ANALYSIS OF RECENT ATTACKS ON SSL/TLS PROTOCOLS

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

BY

DUYGU ÖZDEN

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

SEPTEMBER 2016

Approval of the thesis:

ANALYSIS OF RECENT ATTACKS ON SSL/TLS PROTOCOLS

submitted by **DUYGU ÖZDEN** 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** _____

Assoc. Prof. Dr. Murat Cenk
Supervisor, **Cryptography, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Murat Cenk
Cryptography, METU _____

Assoc. Prof. Dr. Ali Doğanaksoy
Mathematics, METU _____

Asst. Prof. Dr. Fatih Sulak
Mathematics, ATILIM UNIVERSITY _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: DUYGU ÖZDEN

Signature :

ABSTRACT

ANALYSIS OF RECENT ATTACKS ON SSL/TLS PROTOCOLS

Özden, Duygu
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Murat Cenk

September 2016, 46 pages

Transport Layer Security(TLS) and its predecessor Secure Socket Layer(SSL) are two important cryptographic, certificate based protocols that satisfy secure communication in a network channel. They are widely used in many areas such as online banking systems, online shopping, e-mailing, military systems or governmental systems. Being at the center of secure communication makes SSL and TLS become the target of attackers and an important field of study for researchers. So many vulnerabilities and attacks towards these protocols were explored from past to present. In this thesis, we will mention about the design of SSL and TLS, the cryptographic algorithms used in them, important and recent attacks on these protocols with their precautions. At the end, we will touch on the important points and the selection of parameters for their design that will give strong ideas for the future works to fix these vulnerabilities and improve the protocols.

Keywords : attacks, cryptographic certificate based protocols, algorithm, selection of parameters, vulnerability

ÖZ

SSL/TLS PROTOKOLLERİNE YAPILAN SON ATAKLARIN ANALİZİ

Özden, Duygu
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Murat Cenk

Eylül 2016, 46 sayfa

Transport Layer Security(TLS) ve onun öncülü Secure Socket Layer(SSL), bir ağ kanalında güvenli iletişim sağlayan iki önemli kriptografik, sertifika tabanlı protokoldür. Bu protokoller; internet bankacılığı, elektronik alışveriş, elektronik postalama, askeri sistemler ve devlet sistemlerinde yaygın olarak kullanılmaktadır. Güvenli iletişimin merkezinde olmak, SSL ve TLS' i saldırganların hedefi haline getirmekte ve araştırmacılar için önemli bir çalışma alanı olmaktadır. Geçmişten bugüne, bu protokollere karşı bir çok zafiyet ve atak keşfedilmiştir. Bu tezde, SSL ve TLS' in dizaynından, onların içerisinde kullanılan kriptografik algoritmalardan, önemli ve güncel ataklardan ve önlemlerinden bahsedeceğiz. Son olarak, ilerideki çalışmalarda bu atakları düzeltmek ve protokolleri geliştirmek için güçlü fikirler vermek amacıyla, protokollerin dizaynındaki önemli noktalara ve parametrelerin seçimine değineceğiz.

Anahtar Kelimeler : ataklar, kriptografik sertifika tabanlı protokoller, algoritma, parametrelerin seçimi, zafiyet

To My Family

ACKNOWLEDGMENTS

At first, I would like to express my special thanks to my supervisor Assoc. Prof. Dr. Murat Cenk for his positive attitude, endless support, trust, encouragement and valuable guidance during the period of writing and improving this thesis. His advices enlighten me to produce much better study than I can. I am so happy to have a favorable supervisor like him.

Secondly, I would like to say my appreciation to my friend Adnan Kılıç for his helps in programming languages not only during my thesis development but also during my education.

Finally, I would like to thank my family and my close friends for their endless support, love and trust as always. They encouraged me during my education life even in stressful times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SSL	Secure Socket Layer
TLS	Transport Layer Security
DHKE	Diffie-Helmann Key Exchange
RSA	Rivest Shamir Algorithm
ECC	Elliptic Curve Cryptography
SHA1	Secure Hash Algorithm 1
MD5	Merkle Damgard Algorithm 5
HMAC	Hash based Message Authentication Code
PKI	Public Key Infrastructure
DROWN	Decrypting RSA with Obsolete and Weakened Encryption
SLOTH	Security Losses from Obsolete and Truncated Transcript Hashes
SMACK	State Machine Attacks
FREAK	Factoring RSA Export Keys
RC4	Rivest Cipher 4
AES	Advanced Encryption Standard
SSH	Secure Shell
OpenPGP	Pretty Good Privacy
CSR	Certificate Signing Request
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
CPU	Central Processing Unit
KSA	Key Scheduling Algorithm
PRGA	Pseudo-Random Generation Algorithm
DES	Data Encryption Standard
NSA	National Security Agency
NIST	National Institute of Standards and Technology
DLP	Discrete Logarithm Problem
CA	Certification Authority
IETF	Internet Engineering Task Force
RFC	Request for Comments

TCP/IP	Transmission Control Protocol/Internet Protocol
HTTP	Hypertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol
NNTP	Network News Transfer Protocol
CCS	ChangeCipherSpec
CBC	Cipher Block Chaining
CRT	Chinese Remainder Theorem
IEEE	Institute of Electrical and Electronics Engineers
IMDEA	The French Institute for Research in Computer Science and Automation
FBI	Federal Bureau of Investigation
IBM	International Business Machines
VPN	Virtual Private Network
ENISA	European Network and Information Security Agency
GCM	Galois/Counter Mode
DTLS	Datagram Transport Layer Security

CHAPTER 1

INTRODUCTION

Technology and its fast development in recent years bring many opportunities in our lives. Nearly the most important one of them is the discovery of internet. It started to be improved in 1960s but it has begun to be centrally located in daily life since 1990s. The advantages of internet like easy access to information, communication, online banking, e-commerce and many others enhance the life standards. On the other hand, the internet has also some drawbacks as much as its benefits. One of them is related to the security point of view. For example, if people use internet for shopping, banking transactions or even for communication, they sometimes need to give their credit card numbers for payment. Another example is downloading or sharing some applications, photos, personal informations without using a secure channel. Some websites are not secure enough for these types of operations. In the earlier stages of internet, system developers or users are not aware of these issues. However, the rise of plagiarism in all areas causes stealing money, information or even reliability which makes people awaken. At this point, the essentialness of making internet more and more secure arises. The internet uses some protocols that are formats or rules of digital messages for exchanging data between computers across a single network or a series of interconnected networks. See [5] . One of the most important internet protocols are certificate based protocols such as Secure Socket Layer(SSL) or Transport Layer Security(TLS). They are used for ensuring security of communication between the client(service requesters) and the server(the providers of a resource or service) over an insecure channel. SSL is developed by Netscape Communications in 1994 to allow secure access of a browser to a web server and became the accepted standard for web security. See [6] . The version 1 of SSL, named SSL 1.0, was never released due to some deficiencies. After that, SSL 2.0 and SSL 3.0 are developed respectively. TLS, the following version of SSL, was developed in 1999 and mentioned in RFC(Request for Comments). The current versions are TLS 1.0, TLS 1.1, TLS 1.2 and TLS 1.3 is a working draft. The name of the protocol changed from SSL to TLS because TLS works over any bidirectional stream of bytes, not just sockets. See [7] .

Besides of the fact that SSL and TLS are a computer system invention, they are also cryptologic tools in order to satisfy confidentiality, data integrity, authentication and non-repudiation provided by cryptology. Confidentiality means that the data or the message is disclosed to unauthorized people or systems. Data integrity is maintaining the accuracy and completeness of the message. Entity authentication is the corroboration of the identity of an entity and message authentication is corroborating the source

of information; also known as data origin authentication. Non-repudiation means preventing the denial of previous commitments or actions [40]. Cryptology consists of two parts: cryptography and cryptanalysis. Cryptography comprises of asymmetric cryptography (public key cryptography) and symmetric cryptography. Public key part means that two keys are used: the public one and the private one. This system has some algorithms and signature schemes inside itself. The symmetric cryptography uses one key inside its algorithms and the algorithms have two types: block ciphers and stream ciphers. Hash functions and pseudo-random sequences are also valuable parts of cryptography. In addition, basic terms of cryptology are plaintext which is the original message, ciphertext which is the coded message, cipher which is the algorithm for transforming plaintext to ciphertext, key which is the information used in cipher and known only by the sender or the receiver, encryption which is converting plaintext to ciphertext and decryption which is recovering ciphertext from plaintext [29]. Cryptanalysis is a study of cryptosystems for finding or noticing some weaknesses or vulnerabilities that permits capturing the plaintext from ciphertext without knowing the key or obtaining the whole key or a part of the key.

Since the data transferred over an insecure channel, a third person could access the data if SSL/TLS user does not use a strong encryption or an algorithm inside the certificate. In other words, the system remains open to the man in the middle attacks. Although there are some other vulnerabilities or attacks on these systems like configuration mistakes or implementation flaws, the most important one relies on the usage of cryptography in SSL/TLS certificates. Cryptography in SSL/TLS uses symmetric and asymmetric encryption algorithms, hash functions and hash based functions and key exchange mechanisms. The most popular symmetric encryption algorithms are RC4 and AES, the asymmetric encryption algorithms are RSA and Elliptic Curve Cryptography and the hash functions and hash based functions are SHA1, MD5, MAC and HMAC. In addition to these, the key exchange mechanisms used inside certificate have a great importance for not exploiting the system. The significant one of the key exchange mechanisms is named as Diffie-Hellman Key Exchange. Improvements in the technology show that the security of internet usage and not exposing the related attacks depend on both using strong cryptography and building a good system.

In this thesis, we investigated certificate based protocols and focused on SSL/TLS and recent attacks on them.

In Chapter 2, some popular cryptosystems inside these certificates were mentioned according to their types and usages.

In Chapter 3, first, the historical development of SSL/TLS were told shortly and the development processes were given as a table. After that, the public key infrastructure which is the key point of designing a certificate is mentioned. Then, how SSL and TLS works was showed in detail. Last, attacks especially the most effective ones were told in brief.

In Chapter 4, the attack types and the recent attacks on SSL and TLS were given briefly. We focused on the security of these systems, the parameters and the cryptographic infrastructure of the attacks.

In Chapter 5, we investigated important points and selection of parameters on these protocols. The currently recommended key sizes and parameters in algorithms and protocols were given. After that, the protection methods from recent attacks and some other known attacks towards SSL/TLS told shortly.

CHAPTER 2

PRELIMINARIES

2.1 Popular Cryptosystems Used in SSL/TLS

SSL/TLS certificates provides secure communication as it is said before. But what is the protocol behind it? How does it work? The main security object is the encryption. The algorithms used for encrypting transmitted data and session keys should be strong enough in case they would not be exploited. This means that a powerful cryptography is needed. The SSL communication uses both symmetric and asymmetric cryptography for the connection. Additionally, it uses some algorithms that create keys.

2.1.1 Asymmetric Cryptography

Asymmetric cryptography, known also as public key cryptography, uses two types of keys. One is public key, known by anybody, the other one is private key, known only by the owner of the system. Popular algorithms used inside SSL and TLS are RSA and Elliptic Curve Cryptography described in below.

2.1.1.1 RSA

The RSA is one of the most popular asymmetric cryptosystems with its common usage in many systems and protocols such as digital signature schemes, SSH, OpenPGP, SSL/TLS, etc. to achieve data transmission, authentication or encryption securely. It was discovered by Ron Rivest, Adi Shamir and Leonard Adleman in 1978 and named with the initial letters of their surnames. The security of RSA cryptosystems relies on the difficulty of integer factorization. The reason for this is the asymmetric cryptography uses two types of keys. One of them is shared publicly and the other one is kept secret. The user of RSA cryptosystems chooses two large primes and calculates the multiple of them. The multiple of these numbers and an encryption key are shared publicly and the prime factors and the decryption key are kept secret. In this way, it is hard to decrypt the message even if the public parameters are known.

