition to HTTP. But it suffers *poison pin* problem, that is, while pinning TSK, there may be an MitM attack (though it is a very low probability) therefore, a wrong key may be pinned. If we can overcome this, we nearly reached a viable solution.

  At that point, we need a helper approach from other proposals. We must be sure that while pinning TSK, it must be the real key of the SSL server. If client does not have a TSK pinned for that server before, it will ask for help from other tools in the proposals, and use them to be sure whether TSK is the valid key at the time. For that, we can choose among Convergence, DetecTor and CT. We don't care much the performance issue, we can even wait 5-10 seconds if needed. SSL client software can give a detailed message about the operation, and ask us to wait until it finishes. Since for each client it will happen just once for a website, it is acceptable. DetecTor seems the easiest solution since it does not require additional hardware. TOR network has many nodes and operating well although the latency it has. But it will have a problem in closed networks and captive portals, hence, can not be an overall solution. But whatever, since that connection types are less than internet communications, it will help us in most of SSL connections (Note: For closed networks, CT is a good solution but the network administrators in that networks will have to operate a CT log server).

- **Pay attention to the big actors:** There is also another issue that must be taken into account: CAs' role in the proposed solution. CAs are the big actors in the current Web PKI, and earns big amounts of money by issuing certificates to clients. They will not be willing for anything that eliminates their role. Maybe the proposals' slow progress is due to this reality. Because most of the proposals require additional infrastructures, such as dedicated servers and management software which all requires resources. CAs, naturally, will not be volunteer for the projects such as Convergence and DANE, which eliminates CAs role.

  Hence, we have a different approach to the solution. This approach uses the idea of MECAI project. Since CAs are the big actors in current system that can afford money and resources that the ideal solution would require, also since they are very-experienced in the tasks in current Web PKI, we can put all the extra work onto their shoulders. In short, our suggestion is, to choose CT as the solution and expect all CAs (only root-CAs) to operate a CT log server. Also they must be responsible for auditing tasks in CT, auditing other log servers' behaviours. By the way, Google also focused on their CT project and has vision to force their Chrome browser to show warning for sites presenting no or wrong SCTs. As Google said, 94 percent of CAs have accepted to issue EV certificates with SCT. These are good advancement signs and makes us hopeful to reach the ideal solution we desire.

