

Evgeny Malygin

INVESTIGATION OF DIGITAL CERTIFICATES

Creation of self-signed certificate on Windows 8

Bachelor's Thesis
Information Technology

May 2014



DESCRIPTION

		Date of the bachelor's thesis 27.05.2014
Author(s) Evgeny Malygin	Degree programme and option Information Technology	
Name of the bachelor's thesis Investigation of digital certificates Creation of Self-signed certificate on Windows 8		
Abstract The purpose of this study was to create a free of charge self-signed certificate for a local domain. Such certificate can be used for analysing and testing a web server, checking operability of a system with SSL certificate, etc. The use of such certificate gives an idea about all challenges of working with HTTPS protocol. The practical part of this document includes all phases of creating a self-signed certificate on a local web server. It starts from installation webserver itself, DNS server and configuration of all parameters. The next step is creation of self-signed certificate on the server using OpenSSL and applying it to the web site. The last phase of the practical part is testing and analysis of the certificate in different browsers. The tests were analysed with Wireshark, as software for capturing packets during the transmission. The results of the project have shown that a self-signed certificate encrypts the transmitting data using HTTPS protocol. Despite the fact that certificate works as a normal certificate, issued by Trusted Certificate Authority, and vulnerabilities were not found, it has limited implementation. First of all it should not be used for web services with critical data (banks, on-line stores, money exchange, etc.), because it does not provide identity of the server and visitors could easily become a victim of a man-in-the-middle attack. The material contained in this project can be successfully used for education purposes, practical usage of SSL certificates and for web development.		
Subject headings, (keywords) HTTPS, SSL, TLS, OpenSSL, self-signed certificate, digital certificate, Apache, browsers		
Pages 51	Language English	URN
Remarks, notes on appendices		
Tutor Matti Koivisto	Employer of the bachelor's thesis Mikkeli University of Applied Sciences	

CONTENTS

1	INTRODUCTION	1
2	DIGITAL CERTIFICATE	2
2.1	Authentication, integrity, confidentiality	3
2.1.1	Symmetric cryptography	5
2.1.2	Public key infrastructure and asymmetric cryptography	5
2.1.3	Hashing	10
2.2	Alternative CA types	12
2.3	Self-signed certificate	13
3	HTTP(s).....	14
3.1	HyperText Transfer Protocol	14
3.1.1	Security issues of HTTP	18
3.2	HTTPS (HTTP Secure).....	19
4	DNS – DOMAIN NAME SYSTEM	20
4.1	DNS queries	22
5	SELF-SIGNED CERTIFICATE CREATION	24
5.1	Environment and topology.....	24
5.2	Certificate creation and DNS installation	26
5.3	Testing with Firefox, Chrome, Opera and IE	29
5.3.1	Mozilla Firefox	29
5.3.2	Google Chrome	30

5.3.3	Opera	30
5.3.4	Internet Explorer.....	31
5.3.5	Trusted Root Certification Authorities	31
6	CONCLUSION	36
	BIBLIOGRAPHY	37
	APPENDIX	

1 INTRODUCTION

In the modern computer world, the most attention is directed to security. Antivirus programs, utilities, firewalls; specialists design all this software to protect personal computers, laptops, mobile phones or other digital devices. However, how to protect a web page? How to prove that this website can be trusted? Where is a warrant that intruders do not try to steal users' private information (passwords, bank information, etc.) through fake sites that looks as a real one? Moreover, online purchases are quite popular today, but not every user thinks about security of the transactions. Time flies and every second counts, about this property by everyone tries to save time and does not pay attention to minor changes on the usual web pages. To be sure, a web page is secure and belongs to the safe source; digital certificates, SSL connection and HTTPS protocol are used.

This work is a part of larger research security project "Investigation of Digital Certificates" that were develop together with Evgeniia Gromyko. The aim of this thesis is to create a digital certificate for a local web site, and aim of thesis of my colleague is to verify of reliability of digital certificates.

My thesis consists of two parts: theoretical and practical. The aim of the theoretical part is to research different types of certificates and ideas how to prove the actuality of the sources. This part includes describing what the Digital Certificate and SSL are, what is the different between HTTP and HTTPS, how to create own Certificate Authorities and get Digital Certificate. Digital Certificates are covered in Chapter 2 of this thesis. HyperText Transfer Protocol Secure (HTTPS) protects communication over a computer network. Chapter 3 explains most important points of this protocol in. For working with web services, the knowledge about DNS server is also significant (Chapter 4).

How to create HTTPS instead of HTTP for web server is the main goal of this thesis and practical part. Chapter 5 tells about creation of the Certificate without intermediaries. The name of this certificate is "self-signed certificate". Self-signed certificate is unacceptable for organizations which work with private users' data, money transactions (banks, on-line stores), but can be useful for websites which are necessary to encrypt login and password only. For the practical part, three virtual machines are used. They are connected together in one local area network (LAN). One

of them is a server, the second one is a DNS server and the third one is a Client. The server used OpenSSL software to create digital certificate and server Apache for a local website.

The results of this research and analysing are discussed in the final chapter.

2 DIGITAL CERTIFICATE

Business Dictionary (2014) says that passport is a “document issued by a government to allow its citizens to travel abroad, and request other governments to facilitate their passage and provide protection, on a reciprocal basis.”

One of the earliest known references to paperwork that served in a role similar to a passport was founded in the Old Testament dating from approximately 443 BC. Appearance of this document has changed over the years not only by influenced by different cultures, but also from different regions. Nevertheless, main role has never changed – identity. A passport has a unique key (passport number), and some attributes (an expiration date, name of the holder, address, photo, etc.). It is issued by a trusted agency and protected from being tampered with. The validity of such document is easy to check.

Some security services require a way to authenticate users or protect data in transmit. Authentication and confidentiality can be implemented in several ways, starting from simple password exchanges to elaborate hardware-assisted security systems. One way is to use digital certificates, since they can provide both authentication and confidentiality. (Adams 2001, 611).

As a passport, digital certificate provides identifying information. It contains a copy of the certificate holder's public key and several attributes, such as the name of the certificate holder, a serial number, expiration dates, and the digital signature of the certificate-issuing authority (CA) so that a recipient can verify that the certificate is real. (Rouse 2013). (Figure 1.)

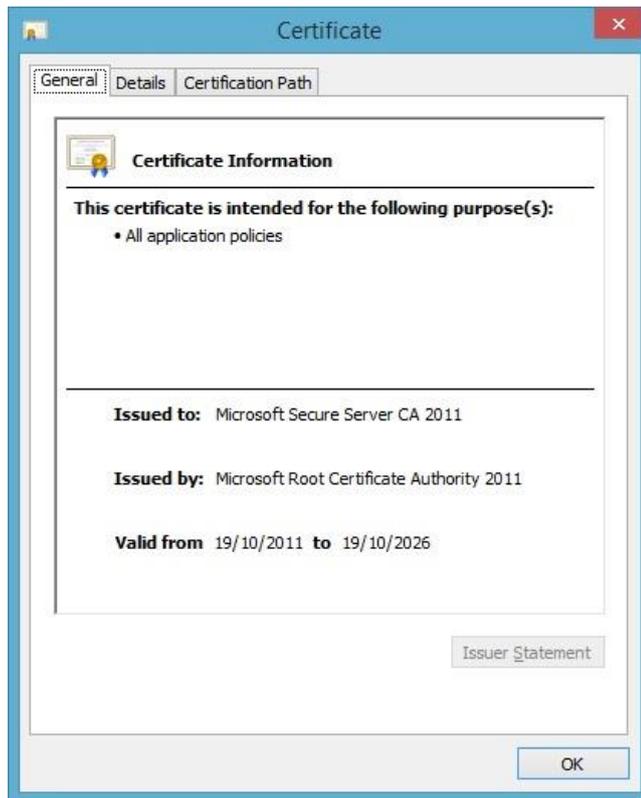


FIGURE 1. The certificate

Accuracy of this information is verified and confirmed by official, trusted agency such as VeriSign, Inc., Thawte Consulting, Symantec, Comodo Group, etc. Generally, web browsers and operation systems comprise list of trusted CA root certificates.

2.1 Authentication, integrity, confidentiality

Nowadays the amount of information, which is flowing over the Internet, is increasing every minute. Eric Schmidt, the CEO of Google, estimated the size at roughly 5 million terabytes of data (wiseGEEK 2014). According to Evans (2013), by 2020 there will be 50 billion things connected to the Internet (Figure 2).

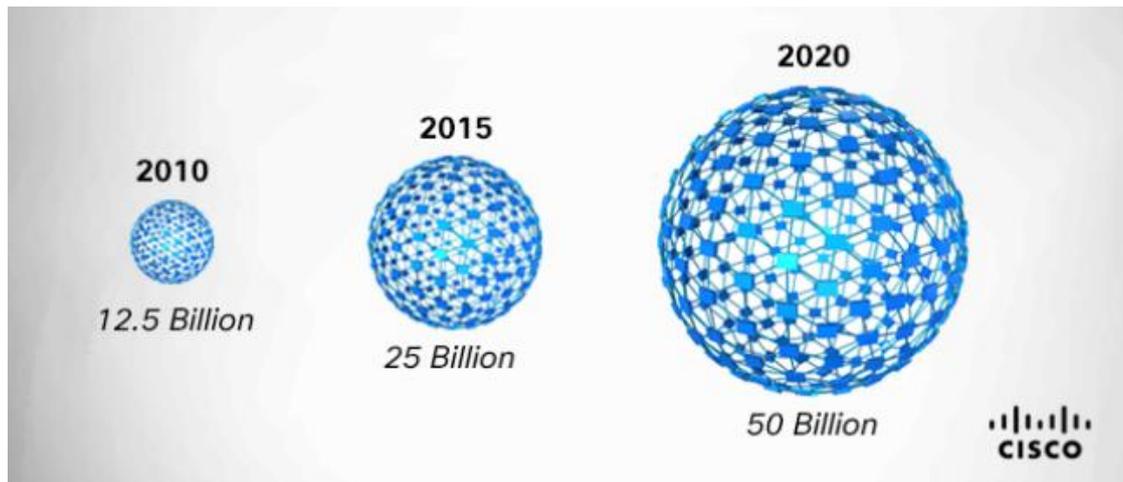


FIGURE 2. Amount of things connected to the Internet (canadablog.cisco.com 2013)

These things are not just some electronic devices such as smartphones and tablets, it can be even a cattle or wireless cardiac monitor. However, even such things need to be protected. To secure this communication a few primary tasks should be used: authentication, integrity and confidentiality.

With *authentication*, the user can be sure that a message comes from the source that it claims to come from. In private and public computer networks authentication is commonly done through the use of logon passwords or personal information number (PIN). This service allows to user to be identified. The password or PIN can be assigned or self-declared. The weakness in this system for transactions that are significant (such as the exchange of money) is that passwords can often be stolen, accidentally revealed, or forgotten.

Integrity ensures that data can only be accessed or modified by those authorized to do so. A receiver can be sure that information is not altered or misrepresent. As example of integrity can be presented a wax seal. An unbroken seal on an envelope guaranteed the integrity of its contents.

Data passes between client and server through one or more intermediary devices. It may also be kept in some repositories. Moreover, some data can contain sensitive information. There is a risk that intruder can get access to that sensitive data. For that reasons data *confidentiality* is used. Data confidentiality ensures privacy so that only

the receiver can read the message. To gain the confidentiality is possible in two ways. The first one is encryption.

Encryption uses a key to encrypt a plaintext to the ciphertext, and decrypt it to the plain text. There are two types of cryptography to provide data confidentiality: symmetric and asymmetric.

The second way is using a hash function.

2.1.1 Symmetric cryptography

With symmetric cryptography, both the sender and receiver use one key. This key is used for encryption from one side and decryption from another (Figure 3) (Microsoft 2005a).