The RSA Algorithm has the key generation part, encryption part and decryption part.

a. Key Generation:

-As a first step, choose two distinct prime numbers p and q randomly. Yet, it is important to choose the primes close to each other but they could have different lengths by a few digits. The reason is to make factorization more difficult. Also, large primes could be effectively found by using primality tests. The prime numbers should be private.

-Compute $n = p.q$ in order to use as a modulus for public and private keys. It could be shared publicly.

-Compute $\phi(n)$ which is Euler' s totient function and is equal to $(p-1)(q-1)$. This value should be private.

-Choose a large integer e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$. This value is public.

-Calculate the integer d by using the equation $d.e \equiv 1(mod\phi(n))$ which means that d is the modular multiplicative inverse of e. This value of d also should be private.

As a result, the RSA user get two keys for asymmetric encryption;

Public key: n, e Private key: p, q, d

b. Encryption:

-The message M should be converted to the integer m as a first step. Note that, the integer m should satisfies that $0 \leq m < n$ and $gcd(m, n) = 1$ by using padding schemes.

-Calculate the ciphertext value $c \equiv m^e mod n$.

c. Decryption:

-To get the original message, calculate $c^d \equiv (m^e)^d \equiv m\ mod(n)$.

-After finding m, one could easily recover the message M by reversing padding scheme.

d. RSA Usage in SSL/TLS:

The key obtained by using RSA algorithm is used for authentication and symmetric key exchange in SSL/TLS. It relies on the Public Key Infrastructure(PKI) which is commonly used in certificate design. An SSL certificate includes public and private keys coming from asymmetric encryption algorithm that is assumed by PKI. By using Certificate Signing Request(CSR), the RSA private key is generated. The key size should be at least 1024-bit and especially in today' s computation power, it could be 2048-bit or larger.

Table 2.1: NIST Recommended Key Sizes

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

2.1.1.2 Elliptic Curve Cryptography(ECC)

An elliptic curve is the set of points described by the equation below:

$y^2 = x^3 + ax + b$ where $4a^3 + 27b^2 \neq 0$

Elliptic Curve Cryptography relies on finding the discrete logarithm of a random elliptic curve. Some protocols based on discrete logarithm is used with the elliptic curves. The most popular ones are Elliptic Curve Diffie Hellman(ECDH) that is used for encryption and Elliptic Curve Digital Signature Algorithm(ECDSA)that is used for digital signing.

Elliptic curves have smaller key size, yet the key is stronger especially compared with RSA and because of this it is easier to implement and efficient to use. However, since they are not commonly used, the certificates generally do not support them when compared to RSA.

From the certificate point of view, smaller data is transmitted from server to client during the handshake process because the key size is smaller when compared to RSA. This brings another facileness: less CPU and memory is needed which increases network performance. Cryptographers and system developers think that when the advantages of elliptic curves are taking into account, they will be very popular by creating stronger cryptography.

2.1.2 Symmetric Cryptography

Symmetric cryptography, uses just one key for both encryption and decryption. There are two types of symmetric cryptography: Stream Ciphers and Block Ciphers. Popular stream cipher algorithm used inside SSL and TLS is RC4 and block cipher algorithm is AES, both of them described in below.

2.1.2.1 Stream Cipher

-RC4

This algorithm is developed by Ron Rivest and started to be used in 1994. Its ease of use in many applications made this algorithm very popular and used in many cryptosystems.

RC4 is a stream cipher and consists of two algorithms: Key Scheduling Algorithm(KSA) and Pseudo-Random Generation Algorithm(PRGA). The keystream generated with this algorithm is used for encryption by bitwise xoring with the plaintext. The decryption is also same logic meaning that ciphertext is xored with the keystream.

This algorithm is used in SSL/TLS communication due to some advantages. One advantage is that it does not require padding operation which strengthens it against TLS attacks like BEAST. The other advantage is its efficiency and fast usage which brings less computation and lower hardware requirements. On the other hand, the algorithm brings some drawbacks like small key size (128-bit etc.) and the key is not completely random which causes small bias. Therefore, RC4 algorithm is no longer considered as secure enough and is not commonly used in SSL/TLS communication.

2.1.2.2 Block Ciphers

-AES

The Advanced Encryption Standard or known as Rijndael is developed by Joan Daemen and Vincent Rijmen and published first in 1998. Its predecessor is DES algorithm but it is not commonly used in today' s technology because it is not very strong against some attacks. AES, on the other hand, became very popular and applied in many systems and considered to be secure enough.

a.Description:

-KeyExpansion: The round keys are derived from the key schedule of AES.

-InitialRound: AddRoundKey step means the bitwise xor of key to the plaintext.

-Rounds: SubBytes, ShiftRows, MixColumns, AddRoundKey are the one round of AES algorithm.

-Final Round: SubBytes, ShiftRows and AddRoundKey is the last round of the algorithm.

-Note that, AES algorithm has 10,12 or 14 rounds with 128-bit,192-bit or 256-bit Keys respectively. The most common one is 10 round AES with 128-bit key.

b.Usage of AES in SSL/TLS:

AES was started to use in TLS connection to make the system strong enough. Before

TLS, SSL systems, specially SSL 3.0, the most powerful one of SSL, uses 3DES, but it is also not secure enough against some attacks. Therefore, encryption in TLS communication generally uses AES algorithm.

2.1.3 Cryptographic Hash Functions and Hash Based Functions

A hash function takes a message and outputs to the fixed-size string (or message digest) that is called hash value. The function is a one-way function and satisfies pre-image resistance (given a hash value h, it is difficult to find the message of this hash value), second pre-image resistance (given a message m_1, it is difficult to find another message m_2 different from m_1 satisfying $h(m_1) = h(m_2)$) and collision resistance (it is difficult to find two different messages giving same hash value) properties. Hash functions do not use a key. In addition, hash based functions consist of a cryptographic hash function with a secret key. Popular hash functions used inside SSL and TLS are SHA1 and MD5, a hash based function is HMAC(Hash Based Message Authentication Code) described in below.

2.1.3.1 SHA1

The Secure Hash Algorithm 1 is designed by United States National Security Agency (NSA) and published by the United States National Institute of Standards and Technology (NIST). It produces fixed (160-bit) hash value named as message digest.

This algorithm is widely used in SSL/TLS connections but it is not considered as secure enough towards some attacks so it is thought that SHA1 algorithm will no longer be used until 2017. Instead, SHA256, which outputs 256-bit fixed length message will be considered in the connections of certificates.

2.1.3.2 MD5

This algorithm is designed by Ronald Rivest in 1991. Its structure is Merkle-Damgard construction which gives its name. It produces 128-bit fixed hash value.

The MD5 algorithm is also widely used in SSL/TLS connections as a signature algorithm in the certificate. Although its common usage in the design of certificates, the algorithm is vulnerable against some attacks. Therefore, MD5 is no longer recommended.

2.1.3.3 HMAC

Hash-based Message Authentication Code is a type of Message Authentication Code(MAC) with the usage of a cryptographic hash function with a secret key. The output of HMAC

function has the same length with the hash function used in the system. For example, HMAC-MD5 outputs 128-bit fixed hash value as in MD5 algorithm.

SSL uses MAC algorithm while TLS uses HMAC which is more secure and newer. While MAC satisfies authentication, HMAC satisfies both authentication and data integrity. TLS 1.0 and 1.1 uses HMAC-MD5 and HMAC-SHA-1 inside the handshake protocol. However, TLS 1.2 recommends that the hash function in HMAC should be at least 256-bit length, which means SHA-256 should be used.

2.2 Key Exchange Mechanisms

This is a cryptographic method for changing the key between the sender and the receiver through a secure channel. The most popular key exchange mechanism for SSL and TLS is Diffie-Hellman Key Exchange described in below.

2.2.1 Diffie-Hellman Key Exchange

This protocol is developed by Whitfield Diffie and Martin Hellman in 1976. In this system, two participants A and B want to communicate each other securely and for this reason, they want to produce and share a random secret key in public but authenticated channel.

Firstly, they agree on a cyclic group G of order n in $\mod p$ where p is a prime number and a generator g of the group G where g is a primitive root $\mod p$. These values could be shared publicly. Note that, the computations should be done under $\mod p$ arithmetic.

Secondly, A chooses an integer a , $1 \leq a \leq n$ as a private key and compute g^a then sends it to B. Similarly, B chooses an integer b , $1 \leq b \leq n$ as a private key and compute g^b then sends it to A.

Thirdly, A takes g^b and computes $(g^b)^a$ which is g^{ab} and B takes g^a and computes $(g^a)^b$ which is again g^{ab}. This g^{ab} is their shared key at the end.

It is important that the values of g, g^a, g^b are public. The only private keys here are a and b. Computing g^{ab} with using g^a and g^b but without using a and b is a hard problem named as Diffie-Hellman Problem. In addition, computing the values a or b with the knowledge of g and g^a or g^b is named as Discrete Logarithm Problem. This shows us that solving Discrete Logarithm Problem means breaking the Diffie Hellman Protocol. However, it is still an open problem whether Diffie-Hellman Problem and Discrete Logarithm Problem is equivalent or not. See [37]

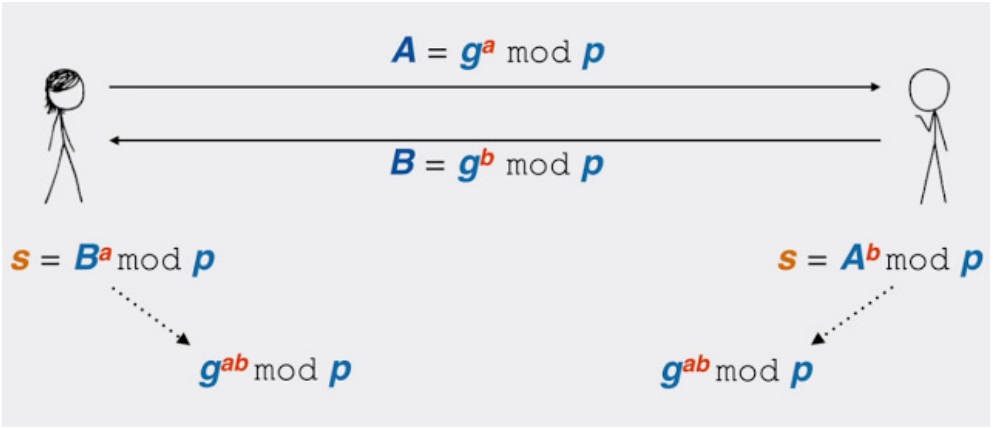

Figure 2.1: Diffie-Hellman Process

2.2.1.1 The Usage of Diffie-Hellman Algorithm in SSL/TLS

The Diffie-Hellman Protocol is an important key exchange mechanism used inside SSL and TLS. The system has three versions: Anonymous Diffie Hellman, Fixed Diffie Hellman and Ephemeral Diffie Hellman.

2.2.1.2 The Anonymous Diffie Hellman

The procedure uses Diffie Hellman without authentication. This could lead to man in the middle attacks. For this reason, anybody should avoid using this anonymous one. The way of doing this is adding "!ADH" command inside the code SSL_set_cipher_list. If you use The Anonymous Diffie Hellman, a call to SSL_get_peer_certificate returns NULL.

2.2.1.3 Fixed Diffie Hellman

In this procedure, certificate contains server' s public key parameters and certification authority(CA) signs the certificate. The parameters never changed.

