

**José López Luque**

# **Comparison of Different Ways to Avoid Internet Traffic Interception**

**Trondheim, September 2010**





**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Comparison of Different Ways to Avoid Internet Traffic Interception

José López Luque

**Master's thesis**

Submission date: September 2010

Professor: Stig F. Mjølsnes

Supervisor: Svein Y. Willassen

Norwegian University of Science and Technology  
Department of Telematic Engineering

# Problem Description

The main objective of this thesis is to analyze and compare different ways to avoid the Internet traffic eavesdropping by the governments (we have the Swedish example with the FRA-Law, but is also extensible to other countries, like China).

It will begin with a general overview of the countries involved, their interception laws, the main reasons that motivated these laws and their social implications.

The further analysis will consist on a description of the technologies involved in each option as well as the difficulties to implement them and the technical knowledge of the users in order to take profit of them.

Assignment given: 27. January 2010

Professor: Stig F. Mjølsnes

To my family, with love

# Abstract

As telephone conversations have moved to the Internet, so have those who want to listen in. But the technology needed to do so would entail a dangerous expansion of the government's surveillance powers. Internet users are watching how their privacy is slowly being undermined with justifications of national security and anti-terrorist purposes.

This surveillance is justified or not, users must know what options are available to them to protect their privacy.

The main objective of this thesis is to analyze and compare different ways to avoid the Internet traffic eavesdropping (carried out both by governments or malicious particulars).

The analysis consists on a description of the different protocols and technologies involved in each option as well as the difficulties to implement them and the technical knowledge of the users in order to take profit of them.

The conclusions state that with nowadays tools achieving a high security level is possible, even for non-professionals users.

# Contents

*Problem Description*

*Abstract*

## *Chapter 1 Introduction*

1.1	Background	8
1.2	Problem Definition	9
1.3	Objectives of the Study	9
1.4	Contribution of Thesis Work	10

## *Chapter 2 Interception Laws and Programs*

2.1	Introduction	11
2.2	European Union	11
2.3	United States	12
2.4	China	13

## *Chapter 3 Source Routing*

3.1	Introduction	15
3.2	Description	16
3.3	Security Drawbacks	19
	3.3.1 IP Spoofing	19
	3.3.2 DoS Attack	20
3.4	Conclusion	21

## *Chapter 4 Secure Transport protocols*

4.1	Introduction	23
	4.1.1. Public Key Cryptography Overview	23
4.2	Transport Layer Security (TLS)	25
	4.2.1 Description	25
	4.2.2 Advantages	27
	4.2.3 Drawbacks	28
4.3	Secure Shell (SSH)	29
	4.3.1 Description	29
	4.3.2 Advantages	30
	4.3.3 Drawbacks	30
4.4	Internet Protocol Security (IPsec)	31
	4.4.1 Description	31
	4.4.2 Advantages	32
	4.4.3 Drawbacks	32
4.5	Virtual Private Network (VPN)	34

## *Chapter 5 Traffic Analysis. Anonymity*

5.1	Introduction	35
5.2	Onion Routing	36
	5.2.1 Description	36
	5.2.2 Weaknesses	37
	5.2.3 TOR	37

## *Chapter 6 Surfing the Web*

6.1	HTTPS Description	39
6.2	TLS Cipher Suite Negotiation Analysis	40

## *Chapter 7 E-mail Security*

7.1	Introduction	43
7.2	Secure MIME	45
7.3	PGP/MIME and OpenPGP	46
7.4	SMTP Over TLS. E-mail Servers Analysis	48

## *Chapter 8 Instant Messaging*

8.1	Introduction	55
8.2	AOL Instant Messaging (AIM) – OSCAR Protocol	57
8.3	GTalk – XMPP	58
8.4	Skype	60
8.5	Windows Live Messenger – Microsoft Notification Protocol	63
8.6	Yahoo! Messenger	64

## *Conclusion*

## *References*



# **Chapter 1**

## **Introduction**

### **1.1 Background**

As long as people have been in private conversations, eavesdroppers have tried to listen them. When important issues were discussed in parlors, people tried to slip in to hear what was being said. When conversations moved to telephones, the wires were tapped. And nowadays that so much human activity takes place in cyberspace, spies have infiltrated that area as well.

Unlike earlier, physical frontiers, cyberspace is an artificial construct. The rules, designs and investments we make in cyberspace will determine the ways espionage, privacy and security will interact. This is the reason why nowadays there is a clear movement to give intelligence agencies all over the world a privileged position, building in the capacity of authorities to intercept cyberspace communications. The advantages of this trend for fighting crime and terrorism are obvious.

However, there is big controversy now about whether these intelligence activities that allow governments to intercept electronic communications are fair since they truncate the basic privacy rights of the users. One of the best examples of this is the Swedish “FRA-Law”, that authorizes the state to warrantlessly wiretap all telephone and Internet traffic that crosses Sweden's borders, affecting not only to the Swedish citizens but also all the people in the neighbouring countries.

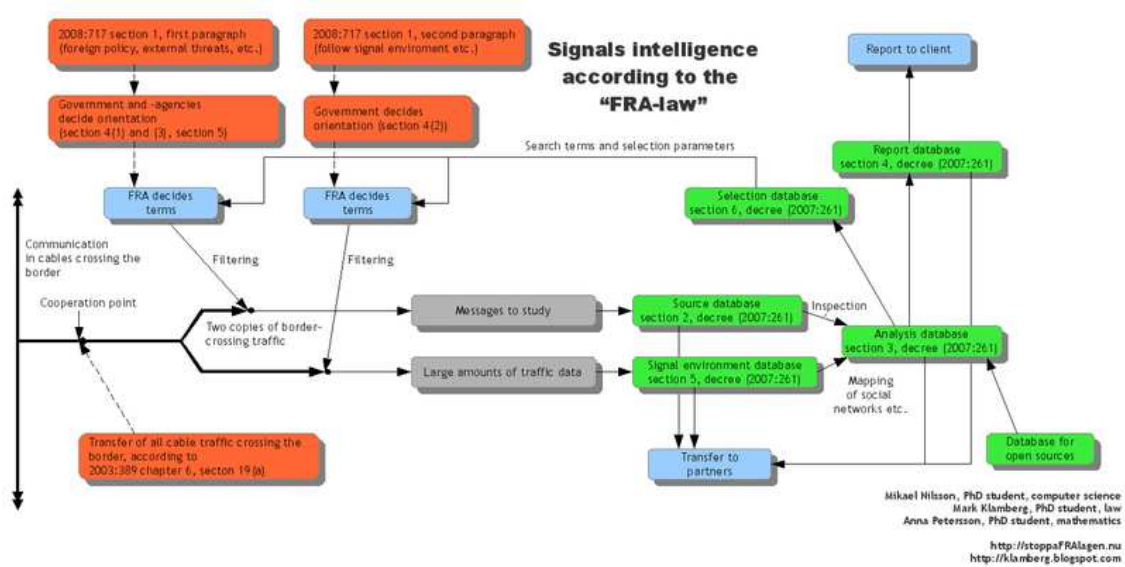


Figure 1.1 - Description how the Swedish Defense Radio Authority (FRA) collects and processes communication.

## 1.2 Problem Definition

Which is the best way to protect our communications? Can we trust the companies offering us “secure” communication software clients? It is strictly necessary that our interlocutor has the same software?

In order to answer these questions we will have to analyze the secure protocols used on the Internet and how the software clients implement them.

## 1.3 Objectives of the Study

The thesis is basically focused on the analysis of the most common techniques used to avoid Internet eavesdropping. The objectives of the study are mentioned below:

- Understanding of the main secure protocols used on the Internet.
- Understanding and avoiding traffic analysis.

- Verify the supposed security level of some e-mail servers.
- Compare different Instant Messaging clients and their protocols.
- Determine the best security option for end-users.

Above all, it is important to note that we will only focus on the security of the data once it is on the Internet, without considering other options such as, for example, our computer infected by a virus which obtains sensitive information and send it to a third party.

## **1.4 Contribution of Thesis Work**

The study concludes that with the available tools end users with security concerns can protect themselves of being eavesdropped and also from other techniques of obtaining information such as traffic analysis in a very easy way.

They only need to choose for their communications the clients (browsers, e-mail clients, etc...) that already work over secure protocols like TLS.

If more security is needed (for instance when using e-mail servers not supporting TLS) they can also use some encryption software like PGP, but then they need to share or publish their cryptographic keys and the same encryption software is needed in both computers.

## Chapter 2

### Interception Laws and Programs

#### 2.1 Introduction

Say the word *privacy* aloud, and you can start a lot of passionate discussions. One person cares about governmental abuse of power; another blushes about his drug use history; a third one complains about how corporations collect private data to target their ads or how insurance companies look for personal medical records to deny coverage to certain people. Some fear a world of increasing commercialization, in which data is used to sort everyone into one or another “market segment”, the best to exploit their most intimate whims.

Because of that, a regulation of data interception and retention is needed, but these regulations seem to be not enough for the privacy protection. We are going now to see the main surveillance programs running in the European Union and United States, as well as China, one country “famous” because of the violation of the privacy rights.

#### 2.2 European Union

The European Union Data Retention Directive (12-15-2005) <sup>[14]</sup> requires telecommunication operators to implement mass surveillance of the general public through retention of metadata (data providing information about one or more other pieces of data) on telecommunications and to keep the collected data at the disposal of various governmental bodies for substantially long times. Access to this information is not required to be limited to investigation of serious crimes, nor is a warrant required for access.

The main mass surveillance activities funded by the European Commission, in association with industrial partners, are HIDE and INDECT.

The consortium HIDE (Homeland Security, Biometric Identification & Personal Detection Ethics) is devoted to the pro-active surveillance system to detect potential abnormal behaviour in crowded spaces, and not related to the topic of this thesis.

But the one referred to telecommunication monitoring, among other activities is the INDECT project <sup>[17]</sup> (Intelligent information system supporting observation, searching and detection for security of citizens in urban environment). It develops an intelligent urban environment observation system to register and exchange operational data for the automatic detection, recognition and intelligent processing of all information of abnormal behaviour or violence. The main objectives of INDECT project are:

- Implementation of a distributed computer system that is capable of acquisition, storage and effective sharing on demand of the data.
- Devices used for mobile object tracking.
- A search engine for fast detection of persons and documents based on watermarking technology used for semantic search.
- Agents assigned to continuous and automatic monitoring of public resources such as closed-circuit television, websites, internet forums, net newsgroups, file servers, P2P networks and individual computer systems.

## 2.3 United States

In 1999 two models of mandatory data retention were suggested for the US. The first one consisted in saving the IP address assigned to a customer at a specific time. In the second model, which is closer to what Europe adopted, telephone numbers dialed, contents of Web pages visited, and recipients of e-mail messages must be retained by the ISP for an unspecified amount of time.

Actually the Communications Assistance for Law Enforcement Act (CALEA) <sup>[15]</sup> requires that all U.S. telecommunications companies modify their equipment to allow easy wiretapping of telephone, VoIP, and broadband Internet traffic.

Besides, millions of dollars per year are spent, by agencies such as the Information Awareness Office (with the "Total Information Awareness" program, later renamed "Terrorism Information Awareness" after a negative public reaction), NSA, and the FBI, to develop, implement, and operate systems such as Carnivore, ECHELON, and NarusInsight to intercept and analyze the immense amount of data that cross the Internet and telephone system every day (for more information about these system see <sup>[18][19]</sup>).

The FBI also developed the computer programs "Magic Lantern" and CIPAV <sup>[20][21]</sup>, which can be remotely installed on a computer system, in order to monitor a person's computer activity.

Also the telecom giant AT&T was accused by The Electronic Frontier Foundation, and has an ongoing lawsuit for its assistance of the U.S. government in monitoring the communications of millions of American citizens.