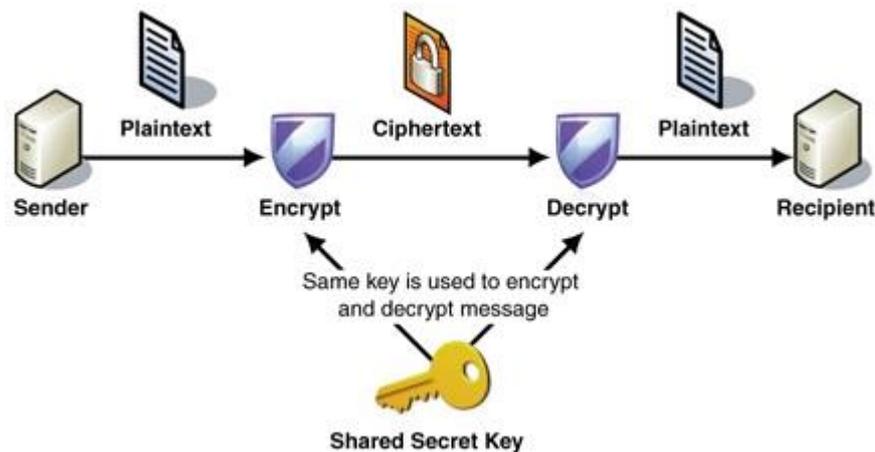


FIGURE 3. The process of symmetric encryption (Microsoft 2005a)

Some of the more common algorithms include DES, 3DES, AES, Software Encryption Algorithm (SEAL), and the Rivest Ciphers (RC) series, which includes RC2, RC4, RC5, and RC6 (Cisco Networking Academy 2012). However, the symmetric encryption has one drawback. The sender and the recipient must exchange a shared secret key, before communication can occur.

2.1.2 Public key infrastructure and asymmetric cryptography

Digital certificate allows a person, organization or computer with using the public key infrastructure (PKI) to exchange the information in secure format over the internet. The PKI assumes the use of public key cryptography. The basic idea of public key

cryptology is that for encryption and decryption are used two separated keys, which are related and combined into a single key pair:

- A public key is a well-known key, which is designed to be spread freely around
- A private or secret key is an encryption/decryption key known only to the party or parties that exchange secret messages. Should take care to ensure that this key should never be publicly disclosed

Data that is encrypted with one key can only be decrypted by the other key (Figure 4) (Microsoft 2005a). This scheme of digital-signing process works also with the digital certificate (DC) itself to identify it as being produced by the certifying authority and serve as proof that digital certificate has not been changed or forged (Microsoft 2014a).

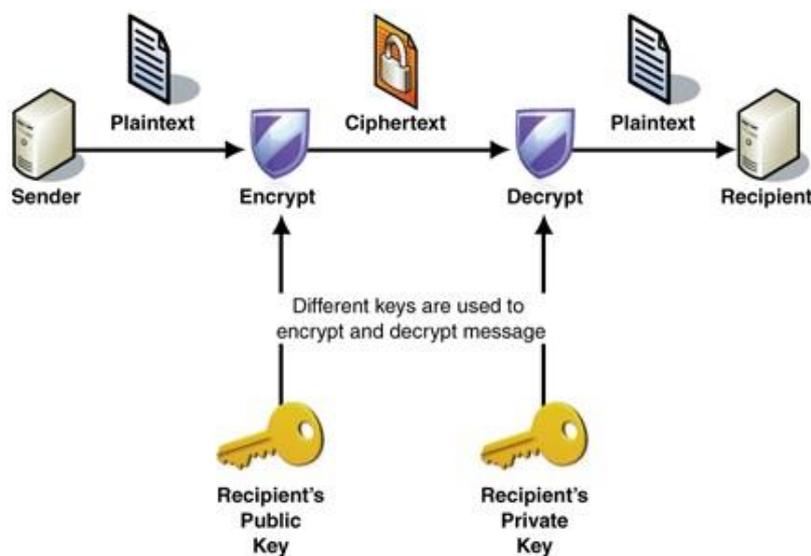


FIGURE 4. The process of asymmetric encryption (Microsoft 2005a)

A data can be decrypted with private key only if it were encrypted with public key, which was created with private key by CA. Rouse (2006) offers the following description of PKI that presented in table 1.

TABLE 1. The principle of public and private key cryptography (Rouse 2006)

To do this	Use whose	Kind of key
Send an encrypted message	Use the receiver's	Public key
Send an encrypted signature	Use the sender's	Private key
Decrypt an encrypted message	Use the receiver's	Private key
Decrypt an encrypted signature	Use the sender's	Public key

In a simple way, it might be represented as a box with a lock. According to Vyrionis (2013), Anna has a box with a lock, which has three states: A, B and C. This lock might be opened with two separate keys. The first key could open the box only turned clockwise ($A \rightarrow B \rightarrow C$) and the second one can turn only anticlockwise ($C \rightarrow B \rightarrow A$). Anna keeps one key to herself (private key) and leaves the second one in a free access (public key). Therefore, Anna has her private key that can turn from A to B to C. Also, everyone else has her public key that can turn from C to B to A.

Now, everyone who wants to send to Anna documents should put it into the box and lock it with the copy of her public key. Anna's public key only turns anticlockwise, so it is necessary to turn it to position A. Now the box is locked. Everyone who has Anna's public key, can put documents in her box, lock it, and know that the only person who can unlock it is Anna.

It also has a possibility to transfer the documents from Anna to everyone. In this case, the lock will be closed in the position C and can be open only with Anna's private key. (Vyrionis 2013).

According to Adams et al. (2001, 616) "PKI has to be able to do the following three things:

- **Manage Keys:** A PKI makes it easy to issue new keys, review or revoke existing keys, and manage the trust level attached to keys from different issues.
- **Publish Keys:** A PKI offers a way for clients to find and fetch public keys and information about whether a specific key is valid or not.
- **Use Keys:** A PKI provides an easy-to-use way for users to use keys. Ranging from moving keys around where they are needed up to providing easy-to-use

applications that perform public key cryptographic operations for securing e-mail, network traffic, and other types of communication.”

PKI consist from several things, such as:

- **Certificate authorities (CA)** is a trusted third party that issues and verifies digital certificate. PKIs can form different topologies of trust, including single-root PKI topologies, hierarchical CA topologies, and cross-certified CA topologies (Figures 5 – 7). (Cisco Networking Academy 2012).

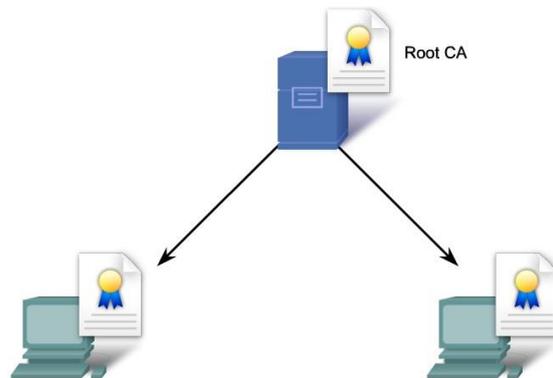


FIGURE 5. Single-root PKI (Cisco Networking Academy 2012)

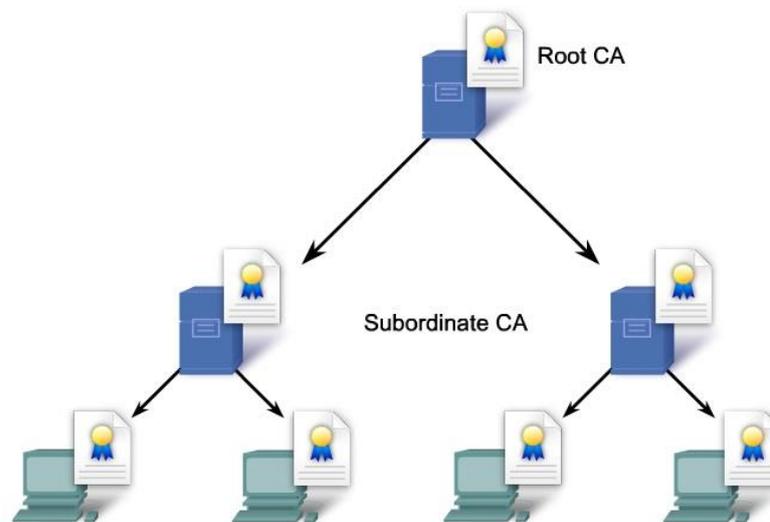


FIGURE 6. Hierarchical CA (Cisco Networking Academy 2012)

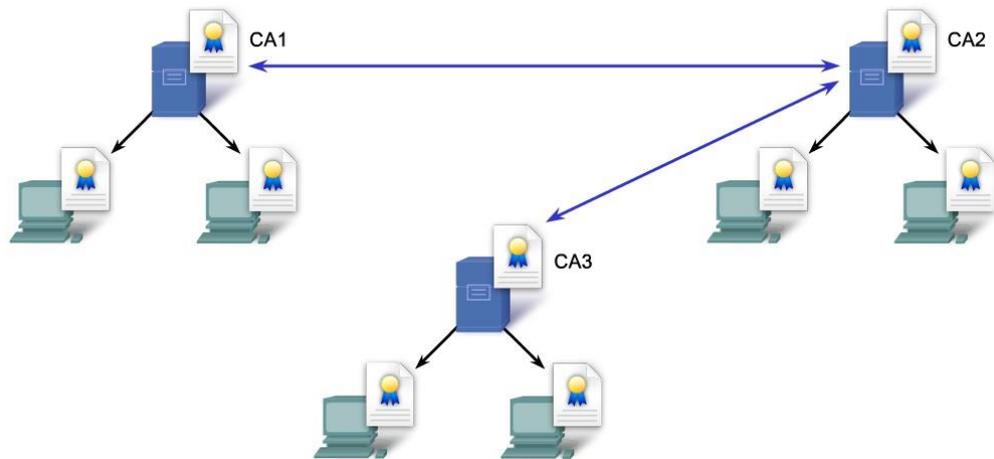


FIGURE 7. Cross-certified CA (Cisco Networking Academy 2012)

The root CA is the CA at the top of hierarchy. CAs, which are lower in the hierarchy, are called intermediate or subordinate CAs. Each CA signs the certificates it issued using its own private key and has own certificate, which contains the CA's public key.

- **Registration authorities (RA)** (Figure 8) verify the user request for a DC and tell the CA to issue it.

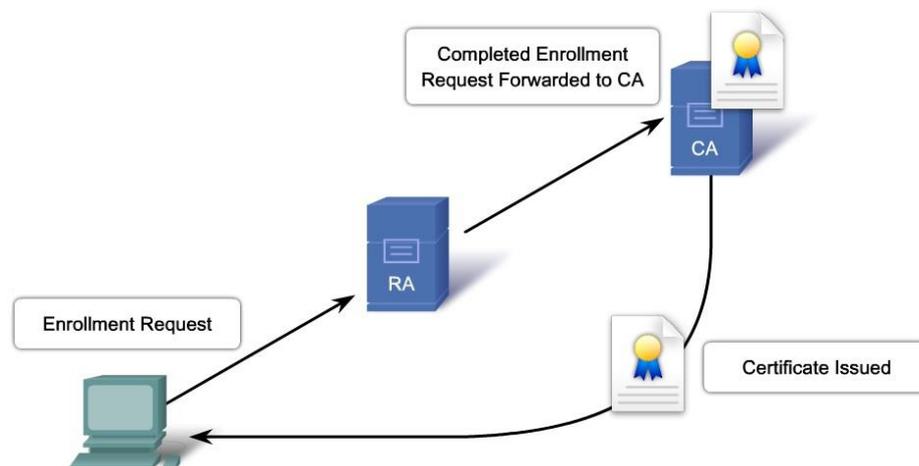


FIGURE 8. Registration authorities (Cisco Networking Academy 2012)

The PKI might employ RAs to accept requests for enrollment in the PKI. RA might be tasked to verify user identity, establishing passwords for certificate management transactions, submitting enrollment requests along with appropriate organizational attributes or other information to the CA. However,

RA is not allowed to issue certificates or publish certificate revocation lists (CRLs). (Cisco Networking Academy 2012).