2.2.1.4 Ephemeral Diffie Hellman

In this procedure, each run of the protocol uses different public keys. The security of these temporary keys could be checked with the signature on the key. Using temporary keys in the session could lead to protect the privacy of the previous sessions even if the server' s key is signed for a long term. This is called as "Perfect Forward Secrecy". Therefore, this system should be preferred. See [3]

2.3 Authenticated Encryption Types inside SSL/TLS

2.3.1 Encrypt-then-MAC

In this system, the message is encrypted first and then the ciphertext produced by encryption is used for applying MAC with a suitable block cipher. Inside TLS, both the ciphertext and MAC are sent between two parties. The system was explained in RFC 7366 in detail. This method is one of the strongest authenticated encryption type if the MAC algorithm is strong enough.

2.3.2 Encrypt-and-MAC

In this system, MAC is generated according to the message and also the message is encrypted without using MAC. After that, both of two versions are sent in the system. This method is generally used in some SSH systems.

2.3.3 MAC-then-Encrypt

In this system, MAC is generated according to the message and the message and MAC are encrypted together. Then both ciphertext and encrypted version of MAC are sent in the system.

2.3.4 MAC-then-Encode-then-Encrypt

In this system, MAC is used to the message and then message is encoded into a bit-string by using some encoding rules. After that, encryption is applied to the output of the encoding. For more detail, see [36].

CHAPTER 3

CERTIFICATION BASED PROTOCOLS: SSL/TLS

3.1 Historical Development of SSL/TLS

SSL contains the versions 1.0, 2.0 and 3.0 in terms of its historical development and then it was converted to TLS protocol in order to make it more secure. TLS, on the other hand, has the versions 1.0, 1.1 and 1.2 respectively and with the latest version of TLS, named as TLS 1.3, designers make great strides in providing privacy and data integrity.

SSL 1.0

This version was formed in 1994 by Netscape web browser which belongs to the Netscape Communications Corporation. However, it came in for criticism and could not be opened for public use because it contained pretty much vulnerabilities.

SSL 2.0

Netscape Communications released SSL 2.0 in February 1995 after making some arrangements in the previous version. Yet, many vulnerabilities were explored again in SSL 2.0.

SSL 3.0

After the first two versions failed in practice, this version was developed by Paul Kocher and Alan Freier. The group, called IETF(Internet Engineering Task Force) and standardize the internet protocols, published this version in 1996. However, this version again has considered unsafe since 2014, because it is open to a variety of attack methods affecting block encryption. This version is only supported by the RC4, a stream cipher algorithm but this algorithm is not enough to make SSL 3.0 reliable since it can be exposed to various attacks.

TLS 1.0

TLS 1.0 was defined in January 1999, inside the documents named as Request For Comments(RFC) and used in identifying TCP/IP (in the RFC 2246 version). This design was created by Christopher Allen and Tim Dierks and was obtained by the development of SSL 3.0 but does not have the capability of interoperability.

TLS 1.1

This version is defined in RFC 4346 in April 2006 and it is the updated version of TLS 1.0.

TLS 1.2

This version is defined in RFC 5246 in August 2008. It is also a protocol created by a number of changes on the previous version, TLS 1.1.

TLS 1.3 (incomplete) This version, started to be designed in March 2015 with several changes on TLS 1.1 and 1.2, has not yet completed its development.

3.2 Public Key Infrastructure(PKI)

In order to make authentication in SSL protocol, one of the most important part is the necessity of public key infrastructure. It is a set of policies that creates and manages digital certificates and public key encryption. In terms of cryptography, public key infrastructure is a body of rules that provides contact between the public keys and the users via certification authority(CA). The informations came from certification authority should be unique up to users in their jurisdiction. The public key infrastructure has three main parts:

1.Certification Authority, which is responsible for giving and verifying digital certificates.

2.Registration Authority, which takes requests from users in the name of certification authority and check on informations came from users.

3.Central Directory, which is the secure storage of informations belong to the certificates.

3.3 How does SSL/TLS protocol work?

3.3.1 SSL protocol

This protocol contains four layers: The Record Layer, ChangeCipherSpec Protocol, Alert Protocol and Handshake Protocol. Let us explain how they work:

Record Layer

This layer contains a header for the message and a hash value formed by MAC. The parts of the layer encapsule the definition and the version of protocol and the length of the message.

ChangeCipherSpec Protocol

This layer represents the beginning of secure communication. In other words, this section says that the data transferred in the communication is changed from unencrypted to encrypted one.

Alert Protocol

This layer explains the warnings about the communication in two steps: Severity Level and Alert Description.

-Severity Level sends messages to the participants by using the values ”1” and ”2”. If the message comes as the value ”1”, it means that the connection between the participants is closed down and they should reconnect to each other. On the other hand, if the message comes as the value ”2”, it means that an important failure occurs and the client and the server should not continue to the communication.

-Alert Description means that a particular error message coming from one side of the communication occurs. The message could include one the following errors: CloseNotify, Handshake Failure, CertificateRevoked, Unexpected Message, NoCertificate, CertificateExpired, BadRecordMAC, BadCertificate, CertificateUnknown, DecompressionFailure, UnsupportedCertificate, IllegalParameter. See [39]

Handshake Protocol

In order to communicate securely, the handshake protocol is used between the client(user’ s browser) and the server(web application). The process is explained step by step below.

1.Client Hello

The first step is a request called the Client Hello since the user wants to communicate with the server. This contains the SSL version number, the cipher settings supported by the client and session-specific data.

2.Server Hello

The second step is the answer of the server to the client hello message in order to communicate with SSL. This section also contains SSL version number, the cipher settings and session-specific data. Additionally, this step includes the public key of the server as a server’ s certificate.

3.Server Key Exchange

Since there is no algorithm to agreed upon, the information is sent with no encryption. The public key of server is used to encrypt a session key which is generated for the secure communication. It is important to say that both the client and the server will use the same key to encrypt session-specific data. On the other hand, digital certificates are used to ensure that the right client wants to communicate with the right server. This certificates include public keys (never private ones) and connect it to the certificate owner.

4.Server Hello Done

When the server key exchange finished, the client receives the server hello done message which shows that the server took her message.

5.Client Key Exchange

This message has an information about the key that the client and the server will use to communicate each other.

6.Change Cipher Spec

This step change the session-specific data transmission from an insecure channel to a secure channel.

7.Finished

After all these steps finish, the secure communication via SSL completed.

3.3.2 TLS protocol

This protocol contains two layers: The Record Protocol and The Handshake Protocol. Let us explain how they work:

The Record Protocol

This part of TLS communication is used to satisfy secret and reliable communication between the participants. The protocol benefits from symmetric key cryptography even if it could use no encryption. By the usage of MAC, the hash functions are generated and the connection uses this hash function to satisfy privacy.

The Handshake Protocol

This protocol satisfy authentication between the client and the server. This means that the participants compromise over an encryption algorithm and the keys before sending the message to each other. The handshake procedure is same with the one used in SSL communication.

CHAPTER 4

RECENT ATTACKS ON SSL/TLS

Even though SSL and TLS protocols make progress about security along their historical development, they have still some vulnerabilities towards many attacks. In this chapter, the recent vulnerabilities and attacks will be explained in detail which could lead the future works to improve them.

4.1 The Attack Types

The attacks on SSL/TLS can be expressed with 4 categories:

-Protocol Logic Flaws

-Cryptographic Design Flaws

-Implementation Flaws

-Configuration and Infrastructure Flaws

4.1.1 Protocol Logic Flaws

The protocols of SSL/TLS could be used in two ways. One of them is by connecting with different TCP port using HTTPS (443.port) instead of HTTP (80.port). The other one is connecting with same TCP port, however client could send a request to switch to TLS protection. (E.g. STARTTLS request for mail and news, in SMTP and NNTP, respectively) See [4].

The attacks arise from protocol logic could lead negotiating with weak algorithms. In other words, even if the client and the server use strong cryptographic primitives inside communication, the algorithm could be vulnerable.

One example is state machinery attacks and early CCS(ChangeCipherSpec) attack could be showed up as a protocol logic flaw in recent years. The other one is the flaws could be developed by an attacker inside TLS context or inside the certificate. The adversary could change the configuration files in order to close down the usage of

strong cryptographic primitives.

The renegotiation attack is the other example of protocol logic attacks. It was discovered in August 2009 and it affects all TLS versions and the last version of SSL. It relies on the plaintext injection attack. It allows an attacker to conjoin his request into the beginning of the conversation between the client and the server. This is not a kind of man-in-the-middle attack because the attacker could not be able to decrypt the message. This attack could cause some problems like injecting messages into HTTPS connection, downgrading HTTPS to HTTP, injecting custom responses, leading to denial of service attacks etc.

The Alert attack and the Triple Handshake attack are the recent ones included in the protocol logic and will be discussed in later.

4.1.2 Cryptographic Design Flaws

These attacks arise from the cryptanalysis on the blocks of the plaintext and the ciphertext used in algorithms inside SSL/TLS. The attacks also come to light due to the improper non-blackbox usage because the blackboxes are used to reach vulnerabilities in SSL and TLS. One of the example for these attacks is the usage of CBC(Cipher Block Chaining) Mode of operation in the block cipher encryption types. The recent attack for this kind of attack is named as BEAST attack.

Additionally, the padding oracle attacks are used for taking advantage of error messages to gain information on encrypted data. These attacks are included in both protocol logic and cryptographic design flaws and the attacks in [45] and [21] are related to that ones. POODLE attack, on the other hand, is also a padding oracle attack which aims CBC mode in SSL 3.0.

These attacks rely on some cryptographic failures in certificate design. Precautions for these attacks could somehow be possible, but implementing the precautions is really hard issue. The attacks towards TLS implementation are described in [19] and [31]. Therefore, even for modern ciphers based SSL/TLS protocols, the RSA based certificates still cause some weaknesses. For more information, see [10].

4.1.3 Implementation Flaws

These types of attacks could be emerged from the buffer overflows mentioned in [18], that are security vulnerabilities overrunning the buffer' s border and overwriting on close memory locations. OpenSSL attacks and timing-based side channel attacks towards cryptographic infrastructure are two types of buffer overflows. See [9] and [20] respectively.

4.1.4 Configuration and Infrastructure Flaws

Other types of attacks are related to the configuration and infrastructure flaws that are the problems in TLS forming. Certification management is an important issue and many mistakes comprise of the incorrect configuration of the certificates. For detailed information, see [35] and [26].

4.2 The Recent Attacks

4.2.1 The Alert Attack

This attack was discovered by the team of miTLS on February, 2012. Along the protocol in use, the client and the server authenticate each other in first handshake process messages by verifying the finished message. They also authenticate the application data by not accepting it. They just not authenticate in Alert protocol during the first handshake. The messages in Alert types are two bytes and they are cleavable. The process of attack is described in Figure 4.1. See [10].

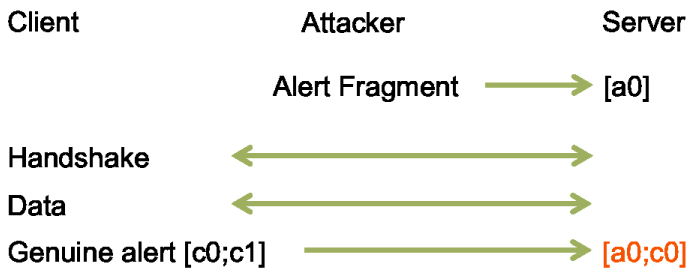

Figure 4.1: Alert Attack

As seen in the Figure 4.1, the attacker add [a0], which is one byte alert part and it is kept secret. When 2 bytes alert [c0,c1] is sent and authenticated, then the 2 bytes alert [a0,c0] is received and it works which breaks alert authentication process. Therefore, it is important to notice that there should be an agreement till the end of first handshake at least.

4.2.2 Timing Attacks

These types of attacks could be used to exploit the devices having weak computing power like smartcards. A timing attack targets to obtain the private information inside the protocol by looking at the time in the system. The attack told in [33] aims to obtain the keys for decrypting RSA algorithm. Many cryptographic libraries could not find a solution for timing attacks. One of the most widely used library in the SSL and TLS is OpenSSL. There is an optimization on the implementation of RSA thanks to the Chinese Remainder Theorem [22], Sliding Windows [32], Montgomery multiplication

[43] and Karatsuba Algorithm [20]. However, all of these optimizations produce timing attacks to get the private key of the system. Another attack for OpenSSL systems is ECC based timig attacks.