## 2.4 China

When talking about Internet and China people always think about censorship. It is true that censorship is the major problem there, but it is out of the scope of this thesis. We will focus, like in the other cases, in the privacy rights.

Privacy rights have been available to Chinese citizens under the Constitution and other legal regulations since the 1980's. However, due to the size and strength of government, the laws have not been applied to a great extent. The legislature is in the process of developing wider privacy rights under a Civil Code. These new rights have the potential to shift privacy power towards the citizens for the first time since the founding of the Communist Party of China.

The personal dignity of citizens of the People's Republic of China is recognized and protected under Article 38 of the Constitution <sup>[16]</sup>. Further, the freedom and privacy of correspondence of citizens of China are protected; however Article 40

provides some significant limitations to such rights – where state secrets or a criminal investigation is involved, police and other authorities can intercept communications if necessary. The expansive concept of “state secret” gives the government enough power to review and monitoring communications.

## Chapter 3

### Source Routing

#### 3.1 Introduction

One of the main keys in choosing or developing a tool for avoiding data interception is the difference between the software the sender and the receiver of this data have. We can not assume, for instance, that if the sender is using some kind of encryption software, the receiver is using the same.

This was the basis of the idea that only the sender should control the security of the transmission, regardless the receiver's equipment.

Instead of encrypting the communication, the only thing we had to do was routing the packets avoiding the points of interception, in other words, do not let the data cross the borders of the neighbouring country who is eavesdropping the Internet traffic, and source routing appeared to be the solution to that.

In the next two sections we will first describe the basics of source routing and then comment some security problems that should be taken into account since they are key in the evaluation of this mechanism.



## 3.2 Description

Source Routing is a technique whereby the sender of a packet can specify the route that this packet should take through the network.

The path information is placed in the packet. When the packet arrives at a switching device, no forwarding decision is necessary. The router looks at the path information in the packet to determine the port on which it should forward the packet. This is the opposite of hop-by-hop IP routing, where packets contain only the destination address and routers at each intersection in the network determine the best path to forward the packet.

Source routing assumes that the sender knows about the topology of the network, and hence can specify a path. However, it is not always possible to expect end-user or end-user's systems to know or learn a network's topology. This gets more difficult as the network grows, and is nearly impossible on the Internet where different provider networks are joined together. From a security point of view, as we will see in the next section, it is unwise to allow the sender to control the path of packets through the network.

There are two main types of source routing: **Strict Source Record Route (SSRR)** and **Loose Source Record Route (LSRR)**. In strict source routing, the sender specifies the *exact* route the packet must take, whereas in LSRR the sender just gives one or more hops that the packet must go through.

For a more careful description, we should take a look into the IP header (Fig. 2.1). The *Options* provided for control functions are necessary or useful in some situations but unnecessary for the most common communications. These options include provisions for timestamps, security, and special routing.

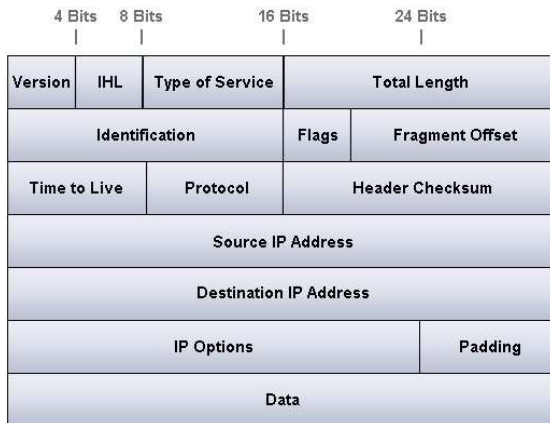


Figure 3.1 – IP Datagram Diagram

The options may appear or not in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation<sup>[1]</sup>.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option:

- Case 1: A single octet of option-type.
- Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

The option-type octet is viewed as having 3 fields: 1 bit copied flag, 2 bits option class, 5 bits option number.

For source routing, the following internet options are defined:

CLASS	NUMBER	LENGTH	DESCRIPTION
0	3	Var.	Loose Source Routing. Used to route the internet datagram based on information supplied by the source
0	9	Var.	Strict Source Routing. Used to route the internet datagram based on information supplied by the source.

Table 3.1 – Source routing options.

Both LSRR and SSRR options provide a means for the source of an Internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The options begin with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the recorded route full) and the routing is to be based on the destination address field.

If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and pointer is increased by four.

The recorded route address is the internet module's own internet address as known in the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in the reverse of the order it must be in to be used as a source route) means the option (and the IP header as a whole) remains a constant length as the datagram progresses through the internet.

## 3.3 Security Drawbacks

Source routing is a legitimate activity in some cases. For example, it can be used to discover the IP addresses of routers within a network. However, it also has the potential for misuse.

A malicious user could use source routing to learn more about a network that he or she is targeting for attack. Data packets contain information about where they have been and what machines they have transited. A malicious user might send data into a network in order to collect information about the network's topology. If he or she can perform source routing, they can probe the network more effectively by forcing packets into specific parts of the network.

### 3.3.1 IP Spoofing

Source routing also enables certain types of attacks. For instance, suppose an attacker is unable to attack "A" because it has a well-configured firewall, but learns that "B", which has no firewall, is allowed to directly connect to "A" behind its firewall. Source routing would allow the attacker to direct packets to "A" via "B", avoiding the firewall.

Another target of the attacker can be machines on private internet addresses such as 192.168.0.1. They are not normally accessible from the internet. If there is a machine on a private network that performs routing and traffic may be routed through it between two other networks, it may be possible for the attacker to specify their data to go through the machine on the private network. He may also fool the machine on the private network into believing it is some other computer using IP spoofing.

### 3.3.2 DoS Attack

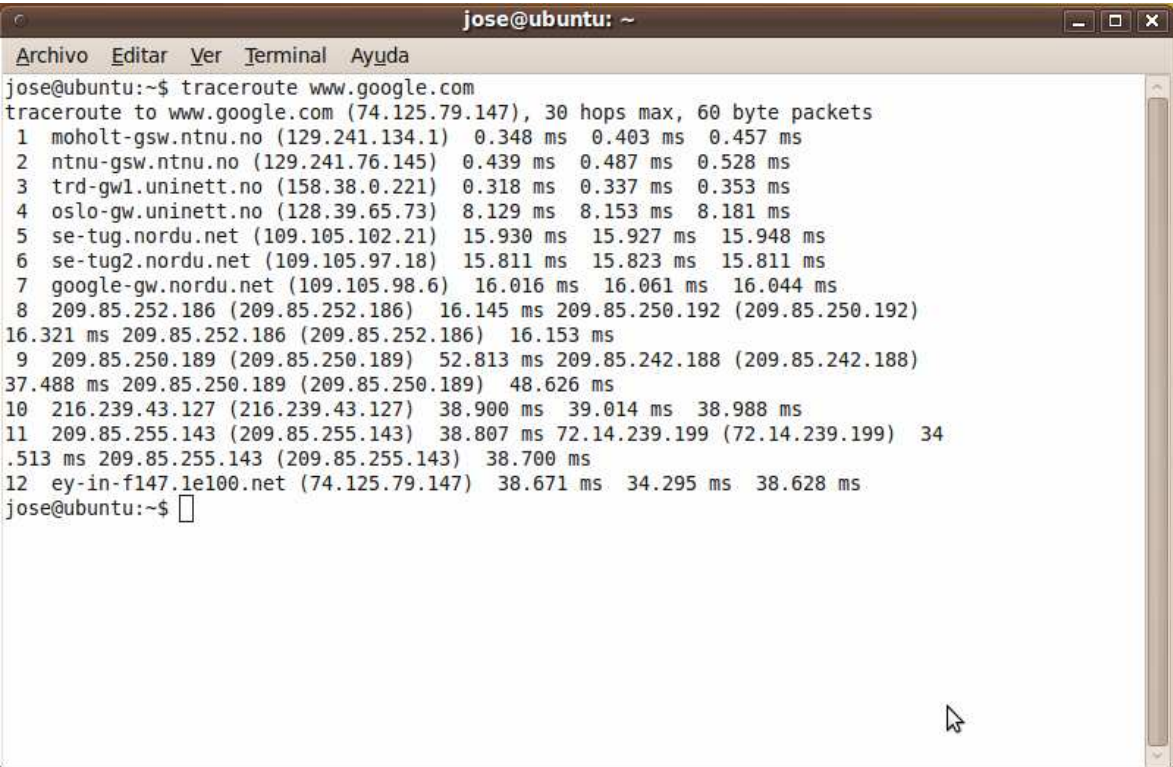
Besides spoofing source addresses for false authentication, attackers can also spoof their own source addresses in attacks where reply packets are not important. Any network-based Denial of Service (DoS) attack fits this description because the point of the attack is not to get a response but instead to flood the target with requests <sup>[2]</sup>.

In DoS attacks, it actually makes more sense for the attacker to spoof the source address, otherwise the attacker might wind up blocking his or her own access to the network. Spoofing source addresses also makes tracking the attack much more difficult, as the packets themselves must be traced on each network and subnet, back to the source.

Source address spoofing requires root access on UNIX systems. The attacker must have root access so that the attack software can open a "raw" network socket. Most applications use "cooked" sockets, in which the IP stack provides the necessary packet headers. A raw socket means that the application must prepare the necessary headers itself—that is, do its own cooking. This permits the attacker to put any information he or she wants in the headers, including spoofed source addresses.

## 3.4 Conclusion

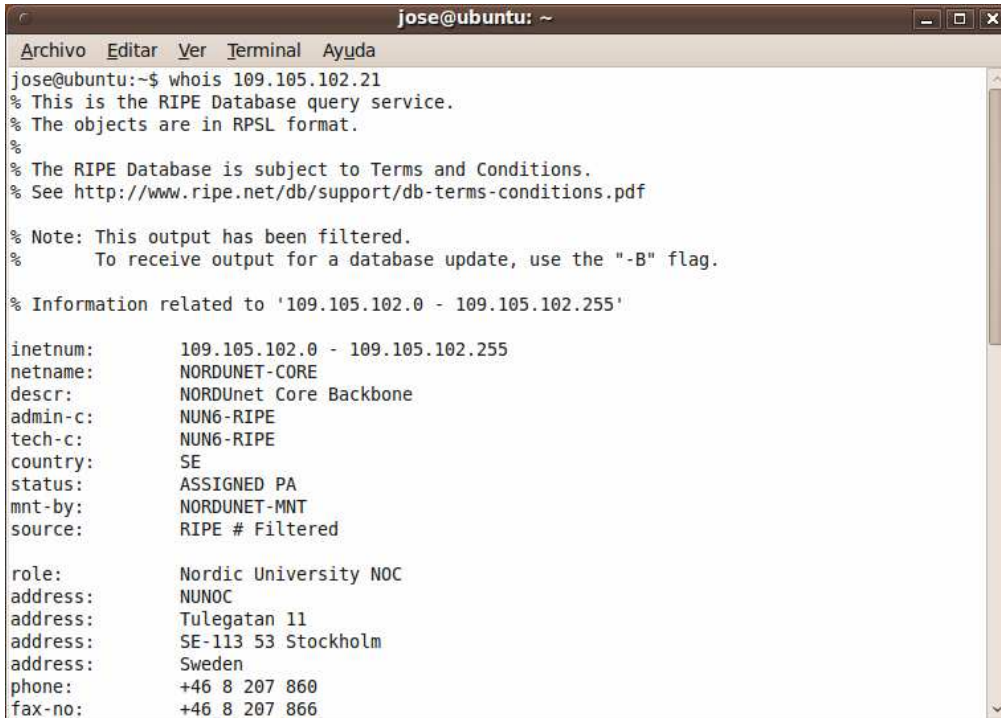
A little experiment was done. The first part consisted in tracing the route a packet follows to a destination (in this case [www.google.com](http://www.google.com)):



```
jose@ubuntu: ~  
Archivo Editar Ver Terminal Ayuda  
jose@ubuntu:~$ traceroute www.google.com  
traceroute to www.google.com (74.125.79.147), 30 hops max, 60 byte packets  
1  moholt-gsw.ntnu.no (129.241.134.1)  0.348 ms  0.403 ms  0.457 ms  
2  ntnu-gsw.ntnu.no (129.241.76.145)  0.439 ms  0.487 ms  0.528 ms  
3  trd-gw1.uninett.no (158.38.0.221)  0.318 ms  0.337 ms  0.353 ms  
4  oslo-gw.uninett.no (128.39.65.73)  8.129 ms  8.153 ms  8.181 ms  
5  se-tug.nordu.net (109.105.102.21)  15.930 ms  15.927 ms  15.948 ms  
6  se-tug2.nordu.net (109.105.97.18)  15.811 ms  15.823 ms  15.811 ms  
7  google-gw.nordu.net (109.105.98.6)  16.016 ms  16.061 ms  16.044 ms  
8  209.85.252.186 (209.85.252.186)  16.145 ms  209.85.250.192 (209.85.250.192)  
16.321 ms  209.85.252.186 (209.85.252.186)  16.153 ms  
9  209.85.250.189 (209.85.250.189)  52.813 ms  209.85.242.188 (209.85.242.188)  
37.488 ms  209.85.250.189 (209.85.250.189)  48.626 ms  
10 216.239.43.127 (216.239.43.127)  38.900 ms  39.014 ms  38.988 ms  
11 209.85.255.143 (209.85.255.143)  38.807 ms  72.14.239.199 (72.14.239.199)  34  
.513 ms  209.85.255.143 (209.85.255.143)  38.700 ms  
12 ey-in-f147.1e100.net (74.125.79.147)  38.671 ms  34.295 ms  38.628 ms  
jose@ubuntu:~$
```