- **Management tools** is a system that allows an administrator to monitor of which certificates were issued, when a given certificate expires.

2.1.3 Hashing

The hash function is designed to verify and ensure data integrity and authentication. The Hash function takes a variable sized input message and produces a fixed-sized string or message digest (Figure 9).

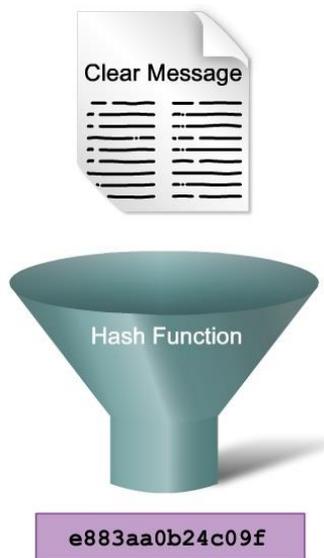


FIGURE 9. A cryptographic hash function (Cisco Networking Academy 2012)

The hash function can guarantee that message was not changed accidentally, but not guarantee that message is not changed purposely. During the session the sending device inputs the message to the hashing algorithm and calculates the fix-length “fingerprint” of this message. The “fingerprint” is attached to the message and sent to a receiver in plaintext. The receiver removes the hash from message and inputs this message to the same hashing algorithm. If the hash that is computed by the receiving device is equal to the one that is attached to the message, the message has not been altered during transit (Cisco Networking Academy 2012). Since the hash depends on all the bits in the input message, any alteration in the original message during the transmission will change the hash. The example of change in hash code produced by

insignificant change in the original message, which was made with MD5 and SHA1 hash functions, is shown below:

To be, or not to be: that is the question.

99c0e4200a8fd19a278d419a881cd222 MD5

6857ddf2f544416ccfb9e1f0c924bd8e778d0227 SHA1

TO be, or not to be: that is the question.

5fb7d665841d9ca182619a855fc10807 MD5

559b890fdc066fc12ebab5c01ee3cc2f57b25c6f SHA1

However, during the transmission the message can be intercepted and changed. An attacker can recalculate the hash and attach it to the message. To prevent man-in-the-middle attacks and provides authentication of the data origin the HMAC was presented.

The HMAC is calculated using a specific algorithm that combines a cryptographic hash function with a secret key. Only the sender and receiver knows this key. The sender puts data and secret key into the hash algorithm and calculates the HMAC “fingerprint”. The receiver removes the “fingerprint” and uses the plaintext message with its secret key as input to the same hashing function. If the fingerprint that is calculated by the receiving device is equal to the fingerprint that was sent, the message has not been altered (Figure 10). (Cisco Networking Academy2012).

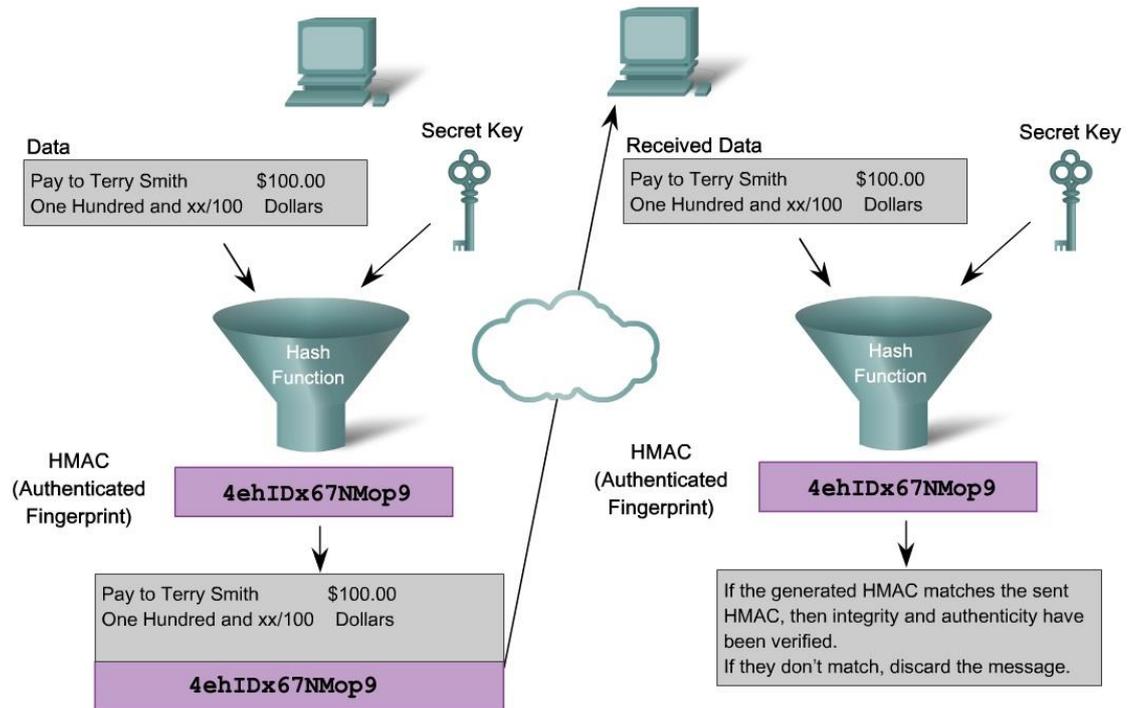


FIGURE 10. HMAC (Cisco Networking Academy2012)

The security of HMAC depends on the security of hashing function and on size and quality of the secret key.

2.2 Alternative CA types

In addition to the above can be added that most PKI systems support two different types of CAs, and each type can operate in one of several roles.

- **The enterprise CA.** The Enterprise Administrator can install the enterprise CA. It acts as a part of PKI for an enterprise. It issues and revoke certificates for end users and intermediate CAs for purposes such as digital signatures, secure e-mail using S/MIME (Secure Multipurpose Internet Mail Extensions), authentication to a secure Web server using Secure Sockets Layer (SSL) or Transport Layer Security (TLS). (Microsoft 2005b).
- **The stand-alone CA.** Stand-alone CAs are issued for external use. These certificates made to issue certificates for Internet users or business partners. In most cases stand-alone certificates is quite similar to enterprise CA. However, stand-alone CA does not require the use of the Active Directory service as enterprise CA. (Microsoft 2005c).

2.3 Self-signed certificate

In case when CA is not available, a self-signed certificate may be all that is required. Self-signed certificate is signed by the same developer who identifies this certificate. This is particularly true in development environment where a developer may simply wish to test that his or her application works over SSL/TLS, or can be used to set up temporary SSL servers. Another application area of self-signed certificate can be situations that require the privacy – to encrypt passwords or other personal information (non-financial).

Nevertheless, this certificate is not secure, and there is no chain of trust. Any remote machine accessing the site will result in a warning being displayed to the user. Schaefer et al. (2013, 487) emphasize that self-signed certificate is not needed to supply a Common Name or any organization details when generating. With this type of certificate, the Common Name is automatically set to the FQDN of the local IIS 8.0 server, and the Organizational Units and organizational details left blank. (Schaefer 2013).

However, self signed certificates have their place in:

- **An Intranet.** When clients only have to go through a local Intranet to get to the server, there is virtually no chance of a man-in-the-middle attack.
- **A development server.** There is no need to spend extra cash buying a trusted certificate for developing or testing an application.
- **Personal sites with few visitors.** A small personal site that transfers non-critical information, there is very little incentive for someone to attack the connections.

In this way a web site, which uses self-signed certificate is saying to a customer: “Trust me – I am who I say I am.” Meanwhile the use of certificate signed by an official CA the web site saying, “Trust me – Verisign agrees I am who I say I am.”

3 HTTP(S)

SSL certificates are required in order to run web sites using the HTTPS protocol. In general, a web site is a set of images, video, pages and other digital content that is available via common host name (<http://www.mysite.com>), or IP address (<http://10.10.10.1>). For today, the dominant protocols for access to a web site are HTTP (HyperText Transfer Protocol) and HTTPS (HyperText Transfer Protocol Secure).

3.1 HyperText Transfer Protocol

HTTP is an application level protocol for transferring files on the World Wide Web. It assumes a reliable, connection – oriented transport protocol such as TCP, but HTTP does not provide reliability or retransmission itself. This protocol was originally developed to publish and retrieve HTML pages. When a user types the URL (Uniform Resource Locator) into address bar of a web browser, the web browser create a connection to the web service with HTTP protocol. After the connection was established, the web server sends the HTML code to the client. Web browser deciphers the HTML code and formats the page for the browser window (Figure 11) (Cisco Networking Academy 2013).

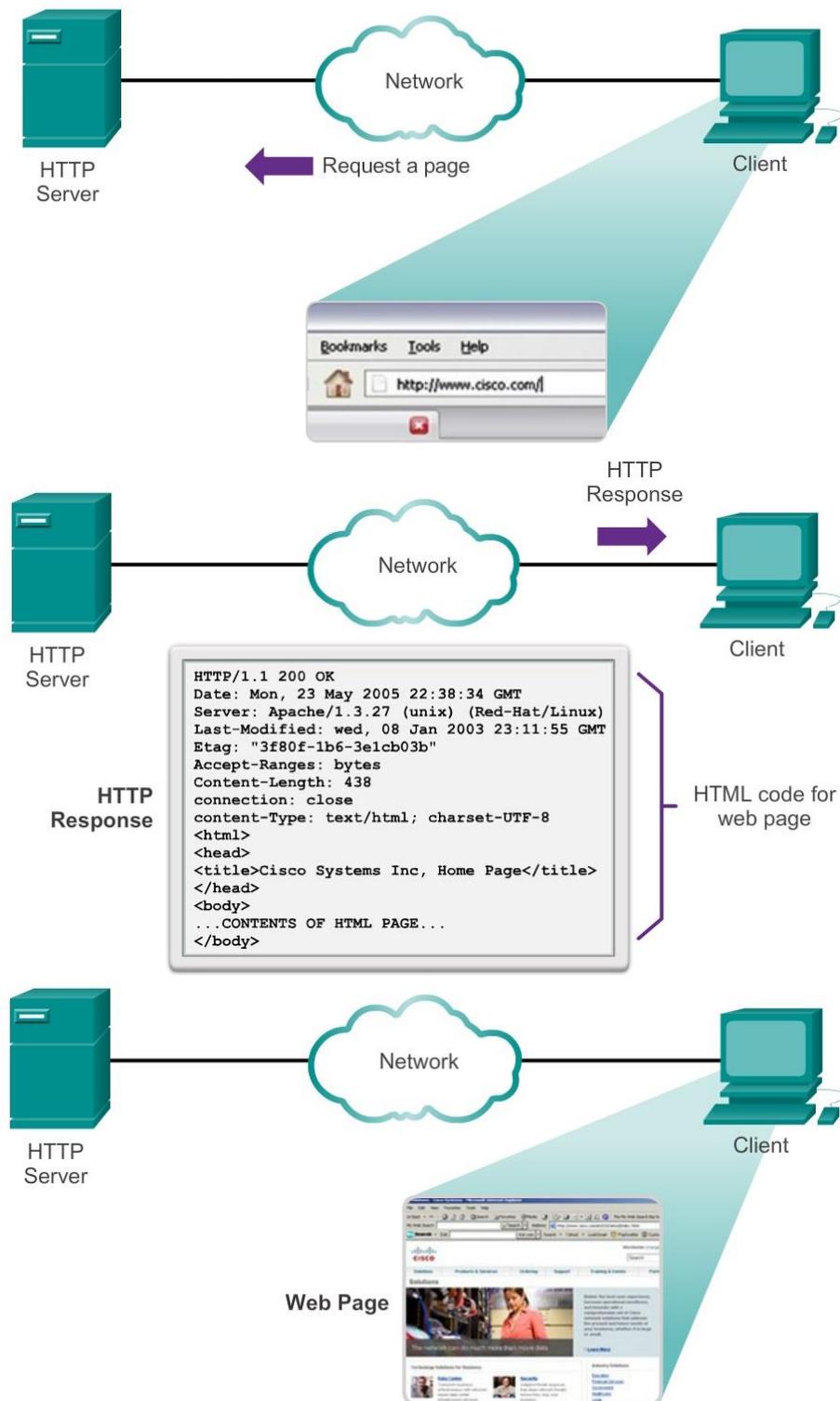


FIGURE 11. Web browser and web client interaction (Cisco Networking Academy 2013)

HTTP specifies as request/response protocol. Each HTTP request is autonomous. This means that the server does not keep the history of previous requests or sessions. When a client sends a request to the server in the form of request method, the HTTP defines

the type of this message and the type of message, which was send from the server as respond. The set of common methods are shown in Table 2 (Sosinsky 2009, 609).