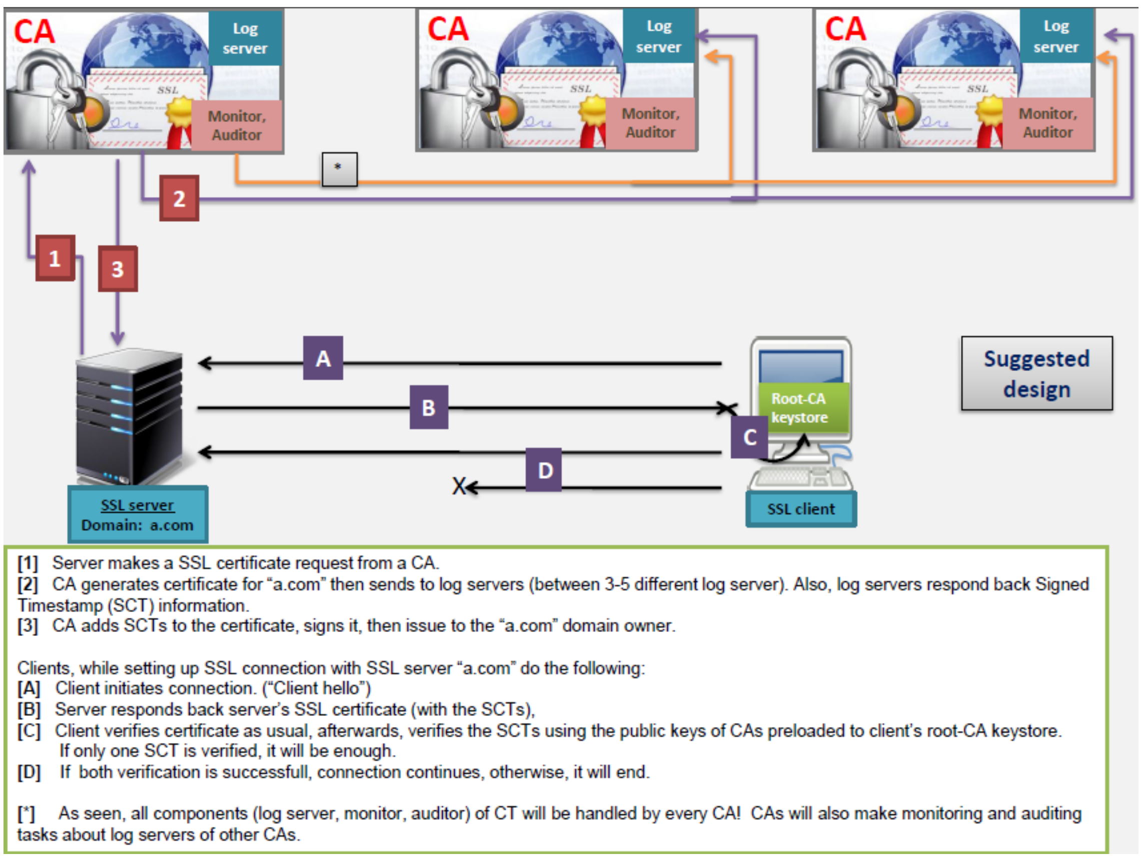
5.2 Our Suggestions

After reviewing the ideas that the proposals are based on, also their pros/cons, key features, fulfilment degree of the desired properties, and looking at the possible course of actions, here are our recommendations for the solution:

- Firstly, we think, CT and Pinning are the top 2 solutions, since they cover most of our desired properties. The others, especially the ones based on the notary idea (Convergence can be seen as the better solution in that category), can be the secondary methods helping the main solution.
- The easiest solution to implement is Pinning. Therefore, pinning must be in the solution. In order to provide flexibility to pinning, we should prefer TACK solution. But, TACK requires change in SSL protocol. Hence, until such change is accepted and implemented, we should start with another way.
- CT comes into play then. It should be in the solution because only CT provides us *impostor discoverability*, the feature that current CA system lacks. Furthermore, that feature makes attackers' job very hard. Attackers will have to add fraudulent certificates to log servers, which will be detected in very short time. In fact, such feature will also deter attackers to perform MitM attacks on SSL.
- These are the strong points of CT but recall that CT had the following cons: Providing required resources, privacy problem, log server key change problem, problem in closed networks and probability of unwillingness of the parties for handling the required tasks in CT. Now, we will explain how to overcome such problems in our suggestion:
  - We are inspired from the MECAI project, and give the tasks of implementing all required components of CT, such as log servers, auditors and monitors to every root-CA. This idea solves the privacy problem of CT (SSL clients were asynchronously checking the existence of SCT from the log server while SSL handshake, in CT proposal) since we have eliminated the audit task of SSL clients. Instead, we want every CA to inspect and audit the log servers of other CAs.
  - We think, this idea will be supported by CAs because it will help them to check the security of their issued-certificates. Also since they will easily afford the resources needed for those additional tasks, the solution will

be implemented in a short time, hopefully. By the way, some CAs have already accepted to operate a CT log server.

- Furthermore, since CAs will operate log servers, the problem of log server public key change is solved because there is currently a mature system to deal with changes of public keys of CAs, automatically via OS or browser updates.
- There will be needed some changes in SSL client software to verify the SCTs. For that, we hope, after Google, the other browser vendors start adopting such verification logic to their browsers.
- In closed networks, obviously, the operator of those networks will have to operate CT log server. Since there will be a few SSL servers in such networks, we think, operating such log server will not be troublesome.
- Furthermore, we can pin bullet-proof responses (such as when all SCTs are verified for a server) for the target SSL server at client cache, and use that pinning information to help us giving trust decision at subsequent visits to that server along with SCT verification.
- Additionally and optionally, if we want to make a more secure system, we can add notary server operating duty to CAs. Then, if SSL client can not verify all SCTs, he may try to ask to a 3 random notary server and decide according to the responses.
- Our suggestion is depicted in Figure 5.1:



[1] Server makes a SSL certificate request from a CA.
[2] CA generates certificate for "a.com" then sends to log servers (between 3-5 different log server). Also, log servers respond back Signed Timestamp (SCT) information.
[3] CA adds SCTs to the certificate, signs it, then issue to the "a.com" domain owner.

Clients, while setting up SSL connection with SSL server "a.com" do the following:
[A] Client initiates connection. ("Client hello")
[B] Server responds back server's SSL certificate (with the SCTs),
[C] Client verifies certificate as usual, afterwards, verifies the SCTs using the public keys of CAs preloaded to client's root-CA keystore. If only one SCT is verified, it will be enough.
[D] If both verification is successfull, connection continues, otherwise, it will end.