4.2.2.1 RSA based Timing Attacks

In OpenSSL Decryption System, the center of RSA decryption is exponentiation over a modulus, say N which is the multiple of two primes, say $p.q$. The exponentiation works as $m = c^d \mod N$ where m is the message, c is the ciphertext and d is the private decryption key. By the CRT, the message is obtained by calculating first $m_1 = c^{d_1} \mod p$, second $m_2 = c^{d_2} \mod q$ and last putting m_1 and m_2 together to get the message m. Note that, d_1 and d_2 are computed before by using d. At this point, the timing attack could be applicable to the factors of N. When the factors were found, it is easy to gain d because it is $e^{-1} \mod (p-1)(q-1)$ where e is the public encryption key of the system.

When we looked at the usage of attack in SSL, we see that the attack gets the private decryption key by observing the differences in time between the ClientKeyExchange and Alert Message. Even if a small difference in time occurs, it gives an idea about RSA parameters of the system.

4.2.2.2 ECC based Timing Attacks

The attack is mentioned in [20] on ECDSA based TLS protocols and there is a vulnerability inside OpenSSL library again. ECC uses scalar multiplications (e.g. point multiplication) that create problem about implementing these multiplications. This implementation is so called Montgomery power ladder [44]. This implementation could not prevent a timing side channel attack.

When we looked at the usage of attack in TLS, we see that the researchers calculated the time between the ClientHello message and the ServerKeyExchange message during the handshake protocol. The ServerKeyExchange message has an ECDSA signature over a digest, covering of necessary parts for producing further cryptographic materials. Since the digital signature could only be created randomly, an adversary is able to measure runtime of the vulnerable scalar multiplication function [28].

4.2.3 Triple Handshake Attack

This attack was presented in 2014 IEEE Symposium on Security and Privacy by a group members of IMDEA, INRIA and Microsoft Research. The attack also a kind of man-in-the-middle and allows an attacker to establish two connections with same keys and handshakes, insert an additional data inside the message and renegotiate to forward the connection between the client and the server.

The attack has 3 parts. Part 1 is shown in the Figure 4.2, part 2 is shown in the Figure 4.3 and part 3 is shown in the Figure 4.4.

In part 1, the client connects with the attacker unconsciously and the attacker connects with the server as if it is a client. The attacker also compel the client and the server to use RSA key exchange method. After that, the attacker in the middle takes encrypted message from client and encrypts it again and send it to the server. With same keys and parameters but different server certificates and finished messages, both handshakes to get a new session are completed.

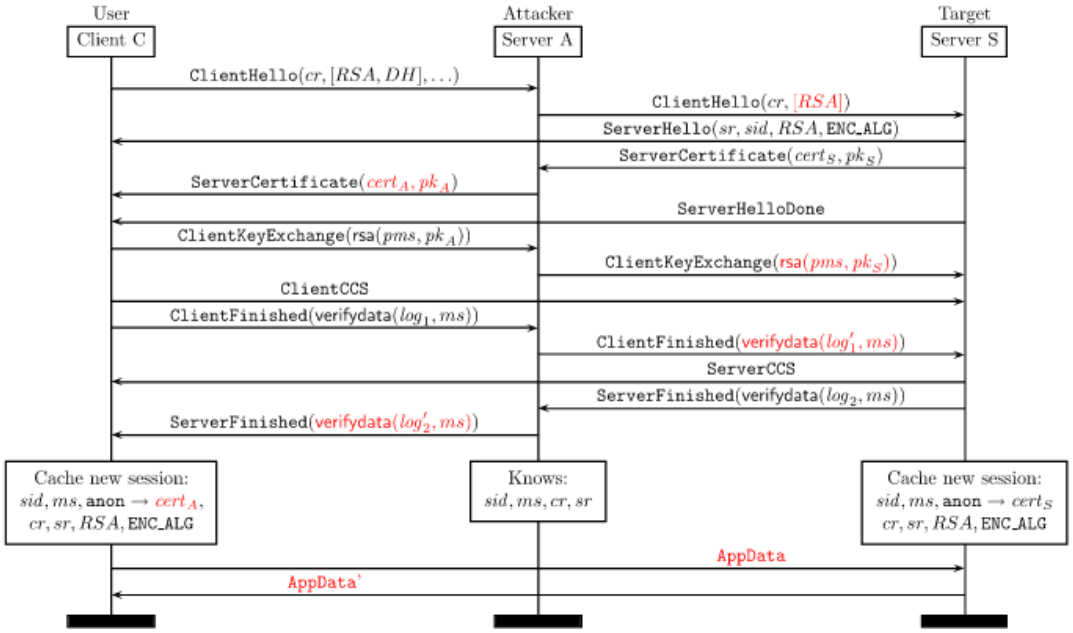

Figure 4.2: Triple Handshake Process 1

In part 2, the client again connects with the attacker and asks whether it could start the previous session or not. The attacker makes the same thing towards the server. Then, he sends the handshake messages that are unchanged between the client and the server. Now, both the keys and the finished messages are same. By the way, the attacker knows the connection keys now.

In part 3, the server wants to renegotiate with client authentication on the connection with the attacker. After that, the attacker sends the renegotiation request to the client and the client authenticates with its client certificate with the attacker. The attacker could send full messages from the client to the server and back. The handshake now succeeds, since the expected Renegotiation Indication extension values on connections are the same. See [10].

When renegotiation part finished, the attacker did not know the key. Also, he cannot read the transferred data on the connection. However, the primary messages of the attacker on both connections could be added to the messages after renegotiation part. See [16] for more information.

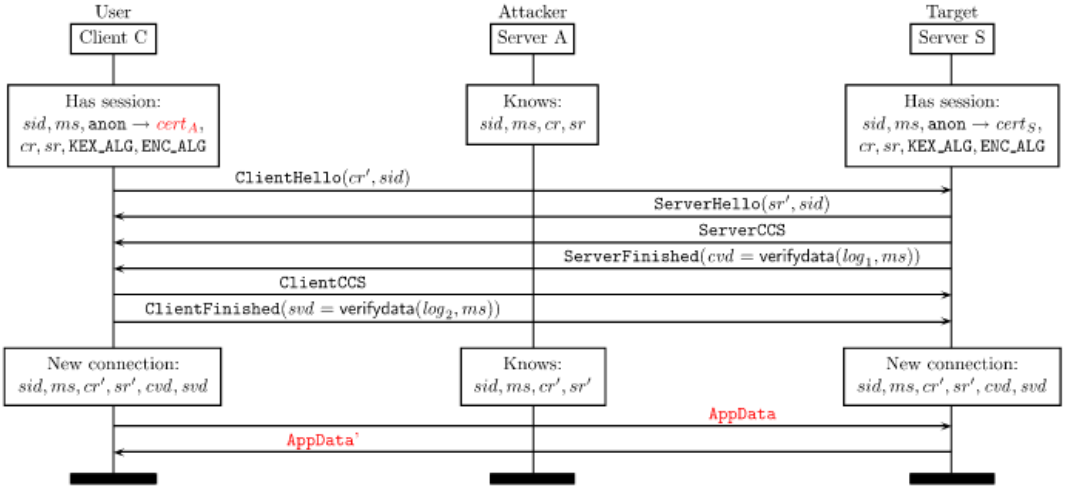

Figure 4.3: Triple Handshake Process 2

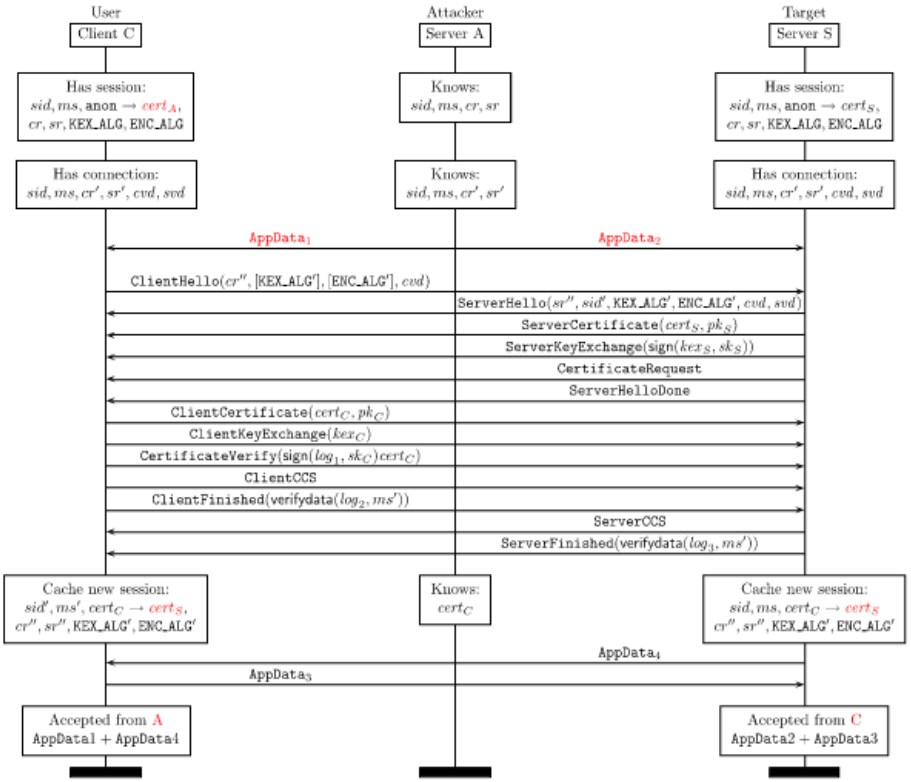

Figure 4.4: Triple Handshake Process 3

In order to avoid this attack, applying the same validation policy for all certificates received over a connection, binding the master secret to the full handshake and binding the abbreviated session resumption handshake to the original full handshake could be some solutions. See [16] in detailed explanations.

4.2.4 Cross Protocol Attacks

A cross protocol attack is a way of leading that the client uses one protocol in the connection whereas the server uses a different one. Adding something to one protocol could not be seen as an invalid data in another protocol. There are some popular cross protocol attacks which of them are mentioned below. For more information, see [38].

4.2.4.1 The Wagner and Schneier Attack

It is a kind of cross protocol attack that relies on the Diffie-Hellman key exchange digital signature does not include an identifier of the ciphersuite inside the protocol. The attacker convinces the client, who uses TLS RSA Export type ciphersuite and thinks different RSA parameters inside the ServerKeyExchange part, to take Diffie Hellman parameters from TLS Diffie Hellman RSA ciphersuite.

In the attack, a man-in-the-middle again tries to damage the connection. The client validates the signature and reads the RSA modulus that is a prime number of a Diffie-Hellman group. The client also reads the RSA exponent that is the generator of the corresponding group. He encrypts the key, say k with the generator, say g as k^g in modulus, say p. Then he adds this encrypted key into the ClientKeyExchange part. With knowing the generator g and as a prime number p, the attacker could reach the gth root of k^g to obtain the session key. See Figure 4.5.

For more information, see [46].