TABLE 2. HTTP Methods (Sosinsky 2009, 609)

Method	Action	Safe
CONNECT	Creates a tunnel using an established network connection. CONNECT is most often used to send encrypted data over HTTPS.	No
DELETE	Deletes the specified resources given by URL.	No
GET	Requests a resource. Requests using GET should only retrieve data and should have no other effect on the data. Refresh button of a browser.	Yes
HEAD	Requests a resource, but only transfer the status line and header section. This is used for retrieving metadata.	Yes
OPTIONS	Requests that the server return a list of methods that the server supports for the resource that is specified.	Yes
POST	Sends data to the server. Submit button of a browser.	No
PUT	Uploads a resource.	No
TRACE	Requires an echo response for a request. TRACE allows the action of any intermediaries to be examined.	Yes

The Tutoriaspoint (2014) gives the examples of each method and you can find it in Appendix 1.

For improving response time, a browser creates cache copy of each web page, which was visited by the client earlier. If the user decides to visit some of the previous pages again, browser creates the request, simultaneously HTTP allows the browser to ask the server to determinate whether the content of this page has changed since the copy was cached. (Comer 2000, 530).

However, if there is any problem, the web server will return a one-line status code to the client, which interprets the response and either displays it in the browser or acts upon it. Error message, usually in valid HTML format, is displayed when a server cannot respond for a request for some reasons. Comer (2000, 531) in his book gives the example of the following error message:

```
<HTML>

<HEAD><TITLE>400 Bad Request</TITLE>

</HEAD>

<BODY>

    <H1>Bad Request</H1> Your browser sent a request that this server
    could not understand.

</BODY>

</HTML>
```

User can see only the “body” (i.e., the items between <BODY> and </BODY>). *Bad Request* as a head of message.

The most famous “Error 404 – Not Found” might be displayed differently in different browsers (Figure 12 – 14).

This page can't be displayed

- Make sure the web address <http://168.192.104.2> is correct.
- Look for the page with your search engine.
- Refresh the page in a few minutes.

[Fix connection problems](#)

FIGURE 12. Error message in Internet explorer ver. 11

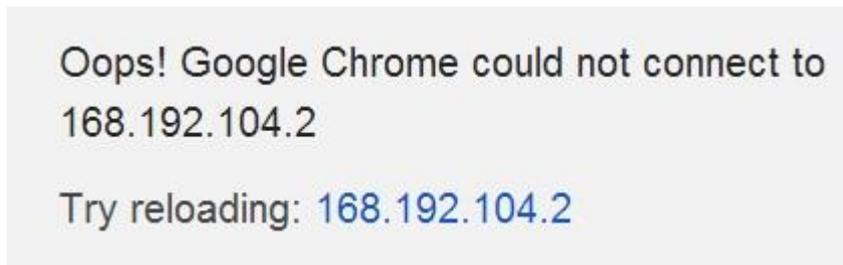


FIGURE 13. Error message in Google Chrome ver. 33.0

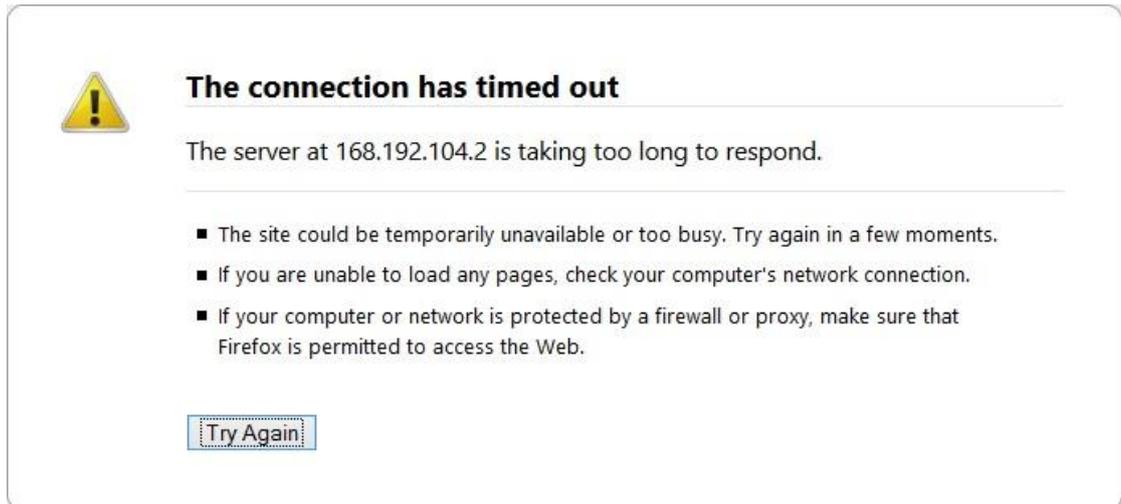


FIGURE 14. Error message in Mozilla Firefox ver. 28.0

At the HTTP level, a response code is followed by a human-readable "reason phrase". Generally, the 404 error returned when page have been moved or deleted.

3.1.1 Security issues of HTTP

In this section some essential security issues of HTTP are covered and it is mainly based on information from w3 (2004) web site. Because, the HTTP is a dominant protocol for access to the web, and the request messages send information to the server in plain text. This situation makes the protocol vulnerable for malefactors. HTTP clients are often privy to large amounts of personal information, such as passwords, e-mail addresses the user's name, locations, etc. All this information can be used against the user or his/her privacy. Like any generic data transfer protocol, HTTP cannot regulate the content of data that is transferred. Therefore, applications should supply as much information to provider, as possible. The opened name of software version of the server also can be a reason for attacks against software that is known to contain security holes. Clients using HTTP rely heavily on the Domain

Name Service, and are thus generally prone to security attacks based on the deliberate mis-association of IP addresses and DNS names. So, clients need to be cautious with DNS name association and IP addresses.

3.2 HTTPS (HTTP Secure)

HTTPS is used for secure communication across the Internet. As a result of combination the HTTP and SSL (Secure Sockets Layer) or TLS (Transport Layer Security), HTTPS can use authentication and encryption to secure data during the transferring between client and server. The use of HTTPS protects against eavesdropping, tampering and man-in-the-middle attack. In practice, this provides a guarantee that the contents of communications between the client and server cannot be read or forged by any third party. To determine if HTTPS is used or not, it is enough to check URL address. HTTP URLs begin with "http://" and use port 80 by default, whereas HTTPS URLs begin with "https://" and use port 443 by default.

Because the HTTPS is a combination of SSL/TLS and HTTP, it uses the same client request-server response process as HTTP, but the data stream is encrypted with SSL before being transported across the network. This means that the request URL query parameters, headers and cookies will be encrypted. However, TCP-port and host address are part of TCP/IP protocols and cannot be protected. This makes a possibility for malefactors to indicate the IP address and port number of the server. However, this information cannot be too beneficial.

According to Symantec (2011), HTTPS connection to a website can be trusted if and only if all of the following are true:

- The user trusts that their browser software correctly implements HTTPS with correctly pre-installed certificate authorities.
- The user trusts the certificate authority to vouch only for legitimate websites without misleading names.
- The website provides a valid certificate, which means it was signed by a trusted authority.

- The certificate correctly identifies the website (e.g., when the browser visits "https://example", the received certificate is properly for "Example Inc." and not some other entity).
- Either the intervening hops on the Internet are trustworthy, or the user trusts that the protocol's encryption layer (TLS or SSL) is unbreakable by an eavesdropper.

As of 04/03/2014, 23,9% from 158270 most popular surveyed sites have a secure implementation of HTTPS (Figure 15) (TIM 2014).

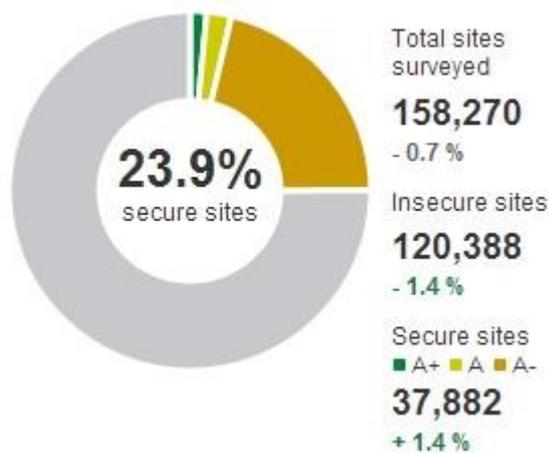


FIGURE 15. The effective SSL security implemented by the most popular web sites (TIM 2014)

Such a small percentage explained by the fact that not for all websites needed encrypted connection, if there is no any private information or information, which is sensitive for users.

4 DNS – DOMAIN NAME SYSTEM

In the computer world everything is decided by numbers. The address of MAMK web site is www.mamk.fi, presented as a simple, recognizable name, but the real address of this web site is <http://195.148.216.131/>. Most people cannot remember numeric addresses of the favourite web sites. Domain names were created to make life easier. DNS automatically converts the names, which the user types in a web browser address

bar to the IP addresses of web servers hosting those sites (Figure 16). Even if the numeric IP address will change, users will not see any changes. The new address is replaced with former, linked to the existing domain name and connectivity is retained.

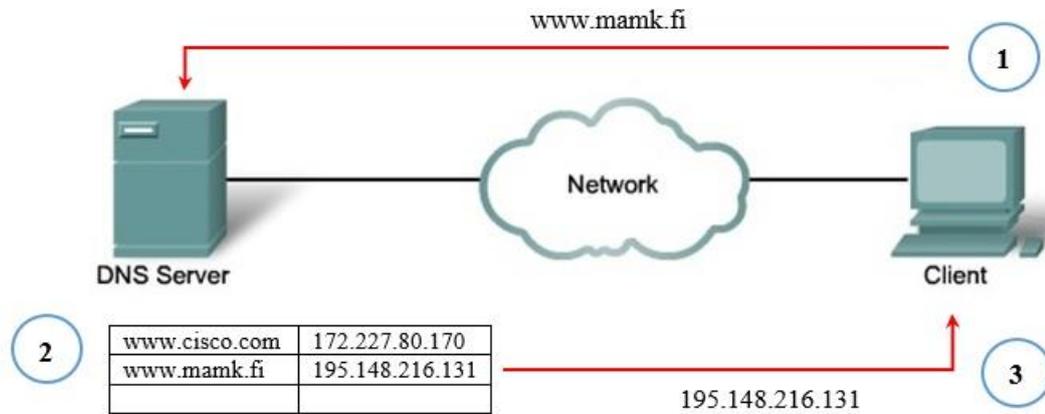


FIGURE 16. Domain Name System

The user’s computer sends to the server a special packet – DNS request. This packet contains the alphabetical form of the web site. If the server knows an answer to the request it responds and sends back the numerical address. If the server does not know the answer, the error message will be returned. (Garfinkel & Spafford 2002, 25 - 28).

The DNS protocol uses a hierarchical system to create a database to provide name resolution. After top-level domains are second-level domain names, and below them are other lower level domains (Figure 17).