Figure 3.2 – Traceroute [www.google.com](http://www.google.com)

Later, using *whois* we can determine whether a router belongs to a country or not:



```

jose@ubuntu: ~
Archivo  Editar  Ver  Terminal  Ayuda
jose@ubuntu:~$ whois 109.105.102.21
% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See http://www.ripe.net/db/support/db-terms-conditions.pdf

% Note: This output has been filtered.
%       To receive output for a database update, use the "-B" flag.

% Information related to '109.105.102.0 - 109.105.102.255'

inetnum:        109.105.102.0 - 109.105.102.255
netname:        NORDUNET-CORE
descr:          NORDUnet Core Backbone
admin-c:        NUN6-RIPE
tech-c:         NUN6-RIPE
country:        SE
status:         ASSIGNED PA
mnt-by:         NORDUNET-MNT
source:         RIPE # Filtered

role:           Nordic University NOC
address:        NUNOC
address:        Tulegatan 11
address:        SE-113 53 Stockholm
address:        Sweden
phone:          +46 8 207 860
fax-no:         +46 8 207 866

```

Figure 3.3 - Whois

Then all we had to do was source routing the packet avoiding this router (or anyone suspicious to be in a country who is wiretapping the communications), but what was found out was that due to the security problems listed in the above section, it was highly recommended to the ISPs to disable the source routing option in their routers, and so they did <sup>[3]</sup>.

Even if we have a look on the traceroute command option to do the source routing (traceroute -g) we find:

#### **-g gateway**

*Tells traceroute to add an IP source routing option to the outgoing packet that tells the network to route the packet through the specified gateway. Not very useful, because most routers have disabled source routing for security reasons.*

Because of that we had to discard the source routing as a feasible way to avoid the eavesdropping.

## Chapter 4

### Secure Protocols

#### 4.1 Introduction

In cryptography, **encryption** is the process of transforming information (referred to as plaintext) using an algorithm to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information. In many contexts, the word *encryption* also implicitly refers to the reverse process, decryption (e.g. “software for encryption” can typically also perform decryption), to make the encrypted information readable.

##### 4.1.1 Public Key Cryptography Overview

Since the topic of this thesis is the security over the Internet, and in order to understand how the protocols involved on it work, we should first describe the basics of the cryptography used by these protocols, the public key cryptography.

The difference between public or asymmetric key and symmetric key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same as the key used to decrypt it. Each user has a pair of cryptographic keys, a public key and a private key. The private key is kept secret, while the public is distributed. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key. The keys are related mathematically <sup>[24]</sup>, but the private key cannot be derived from the public key.



The two main parts of public key cryptography are:

- **Public key encryption:** A message encrypted with the recipient's public key cannot be decrypted by anyone except the possessor of the matching private key (this will be the owner of that key and the person associated with the public key used). This is used for confidentiality.
- **Digital signatures:** A message signed with the sender's private key can be verified by anyone who has access to the sender's public key, proving that the sender had access to the private key (and therefore he has to be the person associated with the public key used).

A central problem for use of public-key cryptography is the need of being sure that a public key is correct, that it belongs to the person or entity claimed, and has not been tampered with or replaced by a malicious third party. The usual approach to this problem is to use a public key infrastructure (PKI), in which one or more third parties, known as certificate authorities, certify ownership of key pairs. Another approach, used by PGP, is the "web of trust" method to ensure authenticity of key pairs, in which identity certificates can be digitally signed by other users who, by that act, endorse the association of that public key with the person or entity listed in the certificate.

## 4.2 Transport Security Layer (TLS)

### 4.2.1 Description

TLS is an IETF standards protocol, last updated in RFC 5246, that was based on the earlier SSL specifications developed by Netscape Corporation.

The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications <sup>[4]</sup>. It is in widespread use in applications like web browsing, electronic mail, Internet faxing, instant messaging and voice over IP (VoIP).

The protocol is composed of two layers: the **TLS Record Protocol** and the **TLS Handshake Protocol**.

The TLS Record Protocol is at the lowest level, layered on top of some reliable transport protocol and provides connection security that has two basic properties:

- The connection is private. Symmetric cryptography is used for data encryption. The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The Record Protocol can also be used without encryption.
- The connection is reliable. Message transport includes a message integrity check using a keyed MAC (Message Authentication Code). Secure hash functions (like SHA-1) are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.

The TLS Handshake Protocol is encapsulated by the TLS Record Protocol and allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol

transmits or receives its first byte of data. The TLS Handshake Protocol provides connection security that has three basic properties:

- The peer's identity can be authenticated using asymmetric cryptography (like RSA). This authentication can be made optional, but is generally required for at least one of the peers.
- The negotiation of a shared secret is secure: the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by a man in the middle attacker.
- The negotiation is reliable: no attacker can modify the negotiation communication without being detected by the parties to the communication.

A TLS client and server negotiate a stateful connection by using a handshaking procedure. During this handshake, the client and server agree on various parameters used to establish the connection's security.

- The handshake begins when a client connects to a TLS-enabled server requesting a secure connection, and presents a list of supported Cipher Suites (ciphers and hash functions).
- Then, the server picks the strongest cipher and hash function that it also supports and notifies the decision to the client.
- The server sends back its identification in the form of a digital certificate (typically in the form of X.509 certificates, which define required fields and data formats). The certificate usually contains the server name, the trusted certificate authority (CA), and the server's public encryption key.
- The client may contact the trusted CA and confirm that the certificate is authentic before proceeding.
- In order to generate the session keys used for the secure connection, the client encrypts a random number with the server's public key and sends the result to the server. Only the server should be able to decrypt it (with

its private key). This is the one fact that makes the keys hidden from third parties, since only the server and the client have access to this data.

- From the random number, both parties generate key material for encryption and decryption.

This concludes the handshake and begins the secured connection, which is encrypted and decrypted with the key material until the connection closes.

If any one of the above steps fails, the TLS handshake fails, and the connection is not created.

### **4.2.2 Advantages**

When implementing security with TLS you place on top of the TCP/IP layers and substitute TCP calls with the TLS calls. It is independent of the applications once you have set up a connection, after the initiating handshake, it acts just like a secure tunnel and you can send anything through it.

A big advantage of TLS is that you don't need special software on the clients. That's because the TLS uses, for instance, the Web browser as the client application. This also means the protocols that can be handled by TLS are more limited. However, this can also be a security advantage. With TLS, instead of giving clients access to the whole network or subnet as with IPsec (we will talk about IPsec later), you can restrict them to specific applications.

There exist several implementation packages, both commercial and free. Implementation packages are available for several platforms as Linux, Windows and others. That means that many people and companies have done implementation of TLS. Many products as web servers (Apache) and browsers (Firefox, Internet Explorer ...) have support for TLS. That has lead to more and more people using TLS.

TLS has all available security functions we want to have to make the project

secure: authentication, session key exchange with asymmetric methods, encryption with symmetric methods, MAC, and certificates.

### **4.2.3 Drawbacks**

TLS is placed just on top of the TCP/IP layers, you substitute the TCP calls with the TLS calls and that means that the programmer of the implementation has to know a lot about the OS and its specific system calls.

Another significant drawback when implementing TLS is that cryptography, specifically public key operations, are CPU intensive. As a result, there is a performance penalty when using TLS. The penalty varies widely depending on how often connections are established and how long they last. The greatest overhead occurs while connections are being set up.

## 4.3 Secure Shell (SSH)

### 4.3.1 Description

**Secure Shell (SSH)** is a network protocol that allows data to be exchanged using a secure channel between two computers. The two major versions of the protocol are referred to as SSH1 and SSH2. SSH was designed as a replacement for Telnet and other insecure remote shells, which send information, including passwords, in plaintext, leaving them susceptible to packet analysis. The encryption used by SSH goal is to provide confidentiality and integrity of data over an insecure network, such as the Internet.

SSH uses public key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

It is typically used to log into a remote machine and execute commands, but it also supports tunneling, mapping TCP ports and X (network protocol that provides a graphical user interface) connections; it can transfer files using the associated SFTP or the earlier SCP protocols.

It can be used for many applications across many platforms. Some of them are:

- Login to a shell on a remote host (replacing Telnet).
- Secure file transfer.
- Executing a single command on a remote host (replacing rsh, a command line computer program that can execute shell commands as another user).
- In combination with rsync (software application for Unix systems which synchronizes files and directories from one location to another) to backup, copy and mirror files efficiently and securely.
- Forwarding or tunnelling a port (permitting access on the private LAN from the Internet).
- Using as a full-fledged encrypted VPN. Note that only OpenSSH server and client supports this feature.

- Forwarding X from a remote host (possible through multiple intermediate hosts).
- Browsing the web through an encrypted proxy connection with SSH clients that support SOCKS (a protocol that facilitates the routing of network packets between client–server applications via a proxy server).
- Securely mounting a directory on a remote server as a filesystem on a local computer using SSHFS (The client interacts with the remote file system via SFTP).
- For automated remote monitoring and management of servers through one or more of the mechanisms as discussed above.

The SSH protocol consists of three major components: the **Transport Layer Protocol**, which provides server authentication, confidentiality, and integrity with perfect forward secrecy, the **User Authentication Protocol**, which authenticates the client to the server, and finally, the **Connection Protocol**, which multiplexes the encrypted tunnel into several logical channels <sup>[4]</sup>.

### 4.3.2 Advantages

SSH never sends password in clear text. It provides encryption of TCP/IP streams from host to host and it also provides protection against DNS/IP spoofing attacks, where a machine inside a network of trusted hosts is made believe that an outside machine is an inside one.

### 4.3.3 Drawbacks

The main SSH security drawback consist in that it does not support certificates, digital signatures or MAC.