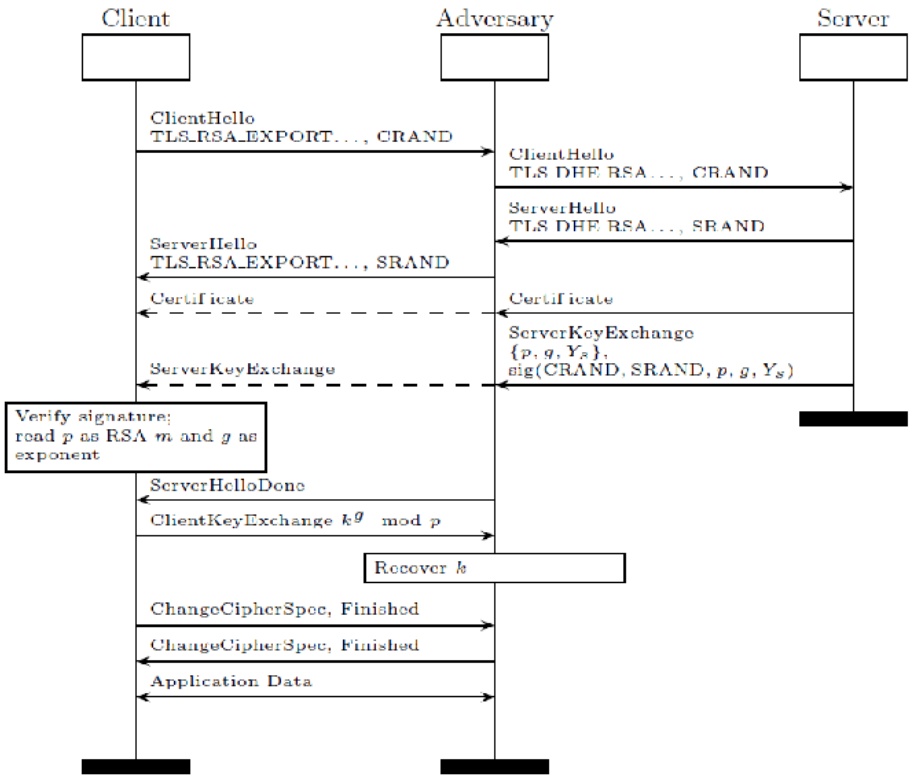

Figure 4.5: The Wagner and Schneier Attack

4.2.4.2 A New Cross Protocol Attack

It is a kind of server impersonation attack which uses Diffie Hellman Key Exchange protocol. In this attack, the server must support a prime curve and the client must support a plain Diffie Hellman process. That is because of the fact that the RSA signature algorithm could be used both in Elliptic Curve Diffie Hellman and Diffie Hellman Key Exchange.

Elliptic Curve Diffie Hellman(ECDH) ServerKeyExchange message should be a valid Diffie Hellman ServerKeyExchange message. The attacker in the middle could obtained the Diffie Hellman exchanged key. The attacks success probability is 2^{40}. One of the recommendation for not being exposed to this attack is changing the ServerKeyExchange signature, identifiers of the algorithms and all previous messages. For detailed informations and recommendations, see [38]. The process of the attack has shown in the Figure 4.6.

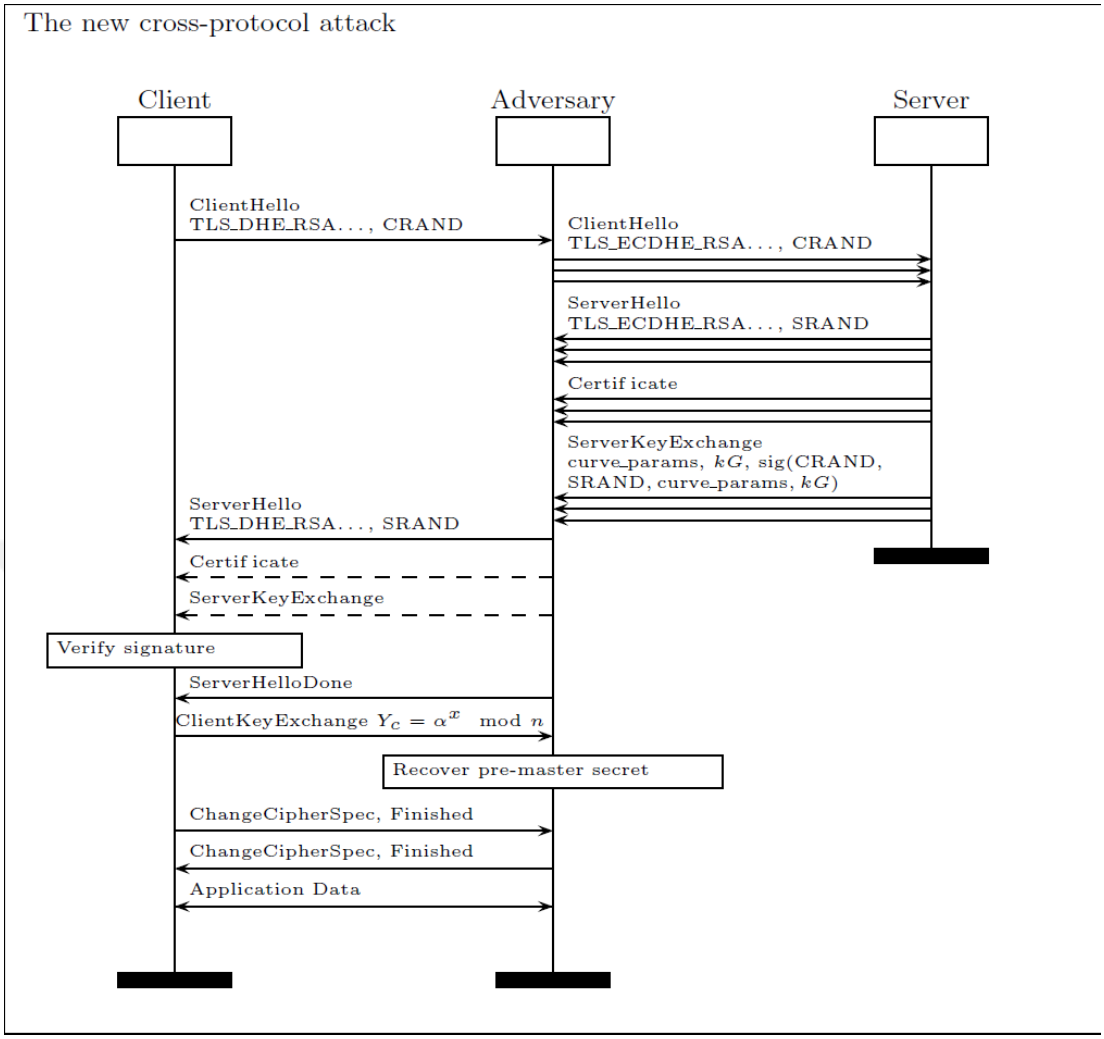

Figure 4.6: A New Cross-Protocol Attack

4.2.4.3 DROWN attack

This attack published in March 2016, very recently by some researchers. The name DROWN comes from Decrypting RSA with Obsolete and Weakened Encryption. It is an important and harmful vulnerability on HTTPS and other SSL/TLS based sessions. This attack allows an attacker to break the encryption and to obtain passwords or important numbers that have financial worth or stealing risky messages. Studies on this attack shows that around 33% of HTTPS websites are vulnerable to this attack. See [12].

A web server is vulnerable to DROWN attack if it uses SSL 2.0 that is used in a considerable amount of web servers right now. The problems of webserver has shown in Figure 4.7.

In order to prevent the attack, system developers should be sure that the private key for a server is not used for another server. The most practical solution may be upgrading

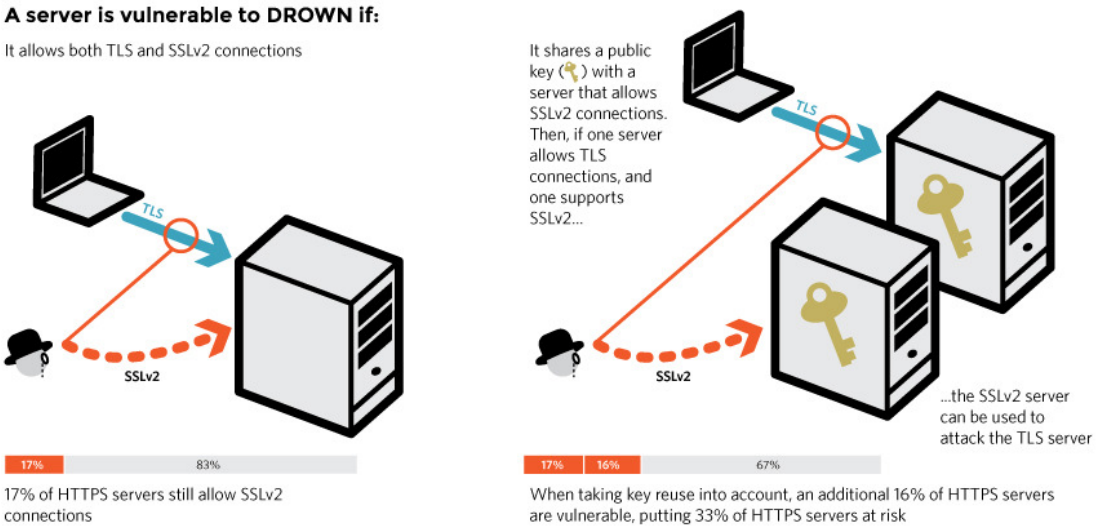

Figure 4.7: Drown Attack

TLS or SSL libraries and browsers.

4.2.5 SLOTH: Security Losses from Obsolete and Truncated Transcript Hashes

The collision of hash functions is a recently developed and highly damaging kind of exploit that brings a necessity of not using hash functions like MD5 and SHA1 in digital signature applications like SSL/TLS certificates. Nevertheless, these hash functions are still in use inside many cryptographic protocols. The Figure 4.1 shows the SLOTH attack results on MD5, SHA1 and HMAC, popular hash and hash based functions.

Table 4.1: SLOTH Attack Results

Protocol	Property	Mechanism	Attack	Collision Type	Precomputation	Work/connection	Wall-clock Time	Preimage Cost	Security Loss
TLS 1.2	Client Auth	RSA-MD5	Impersonation	Chosen Prefix		2^{39}	1 hour (48 cores)	2^{128}	89 bits
TLS 1.2	Channel Binding	Truncated HMAC (96 bits)	Credential Forwarding	Generic		2^{48}	20 days (4 GPUs)	2^{96}	48 bits
TLS 1.2	Client Auth	RSA-SHA1 or ECDSA-SHA1	Impersonation	Chosen Prefix		2^{77}		2^{160}	83 bits
TLS 1.2	Server Auth	RSA-MD5	Impersonation	Generic	2^{X} connections +storage	2^{128-X}		2^{128}	X bits
TLS 1.3	Server Auth	RSA-SHA1 or ECDSA-SHA1	Impersonation	Chosen Prefix		2^{77}		2^{160}	83 bits
TLS 1.1	Handshake Integrity	MD5 ! SHA1	Downgrade	Chosen Prefix		2^{77}		2^{160}	83 bits

The SLOTH attack works on TLS 1.2 client authentication process as a man-in-the-middle client impersonation attack and also on TLS Channel Bindings as a man-in-the-middle credential forwarding attack. Additionally, the attack affects on TLS 1.2 and 1.3 server authentication by removing MD5 from TLS 1.3 and affects softwares and responsible disclosures.

A man-in-the-middle client impersonation attack has shown in the Figure 4.8 and a man-in-the-middle credential forwarding attack has shown in the Figure 4.9. For detailed information, see [17].