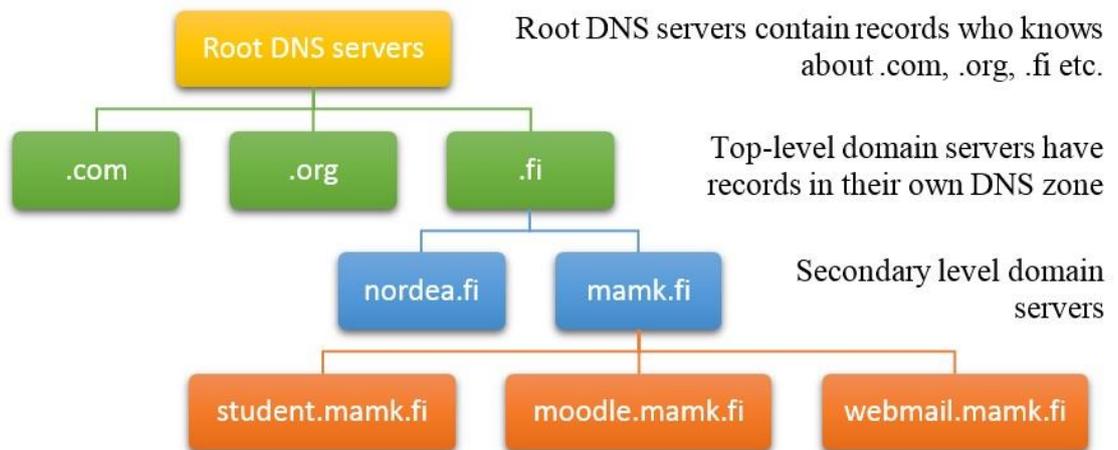


FIGURE 17. A hierarchy of DNS servers (Gromyko 2014)

Each DNS server is responsible for small part of domain names. If the requirement domain is not in the list of the current DNS server, this server sends the request to the upper level. (Cisco Networking Academy 2013).

4.1 DNS queries

Two types of DNS queries are exists: recursive and iterative (non-recursive).

With the *recursive query*, DNS server responds to the client with either the requested resource record or an error message stating that the record or domain name does not exist (Microsoft 2014b). Therefore, if client needs information about web site, it sends request to a nearby DNS server. When the DNS server receives a request for a name translation that is not within its DNS zone, the DNS server forwards the request to another DNS server within the proper zone for translation until it gets the information, or until the name query fails. For example:

1. Client → Local DNS server → .com DNS server → Root DNS server → .fi DNS server: “I need information about www.mamk.fi”.
2. .fi DNS server → Root DNS server → .com DNS server → Local DNS server → Client: “IP address of www.mamk.fi is 195.148.216.131”.

With the *iterative query*, DNS server to return the best answer it can give based on its cache or zone data. If nearby DNS server does not have an exact information, the best possible information it can return is a referral to the DNS server, which “could know”. The client can send the same request to the DNS server for which it obtained a referral. This process will stop when (Microsoft 2014b):

- it locates a DNS server that is authoritative for the queried name;
- receive an error message;
- time-out condition is met.

The example of iterative query is shown below.

1. Client: “I need information about www.mamk.fi”.

Local DNS server: “I do not know, but .com DNS server could know it.”

2. Client: "I need information about www.mamk.fi".

.com DNS server: "I do not know, but Root DNS server could know it."

3. Client: "I need information about www.mamk.fi".

Root DNS server: "I do not know, but .fi DNS server could know it."

4. Client: "I need information about www.mamk.fi".

.fi DNS server: "IP address of www.mamk.fi is 195.148.216.131"

However, the usage of one type of queries is inefficient, because top level and root DNS servers could not store all query information in cache. A combination of DNS queries is usually used (Figure 18).

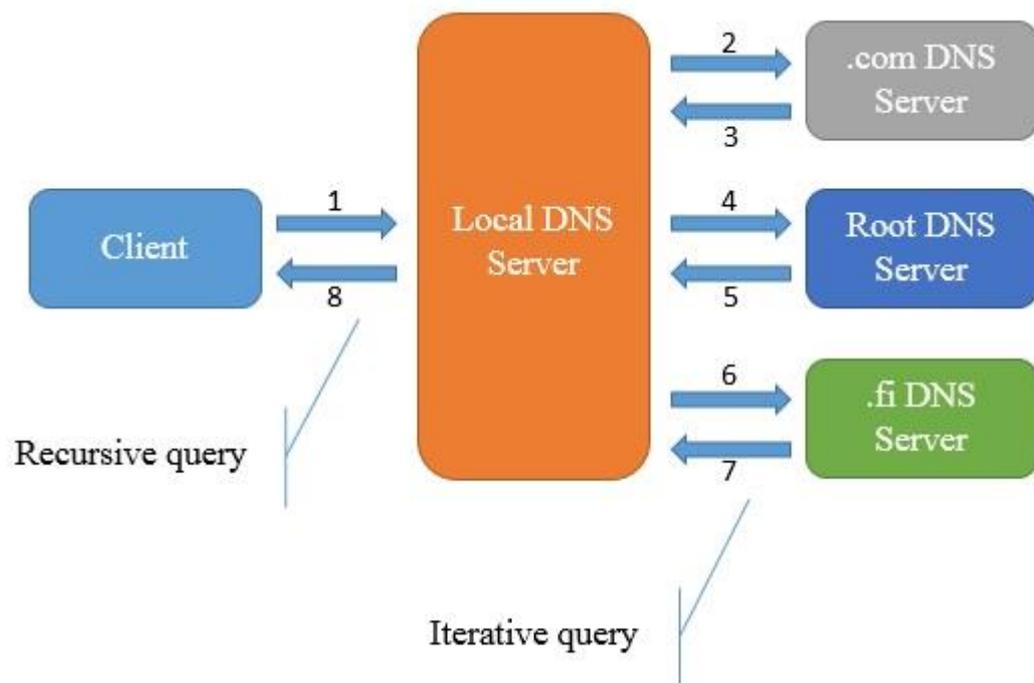


FIGURE 18. Combination of DNS queries

As presented above, iterative query is easier to use on the high levels of DNS hierarchy. It will not take extra time for server to find the information in cache and do not consume overmuch bandwidth.

5 SELF-SIGNED CERTIFICATE CREATION

A certificate serves two essential purposes: distributing the public key and verifying the identity of the server so visitors know they are not sending their information to the wrong person. User or company can buy the digital certificate from trusted certificate authority. However, if the certificate is needed only for testing or local network it is not reasonable to pay for it. In this section, I create free digital certificate for my own www.mywebpage.fi.

5.1 Environment and topology

For creation of the digital certificate, I use Windows OS. The most software for these purposes are made for UNIX. For that reason, to find the solution for Windows was another challenge.

Creation and checking of operability of the certificate require at least:

- Server with my website
- DNS server
- Client PC

It is realised using VMware Player 6.0.2. with three virtual machines under Windows 8.1 OS (Figure 19).

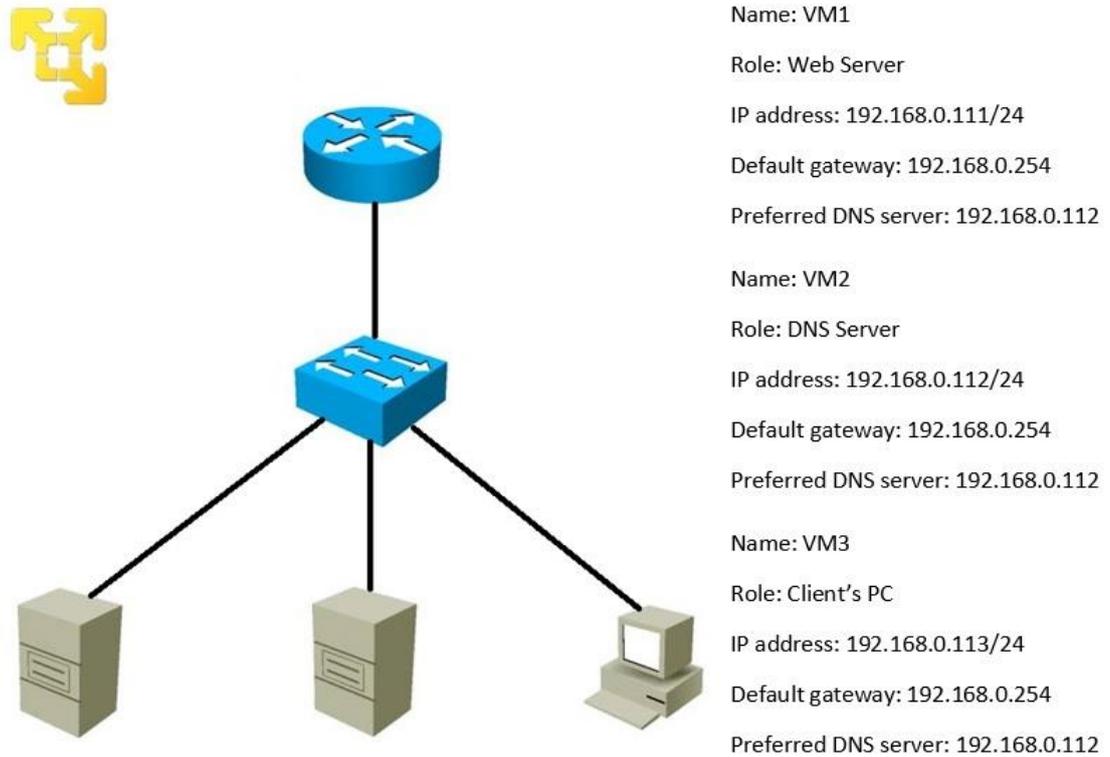


FIGURE 19. Topology and IP settings

In addition to that, on the Web Server machine Apache 2.2.25 with Openssl-0.9.8y mod are installed, DNS server machine has Unbound DNS server and the Client machine use Wireshark and web browsers (Opera, Chrome, Firefox, Internet Explorer).

Used software

VMware Player 6.0.2

Windows 8.1 Pro

Unbound 1.4.22

Apache 2.2.25 with Openssl 0.9.8y

Wireshark 1.10.7

Mozilla Firefox 29.0.1

Google Chrome 34.0.1847.131 m

Opera 21.0.1432.57

Internet Explorer 11.0.96

5.2 Certificate creation and DNS installation

The information and commands for certificate creation is based on information from Rubayat (2010). The description of the commands, which were used for creation this certificate, is based on information from OpenSSL Documents (2014). To create the certificate on Windows OS OpenSSL is required. For some reasons, OpenSSL does not contain *openssl.cnf*, which is the OpenSSL configuration file. It should be downloaded separately and replaced to the *\bin* directory

In the command line, I open the directory where OpenSSL is.

```
C:\Windows\system32>cd C:\Program Files\Apache Software  
Foundation\Apache2.2\bin
```

To create a new certificate request the following command is needed.

```
openssl req -new -out server.csr
```

This generates a 1024-bit RSA private key.

The *req* command primarily creates and processes certificate requests.

The *-new* command generates a new certificate request.

The *-out* command specifies the output filename to write to or standard output by default.

After that, OpenSSL promoted to answer some questions, where a field “Common Name” should be filled in with the fully qualified domain name or IP address of the server to be protected by SSL. An extension “.pem” is a password associated with the private key.

The command below creates a non-password protected key.

```
openssl rsa -in privkey.pem -out server.key
```

The command *rsa* is RSA key management.

The last command creates an X.509 certificate, for 365 days.

```
openssl x509 -in server.csr -out server.crt -req -signkey server.key -days 365
```

The *x509* command outputs a self signed certificate instead of a certificate request.

The *-in* command specifies the input filename to read a request from or standard input if this option is not specified.

The *-days* command specifies the number of days to certify the certificate for.

The self-signed certificate is ready now (Figure 20). Full list of commands you can find in Appendix 2.