## 4.4 Internet Protocol Security (IPsec)

### 4.4.1 Description

Internet Protocol security (IPsec) is a set of open standards for protecting communications over IP networks using cryptographic security services. IPsec supports network-level client-server authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec is operating between the Internet and Transport layers of the Internet Protocol Suite. Some other Internet security systems in widespread use (such as the two described above) operate in the upper layers. Hence, IPsec can be used for protecting any application traffic across the Internet. Applications don't need to be specifically designed to use IPsec. The use of TLS, on the other hand, must typically be incorporated into the design of applications.

It supports two encryption modes: **Transport** and **Tunnel**. Transport mode encrypts only the data portion (*payload*) of each packet, but leaves the header untouched. The more secure Tunnel mode encrypts both the header and the payload.

Three main protocols are used <sup>[6]</sup>:

A **security association** protocol (SA) is set up by another protocol known as Internet Key Exchange (IKE) (which allows the receiver to obtain a public key and authenticate the sender using digital certificates) by handling negotiation of protocols and algorithms and to generate the encryption and authentication keys to be used by IPsec.

An **Authentication Header** protocol (AH) to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replay attacks.



And finally, an **Encapsulating Security Payload** protocol (ESP) to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality.

#### 4.4.2 Advantages

IP sec is universally applicable as it can protect a mixture of applications protocols running over a complex combination of data. It can provide security and communicate with different types of networks from all over the world.

IPSec is not limited to specific applications but is application independent. Whatever be the application the data will traverse the network, it will be routed by IP making it IPSec compatible.

#### 4.4.3 Drawbacks

IPSec is complex. It has a great many features and options. Choosing and setting an option is a bit difficult. Complexity also increases the probability of weaknesses being discovered.

Firewalls are preconfigured rules and IPSec encrypts these rules in the packet which defeats the purpose of a firewall. A solution for this could be firewall along with an IPSec gateway.

The security of IPSec is easily affected by weakness or vulnerability's in the specific methods for key exchange, in hashing or encryption algorithms. The DES encryption algorithm is now susceptible to brute-force attacks using readily available software and hardware. Brute force attacks are methods to decrypt data by simply trying every possible key value.

An IPSec gateway system needs to be secure if this is compromised then no system can be trusted if the underlying machine has been subverted.

IPSec can't provide the same end-end security as it is not working between users or applications but between machines.

Finally, encryption of small packets generates a large overhead. This diminishes network performance. It also requires that the client computers have client software installed.

## 4.5 Virtual Private Network (VPN)

We are going to see now one of the main applications of IPsec: Implementing a virtual private network.

A virtual private network (**VPN**) is a network that uses a public telecommunication infrastructure, such as the Internet, to provide remote offices or individual users with secure access to their organization's network. It aims to avoid an expensive system of owned or rented lines that can be used by only one organization. It provides the organization with the same secure capabilities but at a much lower cost.

It encapsulates data transfers between two or more networked devices not on the same private network so as to keep the transferred data private from other devices on one or more intervening local or wide area networks.

VPNs use cryptographic tunneling protocols to provide confidentiality, authentication and message integrity. The standard protocol is IPsec, but others as TLS, SSH, DTLS (Datagram TLS), MPPE (Microsoft Point-to-Point Encryption) or SSTP (Secure Socket Tunneling Protocol) can also be used.

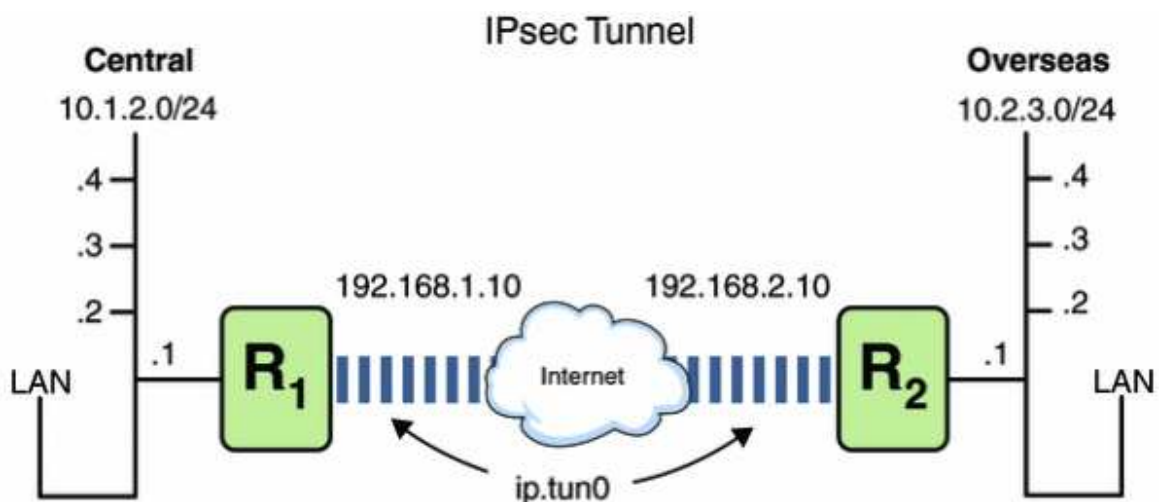


Figure 4.1 - Protecting a VPN with IPsec by Using Tunnels in Tunnel Mode

## Chapter 5

### Traffic Analysis. Anonymity

#### 5.1 Introduction

People normally think that if they use some of the protocols described in the previous chapter or some other encryption software their communication will be safe. This is not true, encryption provides a reliable way to avoid the “message” eavesdropping, but there is another way to find out a lot of information about the sender, this is called traffic analysis.

Traffic analysis is the process of intercepting and examining messages in order to obtain information from patterns in communication. It can be performed even when the messages are encrypted and cannot be decrypted. That's because it focuses on the header, which discloses source, destination, size, timing, and so on. In general, the greater the number of messages observed, or even intercepted and stored, the more can be inferred from the traffic.

It can be used to infer who is talking to whom over a public network. Knowing the source and destination of your Internet traffic allows others to track your behaviour and interests. This can impact your checkbook if, for instance, an e-commerce site uses price discrimination based on your country or institution of origin. It can even threaten your job and physical safety by revealing who and where you are. For example, if you're travelling abroad and you connect to your employer's computers to read or send mail, you can inadvertently reveal your national origin and professional affiliation to anyone observing the network.

A basic problem for the privacy is that the recipient of your communications can see that you sent it by looking at headers. So can authorized intermediaries like Internet service providers, and sometimes unauthorized intermediaries as well. A very simple form of traffic analysis might involve sitting somewhere between sender and recipient on the network, looking at headers.

In the next section we will introduce a technique used to avoid traffic analysis: Onion Routing.

## 5.2 Onion Routing

### 5.2.1 Description

Onion routing is a technique for anonymous communication over a computer network. It protects against traffic analysis.

In Onion Routing, messages are repeatedly encrypted and then sent through several network nodes called onion routers. Each onion router removes a layer of encryption to see the routing instructions, and sends the message to the next router where this is repeated, and route again the messages in an unpredictable path. This prevents the intermediary nodes from knowing the origin, destination, and contents of the message.

To create an onion, the router at the beginning of the transmission randomly selects a number of onion routers and generates a message for each one, providing it with symmetric keys for decrypting messages, and instructing it which router will be next in the path. Each of these messages, and the messages intended for subsequent routers, are encrypted with the corresponding router's public key. This provides the layered structure, in which it is necessary to decrypt all outer layers of the onion in order to reach an inner layer.

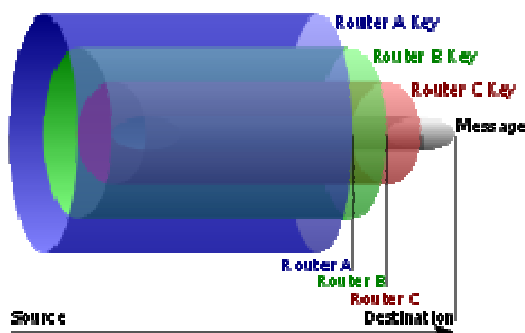


Figure 5.1 – Onion Layered Structure

Once the path has been specified, it remains active to transmit data for some period of time. While the path is active, the sender can transmit equal-length messages encrypted with the symmetric keys specified in the onion, and they will be delivered along the path. As the message leaves each router, it peels off

a layer using the router's symmetric key, and thus is not recognizable as the same message. The last router peels off the last layer and sends the message to the intended recipient.

### 5.2.2 Weaknesses

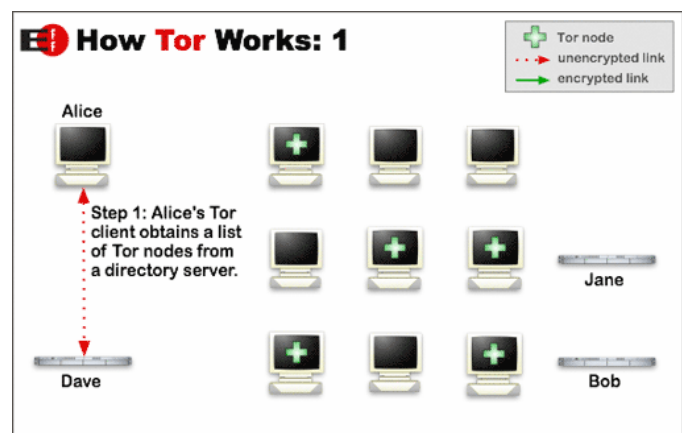
Onion routing does not provide perfect sender or receiver anonymity against all possible eavesdroppers: it is possible for a local eavesdropper to observe that an individual has sent or received a message. It does provide for a strong degree of *unlinkability* (the notion that an eavesdropper cannot easily determine both the sender and receiver of a given message). Even within these limits, onion routing does not provide any absolute guarantee of privacy; in general, the degree of privacy achieved is a function of the number of participating routers versus the number of compromised or malicious routers.

Moreover, onion routing exit nodes give the operator complete access to the content being transmitted (via sniffing) and therefore the onion network should not be used to transmit sensitive information without using end-to-end cryptography, such as TLS.

### 5.2.3 TOR

Tor is a free software implementation of onion routing.

As we have seen in the previous section, and now we illustrate with some figures, to create a private network path with Tor, the user's client incrementally builds a circuit of encrypted connections through the routers on the network. The circuit is extended one hop at a time, and each router along the way knows only which router gave it data and which router it is giving data to. No individual



router along the way knows only which router gave it data and which router it is giving data to. No individual

router ever knows the complete path that a data packet has taken. The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop can't trace these connections as they pass through.

Once a circuit has been established, many kinds of data can be exchanged and several different software applications can be deployed over the Tor network. Because each router sees no more than one hop in the circuit, neither an eavesdropper nor a compromised router can use traffic analysis to link the connection's source and destination.