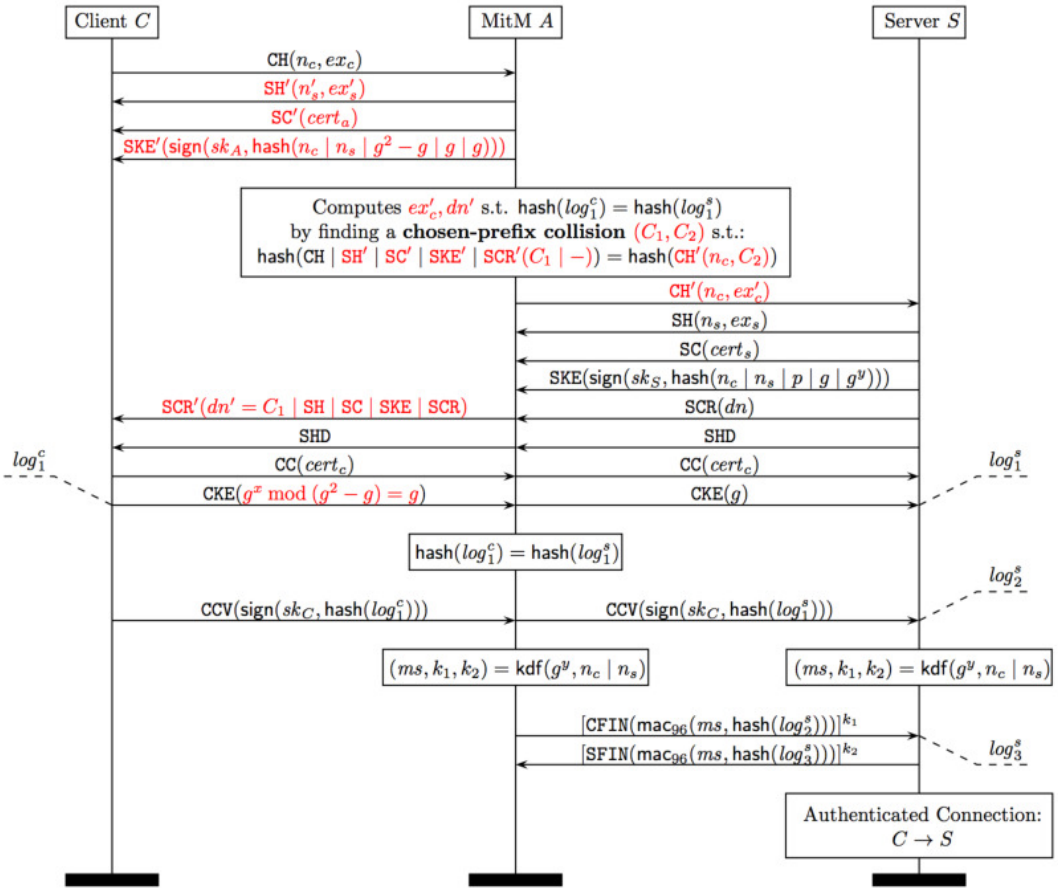

Figure 4.8: A man-in-the-middle Client Impersonation Attack

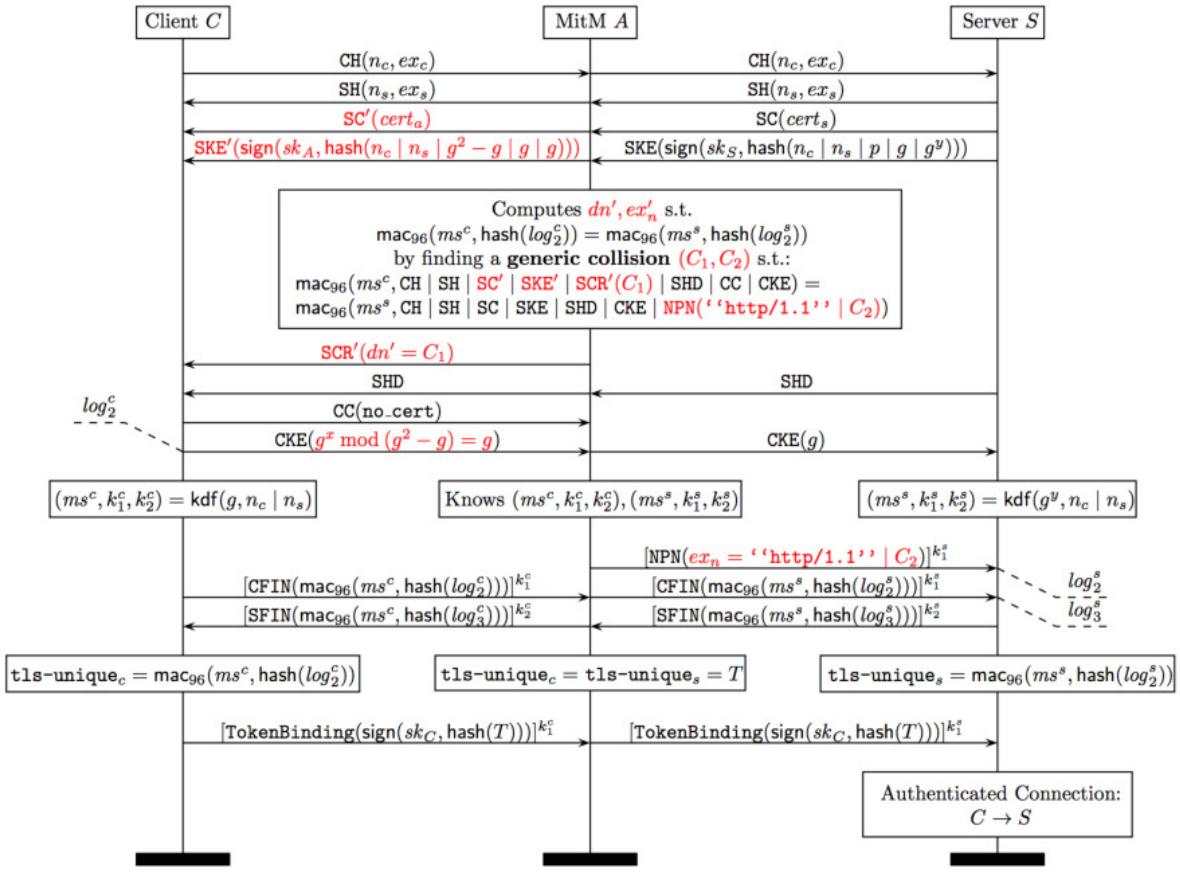

Figure 4.9: A man-in-the-middle Credential Forwarding Attack

4.2.6 SMACK: State Machine Attacks

In these types of attacks, the main approach is the designing flaws of state machines that use TLS protocol. The implementation mistakes in open source TLS machines were tested and realized that many vulnerabilities in these implementations have not been recognized for many years. In this section, we mention about these attacks rely on man-in-the-middle types.

4.2.6.1 SKIP-TLS: Message Skipping Attacks on TLS

This attack was found in 2015. Different cipher suites in TLS use different message sequences. As an example, in ephemeral (forward secret) Diffie-Hellman cipher suites (including ECDHE), server authentication relies on the Server Key Exchange message, whereas this message is not used in the RSA key exchange. The other one is, in non-ephemeral Diffie-Hellman cipher suites, clients use the DH keys fitted in server certificates instead of newly generated keys provided in the Server Key Exchange. See [23]. It was found that some open source TLS implementations (like OpenSSL, GnuTLS, CyaSSL, JSSE etc.) incorrectly let some messages be skipped even if they are required for the selected cipher suite.See [15]

Some TLS implementations were showed in the Table 4.2 according to how they are vulnerable.

Table 4.2: Available Vulnerabilities of TLS Implementations

JSSE(Java Secure Socket Extension): Server impersonation vulnerability. Java 1.5, 1.6, 1.7 and 1.8 prevent the attack.
CyaSSL(C language based SSL library): Client and server impersonation vulnerability. CyaSSL 3.3.0 prevents the attack.
OpenSSL(Open Source Implementation of SSL): Client impersonation vulnerability if server accepts static Diffie-Hellman certificates. OpenSSL 1.0.1k prevents the attack.
axTLS(Highly Configurable client/server TLS 1.2 Library): Client impersonation vulnerability. axTLS 1.5.2 prevents the attack.
Mono(Free Open Source Project): Client impersonation vulnerability. Mono 3.12.1 prevents the attack.

Note that, if you want to connect an HTTPS web site as a client software user with one of above libraries over a network that is not secure (e.g.WI FI), you are most probably not in secure. It is very important that you should use the latest version of the library to decrease the danger. There are some online testing servers for SKIP-TLS attack. See [1] .

4.2.6.2 FREAK Attack: Factoring RSA Export Keys

This attack was discovered on March, 2015 by some researchers from IMDEA, INRIA and Microsoft Research [15]. It causes server impersonation for which uses export-grade encryption in so many browsers and systems such as Apple' s Safari, Google' s Android phone systems, Microsoft Internet Explorer, OpenSSL, Microsoft' s SChannel(Secure Channel) implementation in Microsoft Windows. With this attack, HTTPS connection between the client and the server could be interrupted that allows an attacker to change the transferred data.

By using export RSA moduli less than or equal to 512-bits long, an attacker could factorize it in less than 12 hours with just $100 by using Number Field Sieve algorithm. However, many popular and important web sites including NSA, FBI, IBM, Symantec still enable export ciphersuites on their servers that could lead the FREAK attack directly. See [10].

Table 4.3: Vulnerable Systems on Freak Attack

Vulnerable TLS Client Libraries	Vulnerable Web Browsers
-IBM JSSE	-Cisco using OpenSSL
-Mono versions before 3.12.1	-Blackberry Browser
-LibReSSL versions before 2.1.2	-Android Browser
-Schannel versions before KB3046049	-Opera versions before 28
-SecureTransport versions before iOS 8.2, AppleTV 7.1, OS X Security Update 2015-002	-Safari on OS versions before March 9
-BoringSSL versions before Nov 10, 2014	-Internet Explorer on OS versions before March 9
-OpenSSL versions before 1.0.1k	-Chrome versions before 41

All the attacks in the Table 4.3 shows us that, a user of that systems should update their libraries or browsers constantly.

4.2.7 Imperfect Forward Secrecy: Focus on Logjam Attack

In the article [11], a popular algorithm Diffie-Hellman key exchange has been investigated in terms of security. This algorithm has a wide range of usage in SSH and IPsec systems and SSL/TLS protocols. Since the security of Diffie-Hellman key exchange mechanism relies on the security of Discrete Logarithm Problem(DLP) and it is believed that computing a discrete logarithm is much more difficult than factoring RSA modulus of the same size, this key exchange mechanism is considered as secure and hard to break. However, in this article, it has come to light that this thought is not working properly in practice because of two reasons: The first is, many servers use weak Diffie-Hellman parameters. The second is, design and implementation flaws and configuration mistakes leads many attacks on servers, which use Diffie-Hellman key exchange mechanism in protocols. From this point of view, the Logjam attack against

the TLS protocol and threats from state-level adversaries have been examined in this article. The Logjam attack, a security vulnerability against Diffie Hellman algorithm ranging from 512-bit to 1024-bit keys, allows the man-in-the-middle attack in order to downgrade vulnerable TLS connections to 512-bit export-grade crypto. This means that an attacker in the middle could be able to see and change the data inside the connection. DHE-EXPORT ciphers supported servers could be affected by this attack as much as all modern web browsers. The studies on this attack shows that this attacks applies to 8.4% of Alexa Top Million HTTPS sites and 3.4% of all HTTPS servers that have browser-trusted certificates. On the other hand, in terms of state-level adversaries, many HTTPS, SSH or VPN servers use the same prime numbers inside the Diffie-Hellman key exchange algorithm. Yet, this is a very important vulnerability to exploit the mechanism since this prime number is used into the Number Field Sieve algorithm, which is the most efficient process to break Diffie-Hellman connection. The cryptographers know that the Number Field Sieve algorithm needs an expensive precomputation step that generates a matrix values from the relevant prime number and a cheap descent step which uses this matrix, the generator g and g^a that are public parameters and the attacker could be easily compute the private "a" from this information. This means that an attacker that has a good precomputation power could easily calculate the discrete log of a number used in Diffie-Hellman key exchange and could modify the data transmitted in TLS protocol. The article shows us that the computation against the most common 512-bit prime used for TLS connection and the Logjam attack could be used to downgrade connections to 80% of TLS servers that support DHE-EXPORT. It is estimated that the same thing can be achieved for 768-bit and then 1024-bit primes. Breaking the system for the most common 1024-bit prime number used by web servers could allow passive eavesdropping in the 18% of the Alexa Top Million HTTPS domains. A second prime number could allow passive decryption of connections to 66% of VPN servers and 26% of SSH servers. A close reading of published NSA leaks says that their attacks on VPNs are consistent with having such a break.