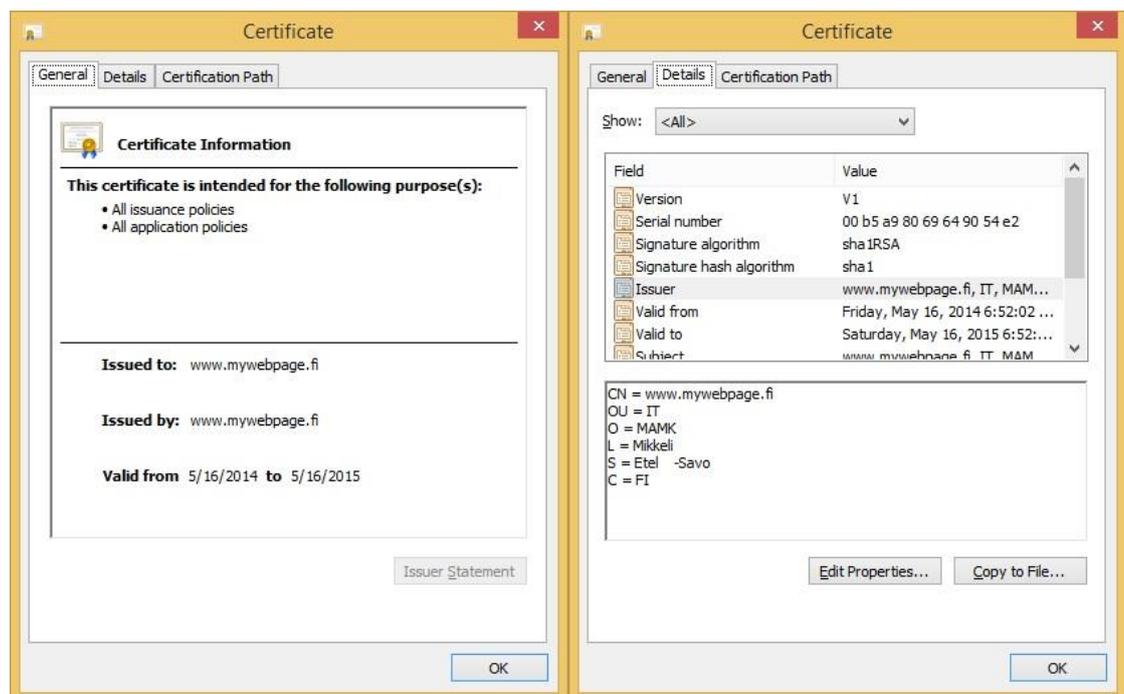


FIGURE 20. Self-signed certificate

In Apache 2.2.25 open *httpd.conf* file which is located in the `\conf` folder. Change the *ServerRoot...* to the *ServerRoot "C:/Program Files/Apache Software Foundation/Apache2.2"*. In the lines *DocumentRoot...* and *Directory...* indicate the way to the place where website is. Uncomment (remove # sign) from lines:

- `LoadModule ssl_module modules/mod_ssl.so` – it allows to use SSL mod on Apache server
- `LoadModule rewrite_module modules/mod_rewrite.so` – mod for rewrite requested URLs.

To display the page `www.mywebpage.fi` only in `https://` form at the end of the document is necessary to add next lines.

```
<VirtualHost *:80>

    RewriteEngine On

    RewriteCond %{HTTPS} off

    RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

</VirtualHost>
```

In the `\conf` create `\ssl` directory and copy `server.crt` and `server.key` there.

In the `\conf\extra\httpd-ssl.conf` file modify `DocumentRoot` and `ServerName` for own data.

```
# General setup for the virtual host

DocumentRoot "C:/Program Files/Apache/Apache2/htdocs/website"

ServerName www.mywebpage.fi:443
```

At the end of configuration is needed to show the way to certificate and key.

```
SSLCertificateFile      "C:/Program      Files/Apache      Software
Foundation/Apache2.2/conf/ssl/server.crt"

SSLCertificateKeyFile  "C:/Program      Files/Apache      Software
Foundation/Apache2.2/conf/ssl/server.key"
```

Restart the Apache server.

For DNS server in the Unbound directory find and change service.conf file. The line *interface:...* should be changed to *interface: 192.168.0.112*. This is the interface for DNS server and this interface should be mark as DNS server address on all computers.

Add lines for local network:

```
private-domain: "mywebpage.fi"
```

```
local-zone: " mywebpage.fi" redirect
```

```
local-data: " mywebpage.fi A 192.168.0.111"
```

```
local-data-ptr: "192.168.0.111 www. mywebpage.fi"
```

Last two lines connect the IP address (192.168.0.111) and the domain name (www.mywebpage.fi) together.

The last thing is to restart DNS server.

5.3 Testing with Firefox, Chrome, Opera and IE

5.3.1 Mozilla Firefox

If the certificate was not added to the *Authorities* in Certificate Manager, when the line www.mywebpage.fi is added to the address bar, the Firefox shows warning that connection is untrusted (Figures 21).

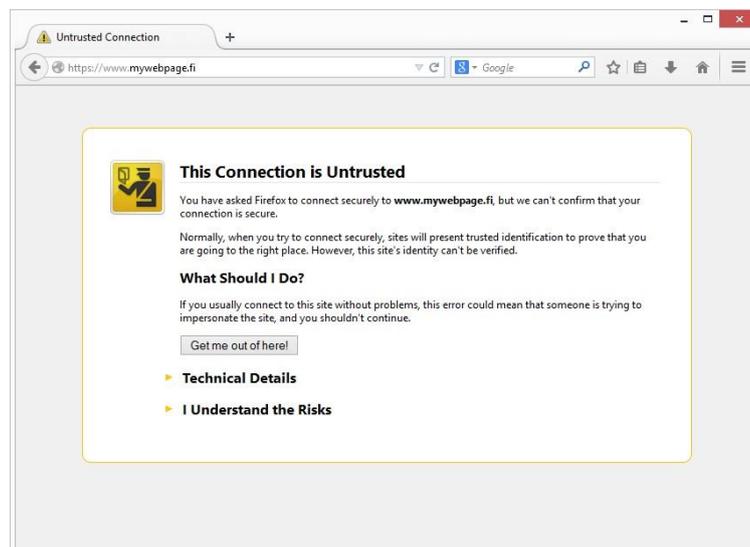


FIGURE 21. Mozilla Firefox warning

The connection is encrypted with SSL protocol and verified by MAMK (Figure 22).

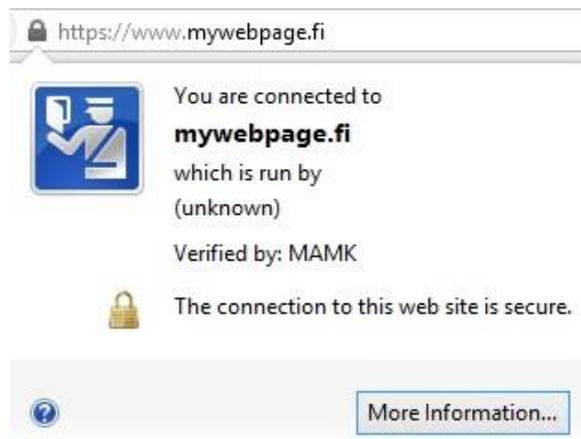


FIGURE 22. Secure connection in the Firefox

5.3.2 Google Chrome

The first start of www.mywebpage.fi in Chrome also gives warning that certificate is not trusted. Moreover, the field https:// crossed out (Figure 23).

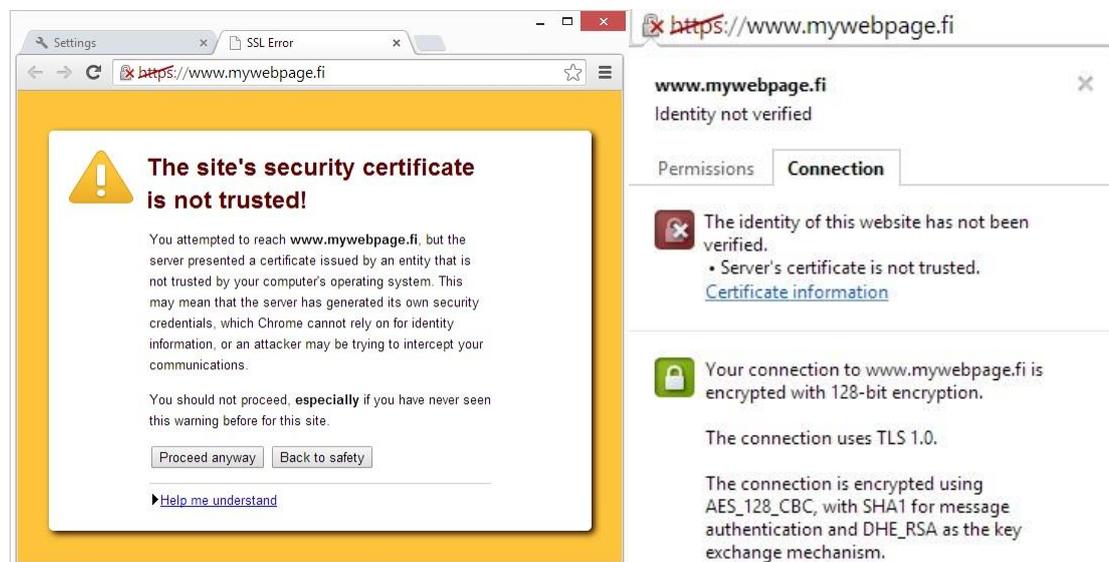


FIGURE 23. Google Chrome https connection

However, it allows encrypted connection with using AES.

5.3.3 Opera

After the warning was accepted, Opera browser does not show https://: in the address bar absolutely (Figure 24).

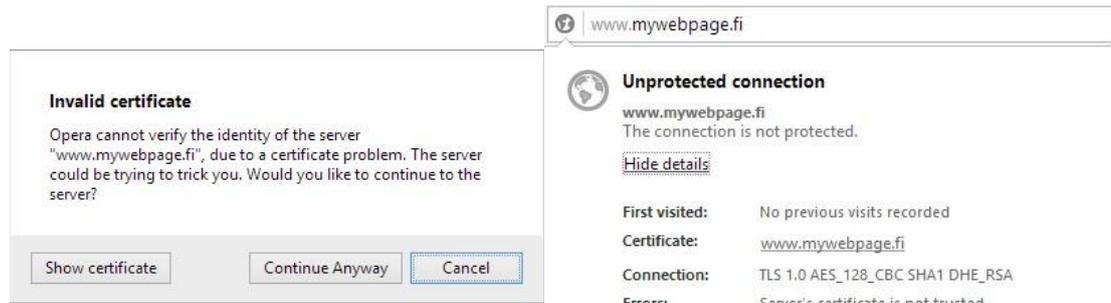


FIGURE 24. Opera warnings

Nevertheless, connection is used TLS 1.0 and AES encryption.

5.3.4 Internet Explorer

As the three previous browsers, Internet Explorer also gives warning that certificate is not trusted, but it uses secured connection (Figure 25).

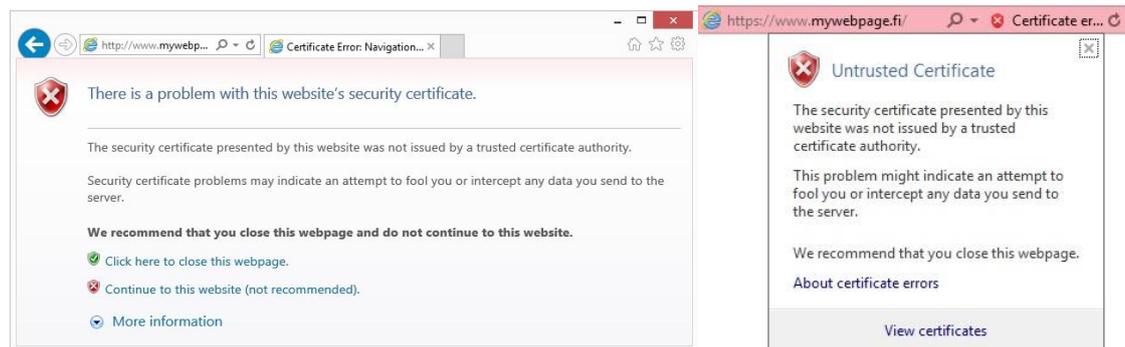


FIGURE 25. Internet Explorer warnings

It occurs when the web site is available in the Internet. To avoid these warnings users need to install the root certificate to the local computers manually.