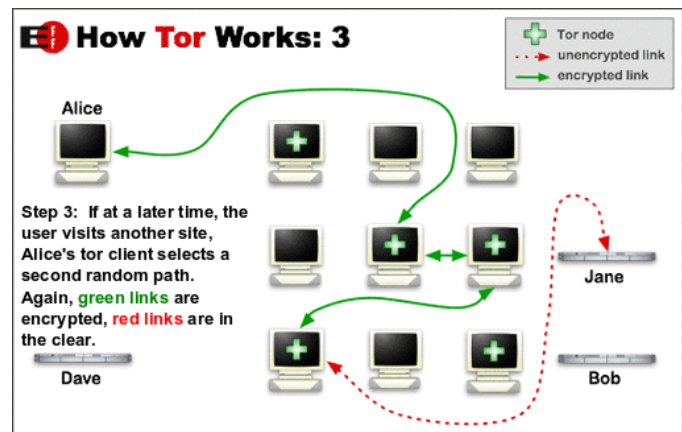
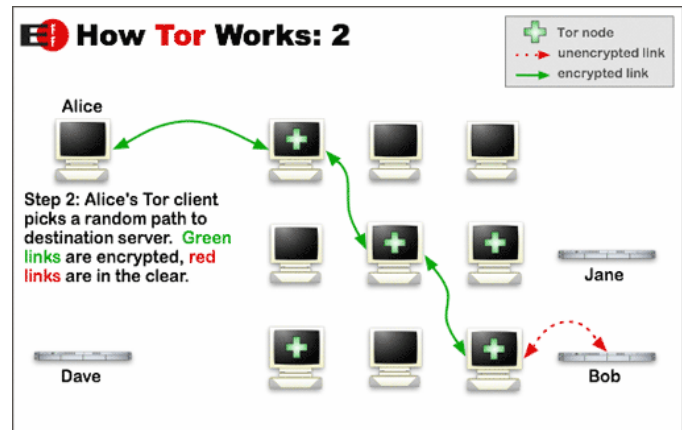


Figure 5.2 – TOR Operation

For efficiency, the Tor software uses the same circuit for connections that happen within the same ten minutes or so. Later requests are given a new circuit, to keep people from linking your earlier actions to the new ones.

## Chapter 6

### Surfing the Web

#### 6.1 HTTPS Description

One of the most used protocols when securing the web browsing is **HTTPS** (Hypertext Transfer Protocol Secure).

HTTP was originally used in the clear on the Internet. However, increased use of HTTP for sensitive applications has required security measures. SSL, and its successor TLS were designed to provide channel-oriented security.

HTTPS is a combination of the Hypertext Transfer Protocol with the TLS protocol to provide encryption and secure identification of the server <sup>[8]</sup>. HTTPS connections are often used for payment transactions on the World Wide Web and for sensitive transactions in corporate information systems.

HTTP operates at the highest layer of the OSI Model, the Application layer; but, as we have seen before, the security protocol operates at a lower sublayer, encrypting an HTTP message prior to transmission and decrypting a message upon arrival.

The HTTP client should also act as the TLS client. It should initiate a connection to the server on the appropriate port and then send the TLS ClientHello to begin the TLS handshake. When the TLS handshake has finished, the client may then initiate the first HTTP request. All HTTP data must be sent as TLS application data.

The first data that an HTTP server expects to receive from the client is the Request-Line production. The first data that a TLS server (and hence an HTTP/TLS server) expects to receive is the ClientHello. Consequently, a



common practice has been to run HTTP/TLS over a separate port in order to distinguish which protocol is being used. When HTTP/TLS is being run over a TCP/IP connection, the default port is 443. This does not exclude HTTP/TLS from being run over another transport, and TLS only presumes a reliable connection-oriented data stream <sup>[8]</sup>.

## 6.2 TLS Cipher Suite Negotiation Analysis

When your browser connects to a web site protected by transport layer security of some kind (usually by accessing an `https://` URL) there is a negotiation between the two parties. Each party (browser, server) comes to the negotiation with a list of *cipher suites* that it is prepared to use, and the result is that one of these suites is chosen for the connection.

Nowadays almost every browser support TLS, and we are going to see how the cipher suite negotiation is done. Specially, we are going to see it in Mozilla Firefox 3.6.8 when browsing the Google encrypted version web site (`https://www.google.com` or `https://encrypted.google.com`).

It is hard to figure out which cipher suites Firefox is prepared to use from its documentation <sup>[22]</sup>, so we decided to determine the answer directly by looking on the negotiation part of the protocol.

The easiest way we could find to look into a TLS connection is to use the Wireshark protocol analyser running on the client machine. All we need to do is start up Wireshark and tell it to capture packets going to or from the appropriate port at the server's IP address.

In Fig. 6.1 we can see the Client Hello part of the Handshake protocol sending to the server all the possible cipher suites the browser implements (the one in blue is the one the server will choose, as we will see in the next capture):

```

⊞ Internet Protocol, Src: 78.91.66.16 (78.91.66.16), Dst: 74.125.79.102
⊞ Transmission Control Protocol, Src Port: cuillamartin (1356), Dst Port
⊞ Secure Socket Layer
  ⊞ TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 380
  ⊞ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 376
    Version: TLS 1.0 (0x0301)
  ⊞ Random
    Session ID Length: 32
    Session ID: 6E206FD5861B43C243E7A540EF236B1AA596213DA638B847...
    Cipher Suites Length: 72
  ⊞ Cipher Suites (36 suites)
    Cipher Suite: Unknown (0x00ff)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x0088)
    Cipher Suite: TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA (0x0087)
    Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
    Cipher Suite: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
    Cipher Suite: TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xc00f)
    Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xc005)
    Cipher Suite: TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x0084)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    Cipher Suite: TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x0045)
    Cipher Suite: TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA (0x0044)
    Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
    Cipher Suite: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
    Cipher Suite: TLS_ECDH_RSA_WITH_RC4_128_SHA (0xc00c)
    Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xc00e)
    Cipher Suite: TLS_ECDH_ECDSA_WITH_RC4_128_SHA (0xc002)
    Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xc004)
    Cipher Suite: TLS_RSA_WITH_SEED_CBC_SHA (0x0096)
    Cipher Suite: TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x0041)
    Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
    Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008)
    Cipher Suite: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
    Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
    Cipher Suite: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x0013)

```

Figure 6.1 – Client Hello

In the next figure we see the sever reply with the chosen cipher suite:

```
⊞ Internet Protocol, Src: 74.125.79.102 (74.125.79.102), Dst: 78.91.66.16 (78.91.66.16)
⊞ Transmission Control Protocol, Src Port: https (443), Dst Port: editbench (1350), Seq: 1, Ack: 386, Len: 133
⊞ Secure Socket Layer
  ⊞ TLSv1 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 81
  ⊞ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 77
    Version: TLS 1.0 (0x0301)
  ⊞ Random
    Session ID Length: 32
    Session ID: 6E206FD5861B43C243E7A540EF236B1AA596213DA638B847...
    Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
    Compression Method: null (0)
    Extensions Length: 5
  ⊞ Extension: Unknown 65281
  ⊞ TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  ⊞ TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

Figure 6.2 – Server Hello

Finally, once the handshake is finished and the communication has started, we can see how the application data is encrypted:

```
⊞ Secure Socket Layer
  ⊞ TLSv1 Record Layer: Application Data Protocol: http
    Content Type: Application Data (23)
    Version: TLS 1.0 (0x0301)
    Length: 1117
    Encrypted Application Data: D042207CA07070EDE8802BAC73FCF1A5B6B30884CC4C3AC4...
```

Figure 6.3 – Encrypted Application Data

## Chapter 7

### E-mail

#### 7.1 Introduction

With the growth of the Internet in the last 10 years, e-mail has suffered a change on his use. It is no longer an internal messaging tool for companies, it has spread everywhere.

Probably e-mail eavesdropping is the biggest concern among the users. Every day millions of mails with private data are sent. So it is very important to secure them, however many of the users don't know most of their e-mail providers are not implementing any security measures at all.

Most Internet-based e-mail systems use a combination of three main protocols: SMTP (for message delivery) and POP and IMAP (for message retrieval). Of course, for proprietary systems, there are other, different, protocols that take the place of these standardized ones. Nevertheless, when it comes to pulling and pushing e-mail across the Internet, these three are the dominant ones.

At the ISP level, a level of protection can be implemented by encrypting the communication between servers themselves, usually employing TLS. In section 7.4 we will analyse how SMTP over TLS works. Also POP and IMAP connection can be secured with TLS.

Although many ISPs have implemented secure sending methods, users have been slow to adopt the habit, sometimes due to the esoteric nature of the encryption process. Without user participation, e-mail is only protected intermittently from intrusion.

To provide a reasonable level of privacy, all routers in the e-mail pathway, and all connections between them, must be secured. This is done by data encryption.

An industry-wide push toward regular encryption of e-mail correspondence is slow in the making. However, there are certain standards that are already in place which some services have begun to employ. In the next two following sections we are going to describe two standards that provide that encryption: PGP and S/MIME.

Finally, as a curiosity, we also can find some other methods to secure our e-mail. Even maybe they are not very practical or secure it is worth to talk about them.

The first method is to send an open message to the recipient which contains no sensitive content but which announces a message waiting for the recipient on the sender's secure mail facility. The recipient then follows a link to the sender's secure website where the recipient must log in with a username and password before being allowed to view the message.

The second one is e-mail jamming, the use of sensitive words in e-mails to jam the authorities that listen in on them by providing a form to divert attention away and an intentional annoyance. It is used by some civil rights activists in an attempt to frustrate government spy networks. Activists deliberately include "sensitive" words and sentences in otherwise innocuous emails to ensure that these are picked up by the monitoring systems. The theory is that the senders of these e-mails will eventually be added to a "harmless" list and their emails no longer intercepted, thus allowing them to regain some privacy.

## 7.2 Secure MIME

S/MIME (Secure / Multipurpose Internet Mail Extensions) is a standard for public key encryption and signing of e-mail encapsulated in MIME.

(MIME is an Internet standard that extends the format of e-mail to support text in character sets other than ASCII, non-text attachments, message bodies with multiple parts and header information in non-ASCII character sets).

S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures) and privacy and data security (using encryption). S/MIME specifies the application/pkcs7-mime (smime-type "enveloped-data") type for data enveloping where the whole MIME entity to be enveloped is encrypted and packed into an object which later is inserted into an application/pkcs7-mime MIME entity<sup>[10]</sup>.

But there are some obstacles to deploying S/MIME in practice:

First of all, not all e-mail software handles S/MIME signatures, resulting in an attachment called *smime.p7s* that may confuse some people.

Also S/MIME is sometimes considered not properly suited for use via webmail clients. Though support can be placed into a browser, some security practices require the private key to be kept accessible to the user but inaccessible from the webmail server, complicating the webmail advantage of providing ubiquitous accessibility. This issue is not fully specific to S/MIME - other secure methods of signing webmail may also require a browser to execute code to produce the signature, exceptions are PGP Desktop and versions of GnuPGP, who will grab the data out of the webmail, sign it by means of a clipboard, and put the signed data back into the webmail page. Seen from the view of security this is even the more secure solution.

Some organizations consider it acceptable for webmail servers to be "in on the secrets"; others don't.

S/MIME functionality is built into the majority of modern e-mail software and interoperates between them, the most known ones are:

- Alpine
- Claws Mail
- eM client
- Eudora
- IBM Lotus Notes
- Microsoft Office Outlook
- Mozilla Thunderbird
- Mutt

### 7.3 PGP/MIME and OpenPGP

PGP/MIME and OpenPGP are standards based on Pretty Good Privacy.

Pretty Good Privacy (**PGP**), created by Philip Zimmermann in 1991, is a data encryption and decryption software that provides cryptographic privacy and authentication for data communication. PGP is usually used for signing, encrypting and decrypting e-mails.

PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and, finally, public-key cryptography. Each step uses one of several supported algorithms. Each public key is bound to a user name and/or an e-mail address.

Nowadays, there is no known method which will allow to break PGP encryption by cryptographic or computational means. Early versions of PGP have been found to have theoretical vulnerabilities and so current versions are recommended. In addition to protecting data in transit over a network, PGP encryption can also be used to protect data in long-term data storage such as disk files.

The cryptographic security of PGP encryption depends on the assumption that the algorithms used are unbreakable by direct cryptanalysis with current equipment and techniques. For instance, in the original version, the RSA algorithm was used to encrypt session keys; RSA's security depends upon the one-way function nature of mathematical integer factoring. As current versions of PGP have added additional encryption algorithms, the degree of their cryptographic vulnerability varies with the algorithm used. In practice, each of the algorithms in current use is not publicly known to have cryptanalytic weaknesses.

However, PGP is still not suitable for fully transparent e-mail encryption. The main missing feature is the lack of MIME integration. Thus, PGP is not suitable for multimedia types other than ASCII text. PGP does contain some support for 8-bit char sets, but at cross-purposes with MIME. Signature checking of non-ASCII data is simply not reliable.

These are the main reasons why the PGP standard evolved to PGP/MIME and OpenPGP, both supporting MIME standard.

Some of the clients supporting PGP/MIME or OpenPGP are:

- Claws Mail
- Eudora (even though it requires a plugin)
- Gnus
- Kmail
- IBM Lotus Notes
- Mozilla Thunderbird
- Mutt



## 7.4 SMTP Over TLS. E-mail Servers Analysis

The protocol adopted to transmit the mail was SMTP (Simple Mail Transfer Protocol). The SMTP standard allows the exchange of e-mails from different platforms. Historically, this protocol was designed to send short messages, without confidentiality in closed networks, and not primarily designed to transmit sensitive information through a global network like the Internet. SMTP carries messages in a way that anyone can read (clear text).

Because the Simple Mail Transfer Protocol (SMTP) standard sends e-mail without using encryption or authentication, every message you send is exposed to view. Client-side solutions such as Secure MIME (S/MIME) or pretty good privacy (PGP) can solve this problem, but they require the user involvement. Another place to focus the security efforts is on securing SMTP traffic using TLS.

But not all servers support TLS. Use of TLS requires:

- The purchase of one or more TLS certificates.
- Configuring the e-mail servers to use them.
- Additional computational resources on the e-mail servers involved.

For these reasons, many e-mail servers do not support TLS at all (i.e. most free e-mail servers like Gmail and Yahoo! do not).

We have an available tool online [7] capable to test the receiving e-mail server.

It makes sure that a receiver will not accept an unprotected e-mail. While e-mail security is mostly the responsibility of the sender, in a high security situation the receiver has too a responsibility to make sure the sender meets security requirements. RFC-3207, the Internet standard for TLS e-mail, states “A publicly-referenced SMTP server must not require use of TLS in order to deliver mail locally” [23]. This implies that security conscious organizations have a normal e-mail receiver for normal and a TLS-only receiver for secure e-mail.

This tool test that a secure e-mail receiver is configured correctly to only receive e-mail if the e-mail is sent securely, i.e. with TLS. It does not accept the receiver’s invitation to use TLS, trying to trick the receiver into accepting the e-mail insecurely.

So, what this test is looking for is the e-mail transfer to fail, meaning the receiver will not receive e-mail without protection, which indicates a correctly configured secure-only receiver.

We have tested 3 different servers: Gmail, Hotmail and Hushmail. The meanings of the columns shown in the result tables are the following:

- Pref: The MX preference for this server.
- Connect: If this server could be reached from the Internet.
- Allowed: If this server allows connections.
- Can Use: If mostly anyone can send mail to users at this site.
- TLS Advertised: If this server announces it can do TLS.
- TLS Negotiated: If this server really can do TLS.
- Sender OK: If this server will accept email from any user.
- Recipient OK: If this server will accept email to the address you entered.

Here are the result we found for Gmail:

CheckTLS Confidence Factor for "jlop84@gmail.com": 0

MX Server	Pref	Connect	Allowed	Can Use	TLS Advertised	TLS Negotiated	Sender OK	Recipient OK
gmail-smtp-in.l.google.com	5	OK (65ms)	OK (406ms)	OK (281ms)	FAIL	FAIL	OK (1,287ms)	OK (1,566ms)
alt1.gmail-smtp-in.l.google.com	10	OK (154ms)	OK (120ms)	OK (200ms)	FAIL	FAIL	OK (596ms)	OK (177ms)
alt2.gmail-smtp-in.l.google.com	20	OK (145ms)	OK (123ms)	OK (231ms)	FAIL	FAIL	OK (619ms)	OK (181ms)
alt3.gmail-smtp-in.l.google.com	30	OK (123ms)	OK (109ms)	OK (454ms)	FAIL	FAIL	OK (605ms)	OK (594ms)
alt4.gmail-smtp-in.l.google.com	40	OK (64ms)	OK (50ms)	OK (137ms)	FAIL	FAIL	OK (321ms)	OK (402ms)
<b>Average</b>		100%	100%	100%	0%	0%	100%	100%

Run same test with:

Figure 7.1 – Gmail TLS test results (table)

As we can see here and in the details above Gmail does not support TLS. (In the details we only put the result of one of the MX servers to avoid repetition).

Checking jlop84@gmail.com

looking up MX hosts on domain "gmail.com"