This article recommends us some important key points for a secure connection. As a server owner part, we should disable support for export cipher suites and use a 2048-bit Diffie-Hellman group. As a browser user, we should update our browser to the latest version. As a system admin or developer, we should use TLS libraries of the latest version and the servers that we maintain use 2048-bit or larger prime numbers and the clients that we maintain refuse prime numbers that are smaller than 1024-bit in Diffie-Hellman key exchange protocol. As a result, to realize and fix these vulnerabilities, both cryptographers and system builders should work together. See [11]

4.3 Some Other Important Attacks

4.3.1 Generalization of attacks on SSL 2.0

This version of SSL is in the wrong in many aspects: -The cryptography used in SSL 2.0 contains identical keys for both encryption and message authentication processes.

-Its MAC (Message Authentication Code) construction is weak against some length extension attacks because it uses the hash function (MD5) with a secret prefix.

-There is no such protection for handshake protocol in SSL 2.0. This could cause man in the middle attacks.

-TCP connection close process is used to point to the end of the message. This could lead the truncation attacks.

-The virtual hosting system used in many web servers is contrary to SSL 2.0. This is because SSL 2.0 uses a single service and a fixed domain certificate. For this reason, many websites unable to use SSL 2.0.

4.3.2 Generalization of Attacks on SSL 3.0

In this version, such vulnerabilities of SSL 2.0 were corrected. However, there are also some weaknesses of SSL 3.0:

-The key derivation process is weak in this version. This is because of the fact that the half of the master key is produced by depending on MD5 hash function which could cause some collisions.

-CBC mode of operation system used in SSL 3.0 is vulnerable to the padding attack (POODLE attack) so this system is critical for SSL connection.

4.3.3 Poodle Attack

This attack was discovered on October 14, 2014 by Bodo Möller, Thai Duong and Krzysztof Kotowicz from Google Security Team. The name Poodle means Padding Oracle on Downgraded Legacy Encryption and it is a kind of man-in-the-middle attack targeting CBC-mode ciphers in SSL 3.0. This attack allows a man-in-the-middle to decrypt the message in this version of SSL. Even if the TLS versions become widespread, there are still so many browsers and servers using SSL 3.0 and vulnerable to this attack.

In this attack system, if an attacker could reach and control a router in an open public area like Wi-Fi, then he could enforce the user' s browser to downgrade to use SSL 3.0 or before versions even if the browser uses any TLS version. After that, the attacker could capture the information transferred by SSL certificate.

As a solution, both the client and the server could support TLS versions, never SSL, but they should not support downgrading the system. One way of doing this is upgrading their browsers or servers to the latest version. See [42] for more information.

The statistics shows us that the POODLE attack is a really serious problem. As of October 12, 2014, nearly all HTTPS Alexa Top 1 Million websites supports SSL 3.0 and 0.12% of them do not support TLS. More statistics are in [2].

4.3.4 Truncation Attack

This attack gives an impression as if the message is finished by inserting a TCP code into the message. In this way, the server could not take the rest of the message. This TCP code is a TCP FIN message which is unencrypted and causes to close the connection.

4.3.5 Protocol Downgrade Attacks

This attack makes the server think that it communicates with the current version of TLS but in reality it uses one of the previous versions. In this way, the system used between the client and the server is accessible to some known vulnerabilities of previous versions. For example, the attacker could use weaker encryption algorithms or smaller key size.

4.3.6 Beast Attack

This attack was discovered on September 2011 and named as Browser Exploit Against SSL/TLS(BEAST). This affects Cipher Block Chaining(CBC) method used in SSL. This is also a kind of man-in-the-middle attack and it allows an attacker silently decrypt and obtain authentication tokens to get the transferred data. For detailed information, see [24].

4.3.7 Crime and Breach Attacks

The Crime attack was announced in 2012 as a kind of side channel attack against compression in HTTPS implementations. Breach attack named as Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext, on the other hand, is a kind of Crime attack on HTTP compression and HTTP responses. It was announced in 2013 on Black Hat conference. For detailed information, see [25] and [27].

CHAPTER 5

IMPORTANT POINTS AND SELECTION OF PARAMETERS

5.1 Recommended Key Sizes and Parameters in Algorithms and Protocols

A very first and important point for selecting a suitable key size in an algorithm or a protocol is that the key size selected by a user should not be broken easily with the techniques or computation powers of the present time. Predictions for the security of keys should not just include the current time. They should also be foresighted ideas and should be advised in the near or long terms. Note that, a suitable key size, say k-bits does not guarantee the k-bits of security. Note also that selecting a long key size does not mean to make algorithms or systems more secure.

Additionally, the recommended cryptographic algorithms and cryptography based systems are at least AES-128 for block ciphers, at least SHA-256 for hash functions and at least 256-bit-Elliptic Curves for public key primitives. In the Table 5.1, the recommended key sizes of the report published in 2014 by ENISA were given.

For key agreement systems or inside the protocols, generally it is better to use discrete logarithm based systems and primitives, integer factorization based algorithms, strong cryptographic components or functions. Discrete Logarithm Problem(DLP) and Integer Factorization Problem are still very hard problems at the present time.

The key agreement systems in SSL/TLS have two main types: RSA-Key Exchange and Diffie-Hellman Key Exchange. Even if RSA is not considered secure enough nowadays, it is a widely used mechanism in SSL/TLS protocols and the attack towards

Table 5.1: Recommended Key Sizes by ENISA

	Parameter	Legacy	Future System Use Near Term	Future System Use Long Term
Symmetric Key Size	k	80	128	256
Hash Function Output Size	m	160	256	512
MAC Output Size	m	80	128	256*
RSA Problem	$l(n) \geq$	1024	3072	15360
Finite Field DLP	$l(p^n) \geq$	1024	3072	15360
	$l(p), l(q) \geq$	160	256	512
ECDLP	$l(q) \geq$	160	256	512
Pairing	$l(p^{k.n}) \geq$	1024	3072	15360
	$l(p), l(q) \geq$	160	256	512

RSA in [19] has been fixed in the latest versions of TLS. Also, a security analysis towards RSA in TLS has been published in [34]. Inside the key agreement step, the key for transferring data is obtained from pre-master secret inside the protocol. This derivation is generated in two ways: either TLS 1.2 usage or the previous versions of TLS usage.

The encrypted data transferred in the TLS are sent and received between the client and the server inside the Record Layer of the protocol. Two systems widely used in TLS are MAC-then-Encode-then-Encrypt using CBC mode or MAC-then-Encrypt using RC4 algorithm. The former one is hard to implement and the latter one is a weak stream cipher type. However, these problems arrived at a solution in TLS 1.2 thanks to the Authenticated Encryption, the GCM mode and CCM mode. Other attacks for this situation is BEAST [24] and CRIME [25] attacks. The list of TLS ciphersuites are in [8].

For the handshake protocol, RSA is also a widely used key agreement protocol in TLS. Even if there are many attacks towards RSA algorithm, the protocol could be made more secure by the usage of strong number theoretic parameters and random oracles(theoretical black boxes).

For the whole protocol security, it is better to choose discrete logarithm based or elliptic curve based systems. Especially, ECDSA is strongly recommended but not in a common usage.

The recommended ciphersuites by ENISA for TLS were given in the Table 5.2.

Table 5.2: Recommended Ciphersuites for TLS

Camellia_128_GCM_SHA256
AES_128_GCM_SHA256
Camellia_256_GCM_SHA384
AES_256_GCM_SHA384
AES_128_CCM
AES_128_CCM_8
AES_256_CCM
AES_256_CCM_8

For more information, see the ”Algorithms, key size and parameters” report-2014 by ENISA.

5.2 Protection Methods from Recent Attacks and Some Other Known Attacks

Many attacks towards SSL/TLS could be fixed by taking in consideration much more to the design of the protocol, implementation flaws, vulnerabilities and cryptographic infrastructures if possible. The main way for not being exposed to these attacks is upgrading the protocol, the browser and the server to the latest version. Also, the informations inside the content of the messages should be given clearly in order not to come across misunderstandings.

The Wagner and Schneier Attack [46] arrived at a solution by authenticating the data in the handshake protocol and adding the hash values of them into the evaluations of ClientFinished and ServerFinished messages. The problem here is that the hash values were not added to the ChangeCipherSpec and Alert messages which could lead to the following attacks. This attack shows that satisfying authentication between the client and the server is really important issue. For detailed information, see [14].

As a cryptographic design flaw, [19] is an important attack because it gets benefits from the error messages coming from the server to obtain information about the plaintext messages. This means that there should be taken less error messages as much as possible for not being a target of an attacker. The attack [31] is an improved version of [19]. These studies showed that precautions towards some attacks could lead to find other problems for exploiting the system. [13] attack moved these previous attacks to further away.

Another cryptographic design flaw is the attack mentioned in [45]. This attack could not been applied to the classical SSL and TLS protocols. Yet, it is applicable to the DTLS(Datagram Transport Layer Security) protocol that provides secure communication for datagram based protocols. As a solution, the use of GCM mode instead of CBC mode could be given. Equal error messages for padding and decrypting could be another solution but they could not prevent timing attacks completely.

Performance optimization attack informed by [30] shows that this optimization is not always beneficial, even it could be harmful. Therefore this fixing could be skipped or rearranged.

As a cross protocol attack type, the key exchange algorithm confusion could work on Elliptic Curve Diffie Hellman mechanisms [38]. With today' s computation power, the attack does not affect too much, but it will be a strong threat in the future as this power increases.

A way of preventing timing attacks is being careful about implementations. They should have equal time responses inside the process of the protocol. Also, timing attacks could arise from the side channels coming from unforeseen places. Additionally, instead of using changeable algorithms for comparing the time to find the secret, using constant time algorithms could be a better choice. The optimizations for the implementations like Montgomery multiplication or Karatsuba algorithm are also remarkable issues for designing the secure systems. For more and detailed information, see [41].

CHAPTER 6

CONCLUSION

Communicating with a computer based systems always reveal the importance of security, especially with regards to the criticality and privacy points. The standardized certificate based protocols like SSL and TLS help such systems to make more secure and to decrease the exploitations. In addition, since these protocols are based on the cryptography, they could benefit somehow from the confidentiality, data integrity, authentication and non-repudiation features of cryptographic systems. Nevertheless, vulnerabilities like the design and implementation flaws of the systems or inability of cryptographic infrastructure of the protocols make this structure become a target of adversaries. Hence, investigating these protocols in detail is a necessary and indispensable issue.

In this thesis, firstly, significant cryptographic algorithms, functions and key exchange mechanisms were examined. More particularly, they were chosen from the most widely used ones inside SSL and TLS. Secondly, the development process of these protocols and what added on them that changes their names consecutively were given from past to present. Additionally, the public key infrastructure, which is the cornerstones of the SSL and TLS were told. After that, the working principle of the protocols was mentioned in summary. Thirdly, the attack types according to their vulnerable points were examined and it was decided that they have four branches: Protocol Logic Flaws, Cryptographic Design Flaws, Implementation Flaws and Configuration and Infrastructure Flaws. At this point, we particularly focused on the recent attacks on SSL and TLS in order to pay attention to the present time disclosures. The Alert Attack, Timing Attacks, Triple Handshake Attack, Cross Protocol Attacks, SLOTH Attack, Smack Attack and Logjam Attack are the most important current attacks. Also, we explained some other important attacks in brief not to ignore them for security. Lastly, we mentioned the important points for designing a prospering protocol and the parameters for selecting them carefully. For this reason, the currently recommended key sizes and parameters in the algorithms and protocols used by SSL and TLS were given. At the end, protection methods from well-known attacks were told.