[*] As seen, all components (log server, monitor, auditor) of CT will be handled by every CA! CAs will also make monitoring and auditing tasks about log servers of other CAs.

Figure 5.1: Our suggested design.

5.3 Directions for Future Work

We suggest that future work for the subject should be done in the following topics:

- Nearly all proposals have problems in closed networks and captive portals. There should be some researches about how to implement suggested proposals in that cases.
- Notary-based solutions are successful but they need to eliminate side-channel need. Hence there should be some researches to give those proposals such ability.

REFERENCES

[1] R. Rivest, "Cryptology," *Handbook of Theoretical Computer Science 1*, 1990.

[2] *Kerckhoffs's principle*, accessed 12.04.2015. `http://www.crypto-it.net/eng/theory/kerckhoffs.html`.

[3] C. Shannon, "Communications theory of secrecy systems," *Bell Systems Technical Journal*, 1949.

[4] E. Chu, P. Kim, and P. Kim, *The Selection of The Advanced Encryption Standard*, 2000. `http://web.mit.edu/6.933/www/Fall2000/aes/AES_final_6933.pdf`.

[5] Y. L. Jonathan Katz, "Introduction to modern cryptography- principles and protocols," 2007.

[6] *Network Security Workshop*, (accessed 15.03.2015). `http://training.apnic.net/docs/WSEC01.pdf`.

[7] R. Holz, T. Riedmaier, N. Kammenhuber, and G. Carle, "X.509 forensics: Detecting and localising the ssl/tls men-in-the-middle," *Network Security*, 2012.

[8] T. Greene, *Father of SSL says despite attacks, the security linchpin has lots of life left*, (accessed 15.03.2015). `http://www.networkworld.com/article/2181968/security/father-of-ssl-says-despite-attacks--the-security-linchpin-has-lots-of-life-left.html`.

[9] I. Ristic, *Bulletproof SSL and TLS*. Feisty Duck, 2014.

[10] M. Stephanos and R. Raphael, "Certificates-as-an-insurance: Incentivizing accountability in ssl/tls," *NDSS Symposium*, 2015. `http://internetsociety.org/sites/default/files/01_6.pdf`.

[11] A. Sotirov, *MD5 Considered Harmful Today*, 2008. `https://www.win.tue.nl/hashclash/rogue-ca/downloads/md5-collisions-1.0.pdf`.

[12] X. W. A. Lenstra and B. de Weger, "Colliding x.509 certificates," *Cryptology ePrint Archive, Report 2005/067*, 2005.

[13] K. Michael and B. Joseph, "Upgrading https in mid-air: An empirical study of strict transport security and key pinning," *NDSS Symposium*, 2015. `www.internetsociety.org/sites/default/files/UpgradingHTTPSinMid-Air-`

AnEmpiricalStudyofStrictTransportSecurityandKeyPinning.pdf.

[14] R. Oppliger, “Ssl/tls session-aware user authentication: A lightweight alternative to client-side certificates,” 2007.

[15] P. Eckersley and J. Burns, *Is the SSLiverse a Safe Place?*, 2010 (accessed 14.02.2015). https://www.eff.org/files/ccc2010.pdf.

[16] B. L. K. N. C. G. Holz, R., “The ssl landscape: A thorough analysis of the x.509 web pki using active and passive measurements,” *11th Annual Internet Measurement Conference (IMC 11), Berlin, Germany*, 2011.

[17] E. S. A. H. A. N. C. L. Sunshine, J., “Crying wolf: An empirical study of ssl warning effectiveness.,” *18th USENIX Security Symposium*, 2009.

[18] S. S. Soghoian, C., “Certifed lies: Detecting and defeating government interception attacks against ssl.,” *15th. Int. Conf. Financial Cryptography and Data Security (FC 11)*, 2011.

[19] D. Goodin, *French agency caught minting SSL certificates impersonating Google.*, accessed 12.04.2015. http://arstechnica.com/security/2013/12/french-agency-caught-minting-ssl-certificates-impersonating-google.

[20] R. Moore and S. Ward, *Multiple Browser Wildcard Cerficate Validation Weakness*, 2010.

[21] M. MarlinSpike, *More Tricks For Defeating SSL In Practice*, 2009. https://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf.