```
gmail-smtp-in.l.google.com (preference:5)
alt1.gmail-smtp-in.l.google.com (preference:10)
alt2.gmail-smtp-in.l.google.com (preference:20)
alt3.gmail-smtp-in.l.google.com (preference:30)
alt4.gmail-smtp-in.l.google.com (preference:40)
```

seconds test stage and result

```
[000.000] Trying TLS on gmail-smtp-in.l.google.com (5):
[000.062] Connected to server
[000.112]<-- 220 mx.google.com ESMTP k12si844971qcu.111
[000.112] We are allowed to connect
[000.112]--> EHLO checktls.com
[000.163]<-- 250-mx.google.com at your service, [24.123.1.3]
250-SIZE 35651584
250-8BITMIME
250 ENHANCEDSTATUSCODES
[000.163] We can use this server
[000.164] TLS is not an option on this server
[000.164]--> MAIL FROM: <test@checktls.com>
[000.213]<-- 250 2.1.0 OK k12si844971qcu.111
[000.213] Sender is OK
[000.213]--> RCPT TO: <jlop84@gmail.com>
[004.912]<-- 250 2.1.5 OK k12si844971qcu.111
[004.912] Recipient OK, E-mail address proofed
[004.913]--> QUIT
[004.960]<-- 221 2.0.0 closing connection k12si844971qcu.111
```

Figure 7.2 – Gmail TLS test results (detail)

At this point it is worth to comment what EHLO and STARTTLS (will appear in later analysis) are.

The main identification feature is for ESMTP (Extended SMTP) clients to open a transmission with the command *EHLO* (Extended HELLO), rather than *HELO* (Hello, the original SMTP standard). A server will respond with success (code 250), failure (code 550) or error (code 500, 501, 502, 504, or 421), depending on its configuration. An ESMTP server would return the code 250 OK in a multi-line

reply with its domain and a list of keywords to indicate supported extensions. A SMTP compliant server would return error code 500, allowing ESMTP clients to try either *HELO* or *QUIT*.

STARTTLS is a protocol feature commonly found in e-mail protocols, which allows TLS and plaintext connections to co-exist on the same port.

To use STARTTLS a client simply sends STARTTLS to the server. After sending this command both client and server switch to TLS mode, based upon the plaintext numeric reply generated by the STARTTLS command, and cannot switch back to plain text mode. The client must then send an TLS handshake to the server and the connection continues in the same manner as one on a dedicated TLS port.

In the next table we present the result for Hotmail. As Gmail, it neither supports TLS.

**CheckTLS Confidence Factor for "piji\_@hotmail.com": 0**

MX Server	Pref	Connect	Allowed	Can Use	TLS Advertised	TLS Negotiated	Sender OK	Recipient OK
mx2.hotmail.com	5	OK (139ms)	OK (106ms)	OK (110ms)	FAIL	FAIL	OK (463ms)	OK (130ms)
mx3.hotmail.com	5	OK (126ms)	OK (103ms)	OK (104ms)	FAIL	FAIL	OK (429ms)	OK (125ms)
mx4.hotmail.com	5	OK (102ms)	OK (81ms)	OK (109ms)	FAIL	FAIL	OK (375ms)	OK (119ms)
mx1.hotmail.com	5	OK (130ms)	OK (110ms)	OK (111ms)	FAIL	FAIL	OK (459ms)	OK (131ms)
<b>Average</b>		100%	100%	100%	0%	0%	100%	100%

Run same test with:

Figure 7.3 – Hotmail TLS test (table)

```

seconds test stage and result
[000.000] Trying TLS on mx3.hotmail.com (5):
[000.124] Connected to server
[000.225]<--
220 bay0-mc3-f11.Bay0.hotmail.com Sending unsolicited commercial or bulk e-mail to Microsoft's computer network is prohibited.
http://privacy.msn.com/Anti-spam/. Violations will result in use of equipment located in California and other states. Wed, 11
[000.225] We are allowed to connect
[000.226]--> EHL0 checktls.com
[000.331]<-- 250-bay0-mc3-f11.Bay0.hotmail.com (3.11.0.113) Hello [24.123.1.3]
250-SIZE 29696000
250-PIPELINING
250-8bitmime
250-BINARYMIME
250-CHUNKING
250-AUTH LOGIN
250-AUTH=LOGIN
250 OK
[000.332] We can use this server
[000.332] TLS is not an option on this server
[000.333]--> MAIL FROM: <test@checktls.com>
[000.435]<-- 250 test@checktls.com...Sender OK
[000.435] Sender is OK
[000.435]--> RCPT TO: <piji_@hotmail.com>
[000.538]<-- 250 piji_@hotmail.com
[000.538] Recipient OK, E-mail address proofed
[000.539]--> QUIT
[000.641]<-- 221 bay0-mc3-f11.Bay0.hotmail.com Service closing transmission channel

```

Figure 7.4 – Hotmail TLS test (details)

Finally we decided to create an account in Hushmail to see if what they state in their web: “Hushmail is the most secure web-based free email service in the world” is true or not.

From the results we can see that all Husmail servers really enable TLS. It seem that three of them are failing, but if we look deeper into the details we can see this is due to connection timed out and not related to TLS. Therefore we can assume the confidence factor is 100 and not 67 as the results state.



CheckTLS Confidence Factor for "jlop84@hushmail.com": 67

MX Server	Pref	Connect	Allowed	Can Use	TLS Advertised	TLS Negotiated	Sender OK	Recipient OK
mx3.hushmail.com	10	OK (111ms)	OK (92ms)	OK (93ms)	OK (91ms)	OK (360ms)	OK (95ms)	OK (115ms)
mx4.hushmail.com	10	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL
mx5.hushmail.com	10	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL
mx6.hushmail.com	10	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL
mx7.hushmail.com	10	OK (106ms)	OK (90ms)	OK (102ms)	OK (88ms)	OK (320ms)	OK (90ms)	OK (103ms)
mx8.hushmail.com	10	OK (116ms)	OK (102ms)	OK (105ms)	OK (93ms)	OK (325ms)	OK (97ms)	OK (112ms)
mx9.hushmail.com	10	OK (130ms)	OK (104ms)	OK (95ms)	OK (91ms)	OK (295ms)	OK (95ms)	OK (411ms)
mx1.hushmail.com	10	OK (112ms)	OK (255ms)	OK (92ms)	OK (93ms)	OK (337ms)	OK (92ms)	OK (119ms)
mx2.hushmail.com	10	OK (109ms)	OK (107ms)	OK (98ms)	OK (93ms)	OK (339ms)	OK (106ms)	OK (103ms)
<b>Average</b>		67%	67%	67%	67%	67%	67%	67%

Run same test with:

Figure 7.5 – Hushmail TLS test (table)

```

seconds test stage and result
[000.001] Trying TLS on mx3.hushmail.com (10):
[000.111] Connected to server
[000.203]<-- 220 smtp5.hushmail.com ESMTP Postfix
[000.203] We are allowed to connect
[000.204]--> EHLO checktls.com
[000.296]<-- 250-smtp5.hushmail.com
                250-PIPELINING
                250-SIZE 41943040
                250-ETRN
                250-STARTTLS
                250-ENHANCEDSTATUSCODES
                250-8BITMIME
                250 DSN
[000.296] We can use this server
[000.297]--> STARTTLS
[000.387]<-- 220 2.0.0 Ready to start TLS
[000.388] TLS is an option on this server
[000.650] Connection converted to SSL

```