6.1 Our Suggestions for Future Works

Table 6.1 shows the usage of SSL certificate authorities for websites from w3techs statistics. This table means that over the half of websites do not use a certificate. This is a main and critical problem, especially for users since they connect to a page having no such security measure. In particular, people could give their credit card numbers or critical personal informations uncaringly, leading the loss of money or more crucial things. Therefore, we suggest that it is important to use websites with certificate ones (generally they begin with "https") as far as possible.

Other issue is a remarkable amount of websites use invalid domain (45.4% from all). This also invites attackers to the system and may be never being noticed. So, we suggest also that especially the system developers should be careful about the domain of the certificate.

Certificate expired is another problem for the websites. Nearly 1% of web sites use expired certificates that causes a thought as if a certificate is still active. The percentage of this problem could seem too small, but if we think on the basis of whole websites that are over millions nowadays, it is a serious number.

Finally, there is a gap between the study areas of system developers and cryptographers. Many attacks arise from either as a design flaw of a system to the hardware or software, or as a lack of cryptography inside the protocol. Therefore, we recommend that both system developers and cryptographers should work together or they should learn both of these issues best. Also, they should pay attention to performed attacks and should design a system by avoiding from the attacks as much as possible.

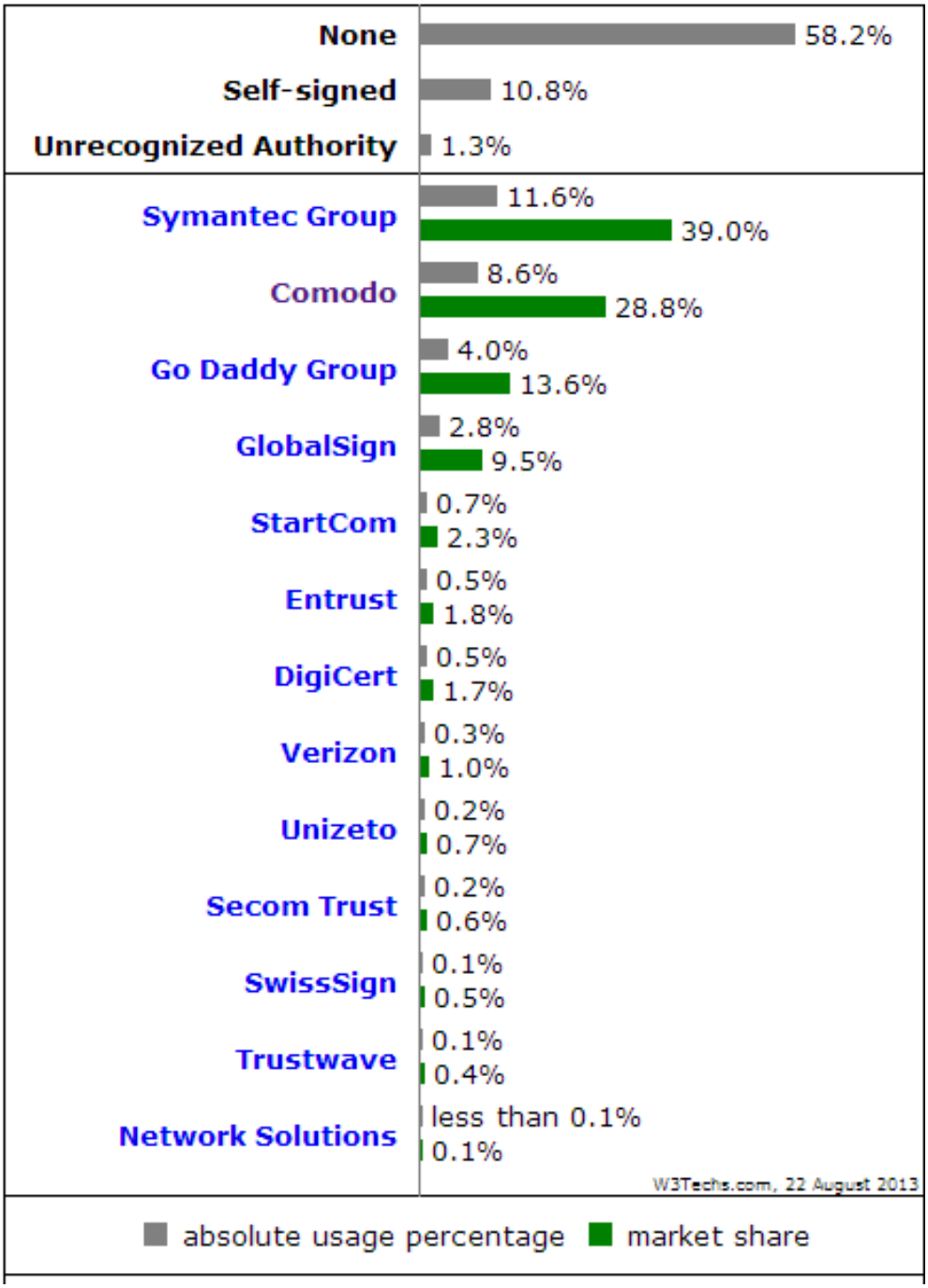

Figure 6.1: Usage Percentages of Certificates

REFERENCES

[1] 12 Online Free Tools to Scan Website Security Vulnerabilities and Malware, https://geekflare.com/online-scan-website-security-vulnerabilities/, accessed: 2016-08-16.

[2] Alexa Top 1 Million Domains, https://poodle.io/, accessed: 2016-08-16.

[3] Diffie-hellman in SSL/TLS, https://wiki.openssl.org/index.php/Diffie_Hellman, accessed: 2016-08-16.

[4] History of the SSL/TLS Protocol Suite, http://cromwell-intl.com/cybersecurity/ssl-tls.html, accessed: 2016-08-16.

[5] Internet Protocol (IP), https://www.techopedia.com/definition/5366/internet-protocol-ip, accessed: 2016-08-16.

[6] Introduction to Secure Sockets Layer, http://euro.ecom.cmu.edu/resources/elibrary/epay/SSL.pdf, accessed: 2016-08-16.

[7] Transport Layer Security over Stream Control Transmission Protocol, https://tools.ietf.org/html/rfc3436, accessed: 2016-08-20.

[8] Transport Layer Security (TLS) Parameters, http://www.iana.org/assignments/tls-parameters/tls-parameters.xml, accessed: 2016-08-16.

[9] Vulnerabilities, https://www.openssl.org/news/vulnerabilities.html, accessed: 2016-08-16.

[10] A Zoo of TLS attacks, https://mitls.org/pages/attacks#deployment, accessed: 2016-08-16.

[11] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al., Imperfect forward secrecy: How diffie-hellman fails in practice, in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 5–17, ACM, 2015.

[12] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, et al., Drown: Breaking TLS using SSLv2.

[13] R. Bardou, R. Focardi, Y. Kawamoto, L. Simionato, G. Steel, and J.-K. Tsay, Efficient padding oracle attacks on cryptographic hardware, in *Advances in Cryptology–CRYPTO 2012*, pp. 608–625, Springer, 2012.

[14] M. Bellare and P. Rogaway, Entity authentication and key distribution, in *Annual International Cryptology Conference*, pp. 232–249, Springer, 1993.

[15] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue, A messy state of the union: Taming the composite state machines of TLS, in *2015 IEEE Symposium on Security and Privacy*, pp. 535–552, IEEE, 2015.

[16] K. Bhargavan, A. D. Lavaud, C. Fournet, A. Pironti, and P. Y. Strub, Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS, in *2014 IEEE Symposium on Security and Privacy*, pp. 98–113, IEEE, 2014.

[17] K. Bhargavan and G. Leurent, Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH, NDSS (Feb. 2016), 2016.

[18] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright, Transport Layer Security (TLS) extensions, Technical report, 2006.

[19] D. Bleichenbacher, Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1, in *Annual International Cryptology Conference*, pp. 1–12, Springer, 1998.

[20] D. Brumley and D. Boneh, Remote timing attacks are practical, Computer Networks, 48(5), pp. 701–716, 2005.

[21] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux, Password interception in a SSL/TLS channel, in *Annual International Cryptology Conference*, pp. 583–599, Springer, 2003.

[22] K. Chemali, Chinese Remainder Theorem, 2005.

[23] J. Ćurguz, VULNERABILITIES OF THE SSL/TLS PROTOCOL.

[24] T. Duong and J. Rizzo, Here come the $\oplus$ ninjas, Unpublished manuscript, 320, 2011.

[25] T. Duong and J. Rizzo, The crime attack, in *Presentation at ekoparty Security Conference*, 2012.

[26] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, The most dangerous code in the world: validating SSL certificates in non-browser software, in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 38–49, ACM, 2012.

[27] Y. Gluck, N. Harris, and A. Prado, BREACH: Reviving the CRIME attack, Unpublished manuscript, 2013.

[28] N. Howgrave-Graham and N. P. Smart, Lattice attacks on digital signature schemes, Designs, Codes and Cryptography, 23(3), pp. 283–290, 2001.

[29] A. Kahate, *Cryptography and network security*, Tata McGraw-Hill Education, 2013.

[30] J. Kelsey, Compression and information leakage of plaintext, in *International Workshop on Fast Software Encryption*, pp. 263–276, Springer, 2002.

[31] V. Klima, O. Pokornỳ, and T. Rosa, Attacking RSA-based sessions in SSL/TLS, in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 426–440, Springer, 2003.

[32] C. K. Koc, High-speed RSA implementation, Technical report, Technical Report, RSA Laboratories, 1994.

[33] P. C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in *Annual International Cryptology Conference*, pp. 104–113, Springer, 1996.

[34] H. Krawczyk, K. G. Paterson, and H. Wee, On the security of the TLS protocol: A systematic analysis, in *Advances in Cryptology–CRYPTO 2013*, pp. 429–448, Springer, 2013.

[35] J. Lawall, B. Laurie, R. R. Hansen, N. Palix, and G. Muller, Finding error handling bugs in openssl using coccinelle, in *Dependable Computing Conference (EDCC), 2010 European*, pp. 191–196, IEEE, 2010.

[36] D. H. Lee and X. Wang, *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011, Proceedings*, volume 7073, Springer Science & Business Media, 2011.

[37] A. Mahalanobis, *Diffie-Hellman Key Exchange Protocol, Its Generalization and Nilpotent Groups.*, Ph.D. thesis, Florida Atlantic University Boca Raton, Florida, 2005.

[38] N. Mavrogiannopoulos, F. Vercauteren, V. Velichkov, and B. Preneel, A cross-protocol attack on the TLS protocol, in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 62–72, ACM, 2012.

[39] H. L. McKinley, SSL and TLS: A beginners guide, SANS Institute, 2003.

[40] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, CRC press, 1996.

[41] C. Meyer and J. Schwenk, Lessons Learned From Previous SSL/TLS Attacks-A Brief Chronology of Attacks and Weaknesses., IACR Cryptology ePrint Archive, 2013, p. 49, 2013.

[42] B. Möller, T. Duong, and K. Kotowicz, This POODLE bites: exploiting the SSL 3.0 fallback, PDF online, 2014.

[43] P. L. Montgomery, Modular multiplication without trial division, Mathematics of computation, 44(170), pp. 519–521, 1985.

[44] P. L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, Mathematics of computation, 48(177), pp. 243–264, 1987.

[45] S. Vaudenay, Security Flaws Induced by CBC Padding—Applications to SSL, IPSEC, WTLS..., in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 534–545, Springer, 2002.

[46] D. Wagner, B. Schneier, et al., Analysis of the SSL 3.0 protocol, in *The Second USENIX Workshop on Electronic Commerce Proceedings*, volume 1, pp. 29–40, 1996.