[22] C. Meyer and J. Schwenk, “Sok: Lessons learned from ssl/tls attacks,” 2013. http://www.hgi.ruhr-uni-bochum.de/media/nds/veroeffentlichungen/2013/08/19/paper.pdf.

[23] M. Green, *TACK*, 2012 (accessed 14.02.2015). http://blog.cryptographyengineering.com/2012/05/tack.html.

[24] *TACK project website*, 2012 (accessed 04.02.2015). http://tack.io.

[25] M. Marlinspike and T. Perrin, *TACK Draft Document*, 2012 (accessed 04.02.2015). http://tack.io/draft.html.

[26] E. Hoz, R. Paez-Reyes, G. Cochrane, I. Marsa-Maestre, J. Moreira-Lemus, and B. Alarcos, *Detecting and Defeating Advanced Man-In-The-Middle Attacks against TLS*, 2014 (accessed 14.02.2015). https://ccdcoe.org/cycon/2014/proceedings/d2r2s2_delahoz.pdf.

[27] C. Evans, C. Palmer, and R. Sleevi, *Public Key Pinning Extension for HTTP draft-ietf-websec-key-pinning-21*, 2014 (accessed 14.02.2015). http://tools.ietf.org/html/draft-ietf-websec-key-pinning-21.

[28] *Public key pinning*, 2011 (accessed 14.02.2015). https://www.imperialviolet.org/2011/05/04/pinning.html.

[29] N. Elenkov, *Certificate pinning in Android 4.2*, 2012 (accessed 14.02.2015). http://nelenkov.blogspot.com.tr/2012/12/certificate-pinning-in-android-42.html.

[30] *FAQ*, (accessed 14.02.2015). http://www.certificate-transparency.org/faq.

[31] *What is Certificate Transparency?*, (accessed 14.02.2015). http://www.certificate-transparency.org/what-is-ct.

[32] B. Laurie, A. Langley, and E. Kasper, *Certificate Transparency*, 2013 (accessed 14.02.2015). http://tools.ietf.org/html/rfc6962.

[33] P. Hoffman and J. Schlyter, *DANE Transport Layer Security (TLS) Protocol: TLSA*, 2012 (accessed 14.02.2015). http://tools.ietf.org/html/rfc6698.

[34] M. Rouse, *Cache poisoning*, 2005 (accessed 14.02.2015). http://searchsecurity.techtarget.com/definition/cache-poisoning.

[35] P. S. I. Danhieux, *Email Phishing Attacks*, 2013 (accessed 14.02.2015). http://www.securingthehuman.org/newsletters/ouch/issues/OUCH-201302_en.pdf.

[36] M. Southam, "Dnssec: What it is and why it matters," *Network Security*, 2014.

[37] *ICANN*, (accessed 14.02.2015). https://www.icann.org/.

[38] *DNSSEC – What Is It and Why Is It Important?*, (accessed 14.02.2015). https://www.icann.org/resources/pages/dnssec-qaa-2014-01-29-en.

[39] *TLD DNSSEC Report*, 2015 (accessed 14.02.2015). http://stats.research.icann.org/dns/tld_report/.

[40] *Securing End-to-End Internet communications using DANE protocol*, (accessed 14.02.2015). http://www.afnic.fr/medias/documents/Dossiers_pour_breves_et_CP/dossier-thematique12_VEn1.pdf.

[41] *TLSA Record Generator*, (accessed 15.02.2015). https://ssl-tools.net/tlsa-generator.

[42] R. Barnes, *Use Cases and Requirements for DANE*, 2011 (accessed 14.02.2015). http://tools.ietf.org/html/rfc6394.

[43] C. Brown and M. Jenkins, "Analyzing proposals for improving authentication on the tls/ssl-protected web," 2014.

[44] D. York, *Adding DNSSEC to Fedora and Red Hat Linux*, 2012 (accessed 14.02.2015). http://www.internetsociety.org/deploy360/blog/2012/11/slides-adding-dnssec-to-fedora-and-red-hat-linux/.

[45] *DNSSEC/TLSA Validator*, 2013 (accessed 14.02.2015). https://www.DNSsec-validator.cz.

[46] D. York, *How To Add DNSSEC Support To Google Chrome*, 2012 (accessed 14.02.2015). http://www.internetsociety.org/deploy360/resources/how-to-add-DNSsec-support-to-google-chrome.

[47] *Tor Project*, (accessed 14.02.2015). https://www.torproject.org/.