```
[000.650]~~> EHLO checktls.com
[000.748]<~~ 250-smtp5.hushmail.com
          250-PIPELINING
          250-SIZE 41943040
          250-ETRN
          250-AUTH PLAIN LOGIN
          250-AUTH=PLAIN LOGIN
          250-ENHANCEDSTATUSCODES
          250-8BITMIME
          250 DSN
[000.748]    TLS successfully started on this server
[000.749]~~> MAIL FROM: <test@checktls.com>
[000.842]<~~ 250 2.1.0 Ok
[000.843]    Sender is OK
[000.843]~~> RCPT TO: <jlop84@hushmail.com>
[000.957]<~~ 250 2.1.5 Ok
[000.958]    Recipient OK, E-mail address proofed
[000.958]~~> QUIT
[001.049]<~~ 221 2.0.0 Bye
```

seconds test stage and result

```
[000.000]    Trying TLS on mx4.hushmail.com (10):
[010.043]    Cannot connect to server (reason: Connection timed out)
```

seconds test stage and result

```
[000.000]    Trying TLS on mx5.hushmail.com (10):
[010.025]    Cannot connect to server (reason: Connection timed out)
```

seconds test stage and result

```
[000.000]    Trying TLS on mx6.hushmail.com (10):
[010.025]    Cannot connect to server (reason: Connection timed out)
```

Figure 7.6 – Hushmail TLS test (details)

## Chapter 8

### Instant messaging

#### 8.1 Introduction

Instant messaging (IM) is a form of real-time direct text-based communication between two or more people using personal computers or other devices, along with shared software clients.

In the figure below we can see the main protocols used for instant messaging, the clients implementing them, and some other useful information such as the TLS support.

Later, in the next section we are going to analyze the security of the protocols the most used IM clients around the world (AIM, Gtalk, Skype, Windows Live Messenger and Yahoo! Messenger) use in order to determine the best one for an end user concerned about his security.

Protocol	Creator	First public release date	License	Identity (not inc. alias)	Asynchronous message relaying	Transport Layer Security	Unlimited number of contacts	Bulletins to all contacts	One-to-many routing	Spam protection	Supports groups or channels for members / nonmembers / nobody	Audio/VoIP	Webcam/Video
Gadu-Gadu	G� Network	2000 Jul 17	Proprietary	Unique number e.g. 12345678	Yes	No	Yes	No	Centralistic	Yes <sup>5</sup> (simple)	Yes	Yes	Yes
Gale	Dan Egnor	?	Open standard	Unique RSA key, aliased to user@domain	?	Yes (public/private key)	?	?	?	?	Yes (multiple simultaneous, any size, programmable, encrypted)	No	No
IRC	JaMko Dikainen	1988 Aug	Open standard	NicknameUsername@hostname (or "hostmask") e.g. user~usr@a.b.com <sup>1</sup>	Yes, but via a memo system that differs from the main system	Yes, depending on individual server support	No <sup>3</sup>	No	Simplistic multicast	Medium	Yes (everyone, multiple simultaneous, any size)	No	No
Microsoft Messenger service Messenger SMD NetBEUI - Net Send	Microsoft	1990	Proprietary	NetBIOS	Yes	No	Yes	Yes	Yes	No	No	No	No



MSN (Windows Live Messenger, etc)	Microsoft	1999 Jul	Proprietary	E-mail address (Windows Live ID)	Yes	No	Only for certified robots	No	Centralistic	Yes	Yes	Yes	Yes
OSCAR protocol (AIM, ICQ)	ADL	1997	Proprietary	Username, Email Address or UIN e.g. 12345678	Yes	Yes (Aim Pro, Aim Lite)	No	No	Centralistic	client-based	Yes (Multiple, simultaneous)	Yes	Yes
PSYC (Protocol for Synchronous Conferencing)	PSYC Project	1995	Open standard	PSYC URI as in psyc://server.example.net /~nickname	Yes	Yes	Yes	Yes	Custom multicast	Yes	Yes (multiple simultaneous, any size, programmable)	?	?
RVP (Windows Messenger, etc)	Microsoft	1997 Mar	Proprietary	Windows Active Directory Login	No	No	?	No	Centralistic	None	No	?	?
SIP/SIMPLE	IETF	2002 Dec	Open standard	user@hostname	Yes	Yes	Yes	Yes	No	Medium	?	Yes	Yes
Skype protocol	Skype	?	Proprietary	Username	No	Proprietary	?	No	?	?	Yes	Yes	Yes
TOC protocol (deprecated)	ADL	1998	Proprietary	Username or UIN e.g. 12345678	Yes	No	?	?	Centralistic	?	paying members only	?	?
TOC2 protocol	ADL	2005 Sep	Proprietary	Username or UIN e.g. 12345678	Yes	No	No	No	Centralistic	No	paying members only	?	?
XMP (Jabber)	Jeremie Miller, standardized via IETF	1999 Jan	Open standard	Jabber ID (JID) e.g. user@a.b.c/home <sup>2</sup>	Yes	Yes	Yes	Yes	Unicast lists	Several Standardized Types	Optional	Yes	Yes
YMSG (Yahoo! Messenger)	Yahoo!	?	Proprietary	Username	Yes	No	No	Yes	Centralistic	Yes	No (groups discontinued due to liability)	PMS, Conferoes, and Chat Rooms	Yes
Zephyr Notification Service	MIT	1987	Open standard	Kerberos principal e.g. user@ATHENA.MIT.EDU	Yes	No	Yes	Yes	Unicast lists	No	Yes	No	No
Protocol	Creator	First public release date	License	Identity (not inc. alias)	Asynchronous message relaying	Transport Layer Security	Unlimited number of contacts	Bulletins to all contacts	One-to-many routing <sup>4</sup>	spam protection	Supports groups or channels for members / nonmembers / nobody	Audio/VoIP	Webcam/Video

Figure 8.1 – IM protocols and its features

## 8.2 AOL Instant Messaging (AIM) and ICQ – OSCAR Protocol

AOL instant messenger (AIM) is the most widely used service in the instant messaging market, with an estimated user base of 53 million people- accounting for around 52% of the Instant Messaging market.

Later America Online also bought to the Israeli company Mirabilis the ICQ program, using the same protocol, and reaching more than 100 millions users [26].

Since so many individuals use the AIM/ICQ protocol, any kind of weakness could affect a very significant number of people.

They use the proprietary OSCAR (**O**pen **S**ystem for **C**ommunic**A**tion in **R**ealtime) instant messaging protocol, used by most of the clients. However, AOL also created a simpler protocol called TOC, with less features, but due to its simplicity, much used for clients that only require basic chat functionality.

The TOC protocol specifications [27] were made available by AOL, while OSCAR is a closed protocol that third parties have had to reverse-engineer.

What was found out is that both protocols send all the data in plain-text, making the eavesdropping almost trivial just using an sniffer if you are sharing the same LAN.

## 8.3 GTalk - XMPP

Google Talk (GTalk) is a Windows web-based application for instant messaging and voice over internet protocol (VOIP), offered by Google Inc.

Instant messaging between the GTalk servers and its clients uses an open protocol, XMPP, allowing users of other XMPP clients also to communicate with Google Talk users. VoIP in Google Talk uses an older version of what would later become the Jingle protocol (an extension of XMPP). However, the technology used within the Google server network is not publicly known.

But the problem here is that XMPP, even being a standardized protocol, admits several ways of implementations (some more secure than others).

Specifically, the official Google client uses a solution called Google-Token. It works like this:

- The client connects to Google via TLS to do the authentication.
- Google gives a token to the client.
- The client use that token as authentication method.

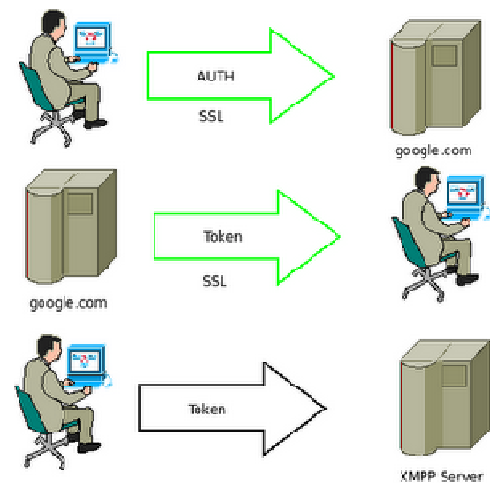


Figure 8.2 – Google client operation

Only the connection between the Google Talk client and the Google Talk server is encrypted, except when using the web-based version of the client, Gmail's chat over HTTP. Hopefully, nowadays the default protocol when you connect the Gmail server is https.

Thus messages are not necessarily encrypted end-to-end, although it is possible to have end-to-end encryption over the GTalk network using Zfone (software for secure voice communication over the Internet) or OTR (off-the-record) chat (a cryptographic protocol that provides strong encryption for instant messaging conversations using the AES symmetric-key algorithm [28], the Diffie-Hellman key exchange [25] and the SHA-1 hash function [29]) when all participants connect over HTTPS. Some XMPP clients also natively support encryption with Google Talk's servers.

But if we choose some alternative such as Gaim or Pidgin (necessary under Linux), things work in a very different way:

- The client starts the communication with the server XMPP.
- A TLS tunnel is negotiated.
- Credentials are sent.

At that point, all the data travels encrypted.

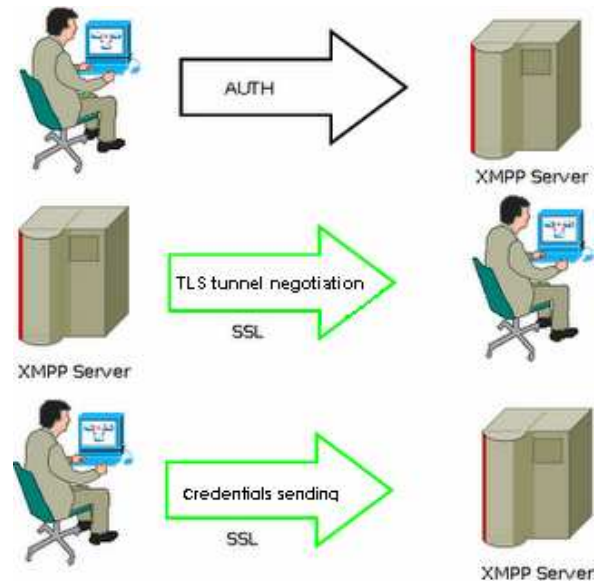


Figure 8.3 - Alternative clients operation

## 8.4 Skype

Skype is a proprietary software application developed by Skype Technologies S.A., a corporation that claims to be registered in Luxembourg. The company was founded by Janus Friis and Niklas Zennstrom, the same entrepreneurs who developed the popular KaZaA file trading system. It allows users to make voice calls over the Internet. Skype has also become popular for its additional features which include instant messaging, file transfer, and video conferencing.

Like KaZaA, Skype is based on peer-to-peer technology: instead transmitting all voice calls through a central server, Skype users search for other users to connect to, enabling them to search other Skype users and send them messages.

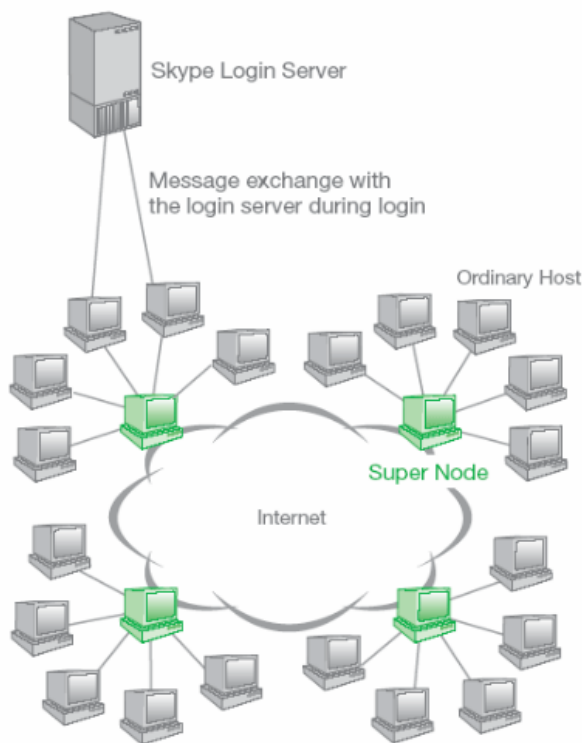


Figure 8.4 – Skype Network

Although it uses peer-to-peer communications for locating other Skype users and for transmitting voice communications, Skype relies on a central authentication server to authenticate users and software distributions.

The Skype protocol is not public, but numerous attempts to study and reverse engineer it have been undertaken to reveal the protocol, investigate security or to allow unofficial clients.

It makes wide use of cryptography to authenticate user and server identities, and to protect the content transmitted across the P2P.

It uses only standard cryptographic primitives (well-established, low-level cryptographic algorithms) to reach its objectives. These primitives include the AES block cipher, the RSA public-key cryptosystem, the ISO 9796-2 signature padding scheme, the SHA-1 hash function, and the RC4 stream cipher [12].

Skype operates a certificate authority for user names and authorizations. Digital signatures created by this authority are the basis for identity in Skype. Skype nodes entering into a session correctly verify the identity of their peer. It is infeasible for an attacker to spoof a Skype identity at or below the session layer.

Skype uses a proprietary session-establishment protocol. The cryptographic purposes of this protocol are to protect against replay (a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed), to verify peer identity, and to allow the communicating peers to agree on a secret session key. The communicating peers then use their session key to have a confidential communication during the lifetime of the session.

Skype stores registration information both on the caller's computer and on a Skype server. Skype uses this information to authenticate call recipients and to assure that callers seeking authentication are accessing a Skype server rather than an impostor. Skype uses RSA public key encryption to accomplish this.