5.3.5 Trusted Root Certification Authorities

The situation is changed when the certificate is added to the *Trusted Root Certification Authorities*. The way to this directory is different in different browsers. For Firefox Mozilla it is needed to open Certificate Manager, which located on:

Menu → *Options* → *Advanced* → *Certificates* → *View Certificates*

Choose *Authorities* and *Import* server.crt file to the directory (Figure 26).



FIGURE 26. Downloading a new certificate into the Mozilla Firefox

Select checkbox “Trust this CA to identify web sites.”, otherwise Certificate Manager will not trust web site certificates issued by this CA. After downloading, the certificate will be added to the list of trusted certificates (Figure 27).

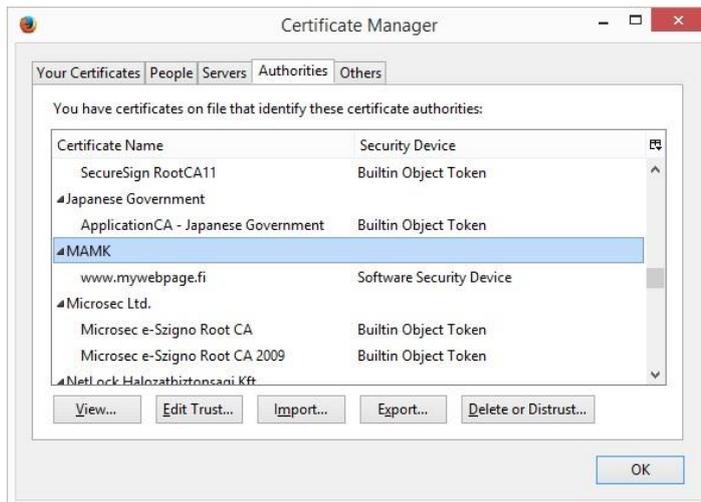


FIGURE 27. Self-signed certificate in the Firefox Authorities

When the certificate was added to the Google Chrome web browser trusted certificates, it was also added to the Opera and Internet Explorer automatically. So, I explain the steps only for Google Chrome.

For Google Chrome Certificate Manager located on:

Settings → *HTTPS/SSL* → *Manage certificates...*

Choose *Trusted Root Certification Authorities* and *Import* server.crt file to this directory following the instructions. After successful completion, the certificate will be added to the trusted root certificates (Figure 28).

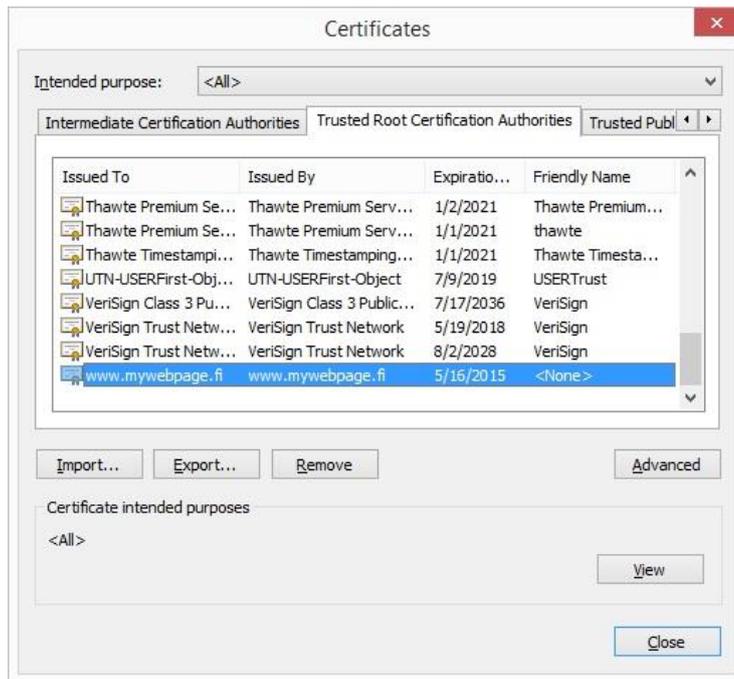


FIGURE 28. Trusted Root Certification Authorities

After these actions, none of the browsers does not open any warnings and access to the www.mywebpage.fi is freely open under HTTPS protocol (Figures 29 –31).



FIGURE 29. Google Chrome



FIGURE 30. Opera



FIGURE 31. Internet Explorer

This certificate is valid only for *www.mywebpage.fi* and do not accept *mywebpage.fi* (Figures 32 – 34). Nonetheless, the connection to the web page has occurred, but certificate is not trusted.

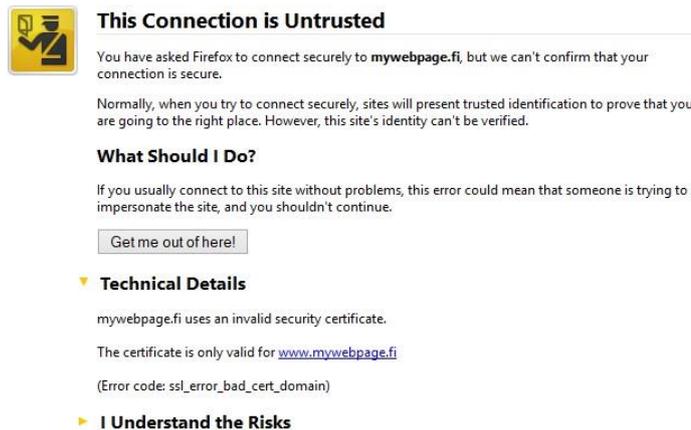


FIGURE 32. Firefox certificate warning

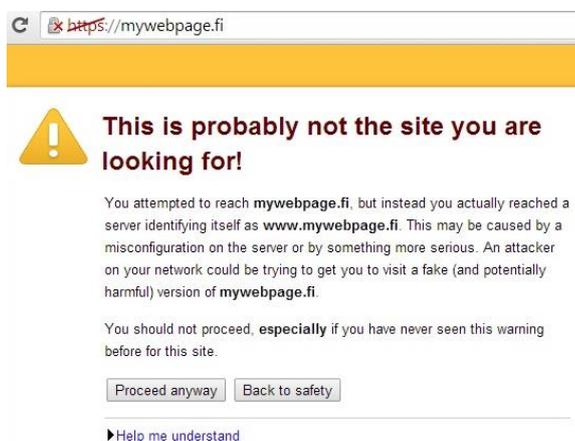


FIGURE 33. Chrome certificate warning

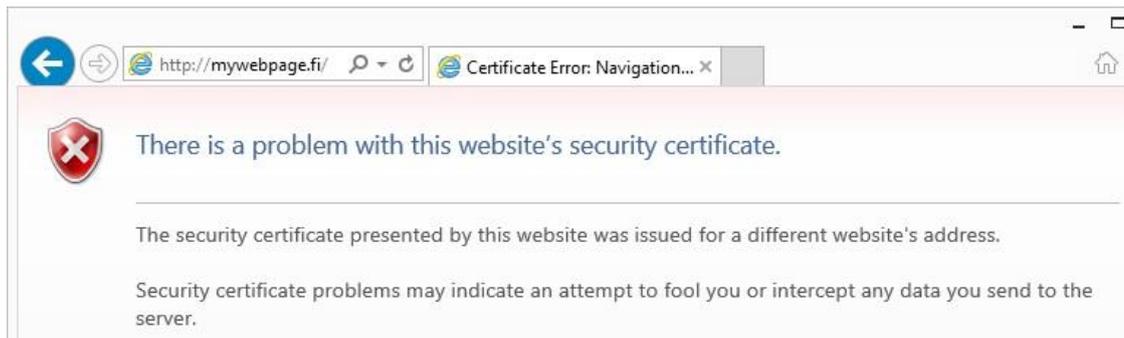


FIGURE 34. Internet Explorer certificate warning

During connection setup, *Encrypted Alert* message has occurred with *Content Type: Alert (21)* (Figure 35).

6	0.00466200	192.168.0.113	192.168.0.111	TCP	66 49282 > https [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	0.00491200	192.168.0.111	192.168.0.113	TCP	66 https > 49282 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
8	0.00494800	192.168.0.113	192.168.0.111	TCP	54 49282 > https [ACK] Seq=1 Ack=1 win=65536 Len=0
9	0.00509700	192.168.0.113	192.168.0.111	TLSv1	231 Client Hello
10	0.00999400	192.168.0.111	192.168.0.113	TLSv1	1104 Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	0.01143100	192.168.0.113	192.168.0.111	TLSv1	252 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.01185400	192.168.0.113	192.168.0.111	TLSv1	91 Encrypted Alert
13	0.01197400	192.168.0.113	192.168.0.111	TCP	54 49282 > https [FIN, ACK] Seq=413 Ack=1051 win=64512 Len=0
14	0.01342100	192.168.0.111	192.168.0.113	TCP	60 https > 49282 [ACK] Seq=1051 Ack=414 win=65280 Len=0
15	0.01542100	192.168.0.111	192.168.0.113	TLSv1	113 Change Cipher Spec, Encrypted Handshake Message
16	0.01545000	192.168.0.113	192.168.0.111	TCP	54 49282 > https [RST, ACK] Seq=414 Ack=1110 win=0 Len=0
17	0.04949000	192.168.0.113	192.168.0.111	TCP	54 49281 > http [ACK] Seq=278 Ack=583 win=65024 Len=0
18	2.57080800	192.168.0.113	192.168.0.111	TCP	66 49283 > https [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
19	2.57148600	192.168.0.111	192.168.0.113	TCP	66 https > 49283 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
20	2.57153500	192.168.0.113	192.168.0.111	TCP	54 49283 > https [ACK] Seq=1 Ack=1 win=65536 Len=0
21	2.57172500	192.168.0.113	192.168.0.111	TLSv1	231 Client Hello
22	2.57753200	192.168.0.111	192.168.0.113	TLSv1	1104 Server Hello, Certificate, Server Key Exchange, Server Hello Done
23	2.58010400	192.168.0.113	192.168.0.111	TLSv1	252 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
24	2.58100200	192.168.0.113	192.168.0.111	TLSv1	91 Encrypted Alert
25	2.58120900	192.168.0.113	192.168.0.111	TCP	54 49283 > https [FIN, ACK] Seq=413 Ack=1051 win=64512 Len=0
26	2.58213600	192.168.0.111	192.168.0.113	TCP	60 https > 49283 [ACK] Seq=1051 Ack=414 win=65280 Len=0
27	2.58401100	192.168.0.111	192.168.0.113	TLSv1	113 Change Cipher Spec, Encrypted Handshake Message
28	2.58402800	192.168.0.113	192.168.0.111	TCP	54 49283 > https [RST, ACK] Seq=414 Ack=1110 win=0 Len=0
29	5.05217900	Vmware_0a:d0:b9	Broadcast	ARP	60 who has 192.168.0.254? Tell 192.168.0.112
30	5.66049800	Vmware_0a:d0:b9	Broadcast	ARP	60 who has 192.168.0.254? Tell 192.168.0.112
31	6.66078500	Vmware_0a:d0:b9	Broadcast	ARP	60 who has 192.168.0.254? Tell 192.168.0.112
32	6.69641600	192.168.0.113	192.168.0.111	TCP	66 49284 > https [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
33	6.69673800	192.168.0.111	192.168.0.113	TCP	66 https > 49284 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
34	6.69678200	192.168.0.113	192.168.0.111	TCP	54 49284 > https [ACK] Seq=1 Ack=1 win=65536 Len=0
35	6.69693300	192.168.0.113	192.168.0.111	TLSv1	231 Client Hello
36	6.70223500	192.168.0.111	192.168.0.113	TLSv1	1104 Server Hello, Certificate, Server Key Exchange, Server Hello Done
37	6.70632700	192.168.0.113	192.168.0.111	TLSv1	252 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
38	6.71001100	192.168.0.111	192.168.0.113	TLSv1	113 Change Cipher Spec, Encrypted Handshake Message
39	6.71018000	192.168.0.111	192.168.0.111	TLSv1	400 Application Data, Application Data
40	6.71249100	192.168.0.111	192.168.0.113	TLSv1	1514 Application Data
41	6.71249100	192.168.0.111	192.168.0.113	TCP	1514 [TCP segment of a reassembled PDU]
42	6.71249100	192.168.0.111	192.168.0.113	TCP	1514 [TCP segment of a reassembled PDU]

FIGURE 35. Encrypted alert

This alert warns that, “handshake cryptographic operation failed, including being unable to correctly verify a signature, decrypt a key exchange, or validate a finished message” (RFC_2246 1999). The message can be sent at any time during the handshake and up to the closure of the session. This is a fatal error and the session is closed immediately after alert was sent. Fatal errors occurs when connection or security may be compromised. In this project an alert was sent because the certificate is untrusted and asks additional verifying. However, Internet Explorer does not show any alerts during the session. After certificate was added to the trusted root certificates this alert is disappeared.