[48] D. Wendlandt, D. Andersen, and A. Perrig, "Perspectives: Improving ssh-style host authentication with multi-path probing," 2008. http://www.cs.cmu.edu/~perspectives/perspectives_usenix08.pdf.

[49] M. Marlinspike, *DefeatSSL*, 2009 (accessed 14.02.2015). https://media.blackhat.com/bh-usa-09/video/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-VIDEO.mov.

[50] A. Grant, *Search for Trust: An Analysis and Comparison of CA System Alternatives and Enhancements*, 2012 (accessed 14.02.2015). http://www.cs.dartmouth.edu/reports/TR2012-716.pdf.

[51] *Convergence*, (accessed 14.02.2015). http://convergence.io/.

[52] M. Marlinspike, *SSL and the future of authenticity*, 2009 (accessed 14.02.2015). http://www.thoughtcrime.org/blog/SSL-and-the-future-of-authenticity/.

[53] T. Greene, *SSL certificate authorities vs. ???*, 2011 (accessed 14.02.2015). http://www.networkworld.com/article/2182669/security/ssl-certificate-authorities-vs-----.html.

[54] E. Messmer, *The SSL certificate industry can and should be replaced*, 20– (accessed 14.02.2015). http://www.networkworld.com/article/2182059/security/the-ssl-certificate-industry-can-and-should-be-replaced.html.

[55] *Convergence*, (accessed 14.02.2015). http://convergence.io/details.html.

[56] *Convergence - github*, 2015 (accessed 14.02.2015). https://github.com/mk-fg/convergence.

[57] M. Schwartz, *New SSL Alternative: Support Grows for Convergence*, 2011 (accessed 14.02.2015). http://www.networkcomputing.com/networking/new-SSL-alternative-support-grows-for-convergence/d/d-id/1100471.

[58] *Convergence Notaries*, 2014 (accessed 14.02.2015). https://github.com/moxie0/Convergence/wiki/Notaries.

[59] K. Engert, *DetecTor.io*, 2013 (accessed 14.02.2015). http://detector.io.

[60] K. Engert, *DetecTor proposal video*, 2013 (accessed 14.02.2015). http://detector.io/download/30c3-detector.io-kaie-lightning-talk-h264-hq.mp4.

[61] *The EFF SSL Observatory*, (accessed 14.02.2015). https://www.eff.org/observatory.

[62] *Nmap - Free Security Scanner For Network Exploration, Security Audits.*, (accessed 14.02.2015). http://nmap.org/.

[63] R. Lemos, *EFF builds system to warn of certificate breaches*, 2011 (accessed 14.02.2015). http://www.infoworld.com/article/2620473/encryption/eff-builds-system-to-warn-of-certificate-breaches.html.

[64] *ICSI*, (accessed 14.02.2015). http://icsi.berkeley.edu/.

[65] *The ICSI Certificate Notary*, (accessed 14.02.2015). http://notary.icsi.berkeley.edu/.

[66] *The Bro Network Security Monitor*, (accessed 14.02.2015). https://www.bro.org.

[67] C. Jeremy and O. Paul, "Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements," *IEEE Symposium on Security and Privacy*, 2013. http://www.ieee-security.org/TC/SP2013/papers/4977a511.pdf.

[68] *A Comparison of HTTPS Reforms*, 2013 (accessed 15.02.2015). http://www.scriptjunkie.us/2013/12/a-comparison-of-https-reforms/.

[69] M. Gielesberger, "Alternatives to x.509," 2012. http://www.net.in.tum.de/fileadmin/TUM/NET/NET-2013-02-1/NET-2013-02-1_07.pdf.

[70] M. B. Henning Perl, Sascha Fahl and M. Smith, *A Qualitative Comparison of SSL Validation Alternatives*, 2013 (accessed 15.02.2015). https://www.owasp.org/images/d/d4/A_Qualitative_Comparison_of_SSL_Validation_Alternatives_-_Henning_Perl%2BMichael_Brenner%2BMathew_Smith.pdf.

[71] M. A. Italo Dacosta and P. Traynor, "Trust no one else: Detecting mitm attacks against ssl/tls without third-parties," *Computer Security*, 2012. http://www.cise.ufl.edu/~traynor/papers/dacosta-esorics12.pdf.

[72] *Comparison with Other Technologies*, (accessed 15.02.2015). http://www.certificate-transparency.org/comparison.