6 CONCLUSION

Theme of obtaining digital certificates is very important because certificate provides the necessary level of protection for personal information. In the rapidly growing volume of information on the Internet, including sensitive information, encryption, authentication, and data integrity is highly relevant. There are many organizations issuing digital certificates, but they all cost money, what is not always justified. In this project, I considered an alternative method of obtaining the certificate without additional cost. A self-signed certificate is such certificate that is created, signed and used by one person or a company. The main advantages are availability, ease of establishment and free of charge. Any operating system may be used for creation, but I used Windows 8.1. In this project, I proved the possibility of creation and implementation of a self-signed certificate on the example of a local web site www.mywebpage.fi. It uses AES encryption and establishes HTTPS connection instead of HTTP. The certificate has been analysed in four different browsers. The warnings were displayed only for those users, who did not install it into trusted certificates authorities in the browser. Other series of experiments with Wireshark showed that connection works with TLS and HTTPS protocol in any cases.

This certificate can be used on a development server, local network or for small web sites with a small amount of visitor. It provides necessary encryption and can be made easily by free sources.

However, a self-signed certificate is not as trusted as certificate, which have been issued by a CA. It is not recommended to use the certificate for e-commerce or any site that transfers valuable personal information as credit cards, social security numbers, etc.

BIBLIOGRAPHY

Adams, Maureen (ed.) 2001. Networking complete second edition. San Francisco, Paris, Düsseldorf, Soest, London: Sybex, Inc.

Business Dictionary 2014. Definition of passport. WWW-document.
<http://www.businessdictionary.com/definition/passport.html/>. No updating information. Referred 4.2.2014.

Cisco Networking Academy 2012. CCNA Security 1.1. Cryptographic Systems. WWW-lecture. <http://cna.mikkeli.amk.fi/Cisco/CCNASecurity>. Updated 2012. Referred 11.4.2014.

Cisco Networking Academy 2013. Introduction to Networks, Application Layer. WWW-lecture.
http://cna.mikkeli.amk.fi/cisco/CCNAv_5/Introd_To_Netw/index.html. Updated 2013. Referred 7.4.2014.

Comer, Douglas 2000. Internetworking with TCP/IP Principles, protocols, and architectures 4th ed. USA: Prentice Hall.

Evans, Dave 2011. The Internet of Things. WWW-document.
<http://blogs.cisco.com/diversity/the-internet-of-things-infographic/>. Updated 15.6.2011. Referred 8.4.2014.

Garfinkel, Simson & Spafford, Gene 2000. Web security, privacy & commerce, 2nd ed. USA: O'Reilly & Associates, Inc.

Gromyko Evgeniia 2014. Investigation of digital certificate: Verification of reliability and resistance to external attacks. Finland: MAMK.

Microsoft 2005a. Data confidentiality. WWW-document.
<http://msdn.microsoft.com/en-us/library/ff650720.aspx>. Updated 12.2005. Referred 8.4.2014.

Microsoft 2005b. Enterprise certification authorities. WWW-document.
[http://technet.microsoft.com/en-us/library/cc776874\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc776874(v=ws.10).aspx). Updated
21.1.2005. Referred 15.2.2014.

Microsoft 2005c. Stand-alone certification authorities. WWW-document.
[http://technet.microsoft.com/en-us/library/cc780501\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc780501(v=ws.10).aspx). Updated
21.1.2005. Referred 15.2.2014.

Microsoft 2014a. What are digital certificates? WWW-document.
<http://msdn.microsoft.com/en-us/library/office/>. No updating information. Referred
4.2.2014.

Microsoft 2014b. Recursive and iterative queries. WWW-document.
<http://technet.microsoft.com/en-us/library/cc961401.aspx/>. No updating information.
Referred 18.4.2014.

OpenSSL Documents 2014. OpenSSL Documents. WWW-document.
<https://www.openssl.org/docs>. No updating information. Referred 11.5.2014.

RFC_2216 1999. The TLS Protocol Version 1.0. WWW-document.
<http://tools.ietf.org/html/rfc2246> Updated 01.1999. Referred 12.5.2014.

Rouse, Margaret 2006. PKI (Public key infrastructure). WWW-document.
<http://searchsecurity.techtarget.com/definition/PKI/>. Updated 10.2006. Referred
11.2.2014.

Rouse, Margaret 2013. Digital certificate. WWW-document.
<http://searchsecurity.techtarget.com/definition/digital-certificate/>. Updated 11.2013.
Referred 4.2.2014.

Rubayat, Hasan 2010. Setting up Apache HTTP/SSL on Windows. WWW-document.
<http://rubayathasan.com/tutorial/apache-ssl-on-windows/>. Updated 2.4.2010. Referred
11.5.2014.

Schaefer, Ken (ed.) 2013. Professional Microsoft IIS 8. Indianapolis: John Wiley & Sons Inc.

Sosinsky, Barrie 2009. Networking Bible. Indianapolis: Wiley Publishing, Inc.

Symantec 2011. HTTPS. WWW-document.

http://www.symantec.com/business/support/index?page=content&pmv=print&impressions=&viewlocale=es_ES&id=GL1377. Updated 15.8.2011. Referred 5.4.2014.

TIM 2014. SSL Pulse. WWW-document. <https://www.trustworthyinternet.org/ssl-pulse/>. Updated 4.3.2014. Referred 5.4.2014.

Tutorialspoint 2014. HTTP – Methods. WWW-document.

http://www.tutorialspoint.com/http/http_methods.htm. No updating information. Referred 28.2.2014.

Vyronis, Panayotis 2013. Explaining public-key cryptography to non-geeks. WWW-document. <https://medium.com/how-to-use-the-internet/f0994b3c2d5>. Updated 22.12.2013. Referred 11.2.2014.

w3 2004. Security Considerations. WWW-document.

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html>. Updated 1.9.2004. Referred 15.3.2014.

wiseGEEK 2014. How big is the Internet? WWW-document.

<http://www.wisegeek.org/how-big-is-the-internet.htm>. Updated 18.3.2013. Referred 8.4.2014.

HTTP Methods (Tutorialspoint 2014)

HTTP Method	
Request	Server response
CONNECT	
CONNECT www.tutorialspoint.com HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)	HTTP/1.1 200 Connection established Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32)
DELETE	
DELETE /hello.htm HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT) Host: www.tutorialspoint.com Accept-Language: en-us Connection: Keep-Alive	HTTP/1.1 200 OK Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) Content-type: text/html Content-length: 30 Connection: Closed <html> <body>

APPENDIX 1 (2)

	<pre><h1>URL deleted.</h1> </body> </html></pre>
GET	
<pre>GET /hello.htm HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT) Host: www.tutorialspoint.com Accept-Language: en-us Accept-Encoding: gzip, deflate Connection: Keep-Alive</pre>	<pre>HTTP/1.1 200 OK Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT ETag: "34aa387-d-1568eb00" Vary: Authorization,Accept Accept-Ranges: bytes Content-Length: 88 Content-Type: text/html Connection: Closed <html> <body> <h1>Hello, World!</h1> </body></pre>

	</html>
HEAD	
<pre>HEAD /hello.htm HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT) Host: www.tutorialspoint.com Accept-Language: en-us Accept-Encoding: gzip, deflate Connection: Keep-Alive</pre>	<pre>HTTP/1.1 200 OK Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT ETag: "34aa387-d-1568eb00" Vary: Authorization,Accept Accept-Ranges: bytes Content-Length: 88 Content-Type: text/html Connection: Closed</pre>
OPTIONS	
<pre>OPTIONS * HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)</pre>	<pre>HTTP/1.1 200 OK Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) Allow: GET, HEAD, POST, OPTIONS, TRACE</pre>

	Content-Type: httpd/unix-directory
POST	
<pre> POST /cgi-bin/process.cgi HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT) Host: www.tutorialspoint.com Content-Type: text/xml; charset=utf-8 Content-Length: 88 Accept-Language: en-us Accept-Encoding: gzip, deflate Connection: Keep-Alive <?xml version="1.0" encoding="utf-8"?> <string xmlns="http://clearforest.co m/">string</string> </pre>	<pre> HTTP/1.1 200 OK Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT ETag: "34aa387-d-1568eb00" Vary: Authorization,Accept Accept-Ranges: bytes Content-Length: 88 Content-Type: text/html Connection: Closed <html> <body> <h1>Request Processed Successfully</h1> </body> </html> </pre>

PUT	
<pre> PUT /hello.htm HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT) Host: www.tutorialspoint.com Accept-Language: en-us Connection: Keep-Alive Content-type: text/html Content-Length: 182 <html> <body> <h1>Hello, World!</h1> </body> </html> </pre>	<pre> HTTP/1.1 201 Created Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) Content-type: text/html Content-length: 30 Connection: Closed <html> <body> <h1>The file was created.</h1> </body> </html> </pre>
TRACE	
<pre> TRACE / HTTP/1.1 Host: www.tutorialspoint.com User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT) </pre>	<pre> HTTP/1.1 200 OK Date: Mon, 27 Jul 2009 12:28:53 GMT Server: Apache/2.2.14 (Win32) </pre>

APPENDIX 1 (6)

	<p>Connection: close</p> <p>Content-Type: message/http</p> <p>Content-Length: 39</p> <p>TRACE / HTTP/1.1</p> <p>Host:</p> <p>www.tutorialspoint.com</p> <p>User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)</p>
--	--

List of required commands for creating self-signed certificate in OpenSSL

*C:\Program Files\Apache Software Foundation\Apache2.2\bin>openssl req -new -out
server.csr*

Loading 'screen' into random state - done

Generating a 1024 bit RSA private key

.....++++++

.....++++++

writing new private key to 'privkey.pem'

Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:

*You are about to be asked to enter information that will be incorporated into your
certificate request.*

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:FI

State or Province Name (full name) [Some-State]:Etelä-Savo

Locality Name (eg, city) []:Mikkeli

Organization Name (eg, company) [Internet Widgits Pty Ltd]:MAMK

Organizational Unit Name (eg, section) []:IT

Common Name (e.g. server FQDN or YOUR name) []:www.mywebpage.fi

Email Address []:

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:

```
C:\Program Files\Apache Software Foundation\Apache2.2\bin>openssl rsa -in  
privkey.pem -out server.key
```

Enter pass phrase for privkey.pem:

writing RSA key

```
C:\Program Files\Apache Software Foundation\Apache2.2\bin>openssl x509 -in  
server.csr -out server.crt -req -signkey server.key -days 365
```

Loading 'screen' into random state - done

Signature ok

subject=/C=FI/ST=Etelx84-

Savo/L=Mikkeli/O=MAMK/OU=IT/CN=www.mywebpage.fi

Getting Private key