The Skype server has a private key, and distributes that key's public counterpart with every copy of the software. As part of user registration, the user selects a desired username and password. Skype locally generates public and private keys. The private key and a hash of the password are stored as securely as possible on the user's computer.

The Skype server verifies that the selected username is unique and that follows Skype's naming rules. The server stores the username and a hash of the hash of the user's password  $[H(H(P))]$  in its database.

The server now forms and signs an identity certificate for the username that binds the username, its verification key and the key identifier.

Then, for each call, Skype uses 256 bit AES encryption to encrypt communication between users, making difficult the decryption of these communications. Skype's encryption is inherent in the Skype Protocol and is transparent to the users. The client creates a session key using its random number generator.

In conclusion, the designers of Skype did not hesitate to employ cryptography widely and well in order to establish a foundation of trust, authenticity, and confidentiality for their peer-to-peer services. The implementers of Skype implemented the cryptographic functions correctly and efficiently. As a result, the confidentiality of a Skype session is far greater than that offered by a wired or wireless telephone call or by email and email attachments.

However, it should be possible for even unprivileged participants of the network to perform traffic analysis and determine when one user calls another user. It is unknown if the design of the Skype network makes it possible for some nodes to monitor all searches and call set-up traffic, or if instead each node would only see a portion of the overall traffic.

Finally, we have to remember that the security of the Skype system also depends entirely on the good will of Skype's programmers and the organization running Skype's back-end servers. It is possible that there are back doors in the system allowing the Skype organization or others to eavesdrop or record Skype conversations.

## 8.5 Windows Live Messenger – Microsoft Notification Protocol

Windows Live Messenger is an instant messaging client created by Microsoft that is currently designed to work with Windows.

Windows Live Messenger uses the Microsoft Notification Protocol (MSNP) over TCP (and optionally over HTTP to deal with proxies) to connect to the .NET Messenger Service.

The protocol is not completely secret; Microsoft disclosed version 2 (MSNP2) to developers in 1999, but never released versions 8 or higher to the public. The .NET Messenger Service servers currently only accept protocol versions from 8 and higher, so the syntax of new commands sent from versions 8 and higher is only known by using packet sniffers like Wireshark.

This has been an easy task because, in comparison to many other modern instant messaging protocols like XMPP, the Microsoft Notification Protocol does not provide any encryption and everything can be captured easily using packet sniffers. Actually the latest versions include TLS, but only for authentication.

This lack of proper encryption also makes wiretapping friend lists and personal conversations a trivial task, especially in unencrypted public Wi-Fi networks.



## 8.6 Yahoo! Messenger

Yahoo! Messenger (YMSG) is an instant messaging client and its associated protocol provided by Yahoo!.

The login process for YMSG is quite complex. First of all, the client introduces itself with a message containing its username. Then the server responds with a rather long seed value. Finally, the client put this into an algorithm, along with the account's password, to produce two response values looking like variable assignments which are sent to the server. If these values match the server's expectations, the client is admitted and sent data associated with that account (such as friend lists).

Originally the YMSG login procedure suffered from a security flaw known as a replay attack, in which a given password (or other authentication information) is always identically scrambled when sent across the network. This allows any attacker who witnesses the transmission to merely reproduce the message in order to successfully log in, without actually needing to know the original password (or other details) which generated it. But some years ago Yahoo! upgraded its service to introduce a random element to each login attempt, defeating any further potential for replay attacks.

With the exception of the login authentication details, data sent over a YMSG connection is not encrypted. YMSG uses a binary format in which the text portions of the data are transmitted in plain view. Hence, while it is difficult for an attacker to seize control of a Yahoo! IM account, it is quite easy for them to read all messages sent to and from the account holder, along with other details such as the list of friends, if the attacker has control of one of the computers through which the data is routed.

## **Conclusion**

During this master thesis we have presented several ways to avoid Internet data eavesdropping.

We started trying out an alternative way to encryption, source routing. Source routing basis is heading our data through the routers we know are secure (since the initial scope of this thesis was avoiding the interception by foreign governments, so we assumed national networks were secure).

After developing a way to find a way to identify the possible compromised routers, we bumped into some security problem associated to source routing that led Internet providers to disable this option.

After that we decided to continue with the encryption option studying the most common secure protocols used over the Internet (TLS, SSH and IPsec) in combination with a tool to avoid traffic analysis, and then we studied how they are implemented in the most common software clients used for web browsing, e-mailing or instant messaging.

After this study we can conclude that the best option for end-users to secure their Internet communication is using software (browsers, e-mail clients, etc) implementing TLS encryption. We think this option is much better than IPsec because users don't have to worry about installing extra software.

We also encourage to use TOR as a way to avoid traffic analysis. It is really simple to use, just install it and it will work in background without any problem. This combined with TLS is great to fight against eavesdropping.

As we have seen in our little experiment, most of the common web browsers like Firefox are ready to negotiate a TLS connection with the server.

We only recommend employing SSH for its primary use, a substitute of Telnet to do remote login in other computers.

As we have seen, TLS encryption is not an option for most of e-mail providers, and they can only protect the information intermittently from intrusion.

Because of that we think that using one of the e-mail clients described above is the best and simple option when talking about e-mail.

When talking about Instant Messaging, we do the same recommendation. Once you know which available clients are secure the best option is to use them.

As we have seen, Skype is much secure than the others, even though it can have backdoors. In most cases the security Skype provides will be enough, but when dealing with very sensitive data we strongly recommend running our IM client over a virtual private network.

Finally, we have to say that the cost of being protected is null, because we always have free software available for it (like most of the web browsers and IM clients). If we need some other encryption programs such as PGP (now part of Symantec) we also have the open source option for free (OpenPGP).

## References

- [1] IETF, RFC 791: Internet Protocol. Darpa Internet Program Protocol Specification. (1981).
- [2] Microsoft. Microsoft Security Bulletin (MS99-038)  
<http://www.microsoft.com/technet/security/bulletin/fq99-038.msp>. Retrieved 05-14- 2010.
- [3] IETF, RFC 2827: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing (2000).
- [4] IETF, RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 (2008).
- [5] IETF, RFC 4251: The Secure Shell (SSH) Protocol Architecture (2006).
- [6] Cisco Systems. An Introduction to IP Security (IPSec) Encryption (2008).  
<http://www.cisco.com/application/pdf/paws/16439/IPSECpart8.pdf>
- [7] CheckTLS.com. <http://www.checktls.com/> Retrieved 07-15-2010.
- [8] IETF, RFC 2818: HTTP Over TLS (2000).
- [9] IETF, RFC 4880: OpenPGP Message Format (2007).
- [10] IETF, RFC 5751: Secure/Multipurpose Internet Mail Extensions (S/MIME) (2010).
- [11] The Tor Project. Tor Overview. <http://www.torproject.org/>. Retrieved 07-28-2010.
- [12] Tom Berson. Skype Security Evaluation (2005).  
<http://www.anagram.com/berson/skyeval.pdf>
- [13] Deb Shinder. Comparing VPN Options.  
<http://www.windowsecurity.com/articles/VPN-Options.html>. Retrieved 08-9-2010.
- [14] European Parliament. Directive 2006/24/EC of the European Parliament and of the Council.  
<http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32006L0024:EN:NOT>. Retrieved 08-14- 2010.
- [15] Electronic Frontier Foundation. CALEA: Frequently Asked Questions.  
<http://www.eff.org/pages/calea-faq>. Retrieved 08-15-2010.

- [16] MMLC Group. Data Protection and Privacy Issues in China.(2008). Retrieved 08-16-2010.
- [17] INDECT. <http://www.indect-project.eu/>. Retrieved 08-17-2010
- [18] Electronic Privacy Information Center. Carnivore FOIA Documents. Retrieved 08-17-2010
- [19] Jack O'Neill. Echelon: Somebody's listening. (2005)
- [20] Mike Sposato. The FBI's Magic Lantern. (2001).  
[http://www.wnd.com/news/article.asp?ARTICLE\\_ID=25471](http://www.wnd.com/news/article.asp?ARTICLE_ID=25471). Retrieved 08-17-2010
- [21] Kevin Poulsen. FBI's Secret Spyware Tracks Down Teen Who Made Bomb Threats. (2007).[http://www.wired.com/politics/law/news/2007/07/fbi\\_spyware](http://www.wired.com/politics/law/news/2007/07/fbi_spyware). Retrieved 08-17-2010
- [22] Mozilla Europe. Stay safe while you surf.  
<http://www.mozilla-europe.org/en/firefox/security/#features>. Retrieved 08-17-2010.
- [23] IETF, RFC 3207: SMTP Service Extension for Secure SMTP over Transport. (2002).
- [24] CrypTool. The functionality of RSA chiper.  
<http://cryptool.org/download/RSA/RSA-Flash-en/player.html>. Retrieved 08-18-2010.
- [25] RSA Laboratories. What is Diffie-Hellman?  
<http://www.rsa.com/rsalabs/node.asp?id=2248>. Retrieved 08-18-2010.
- [26] Time Warner. ICQ Celebrates 100 Million Registered Users. (2001).  
<http://www.timewarner.com/corp/newsroom/pr/0,20812,668719,00.html>. Retrieved 08-25-2010.
- [27] America Online, Inc. TOC1.  
<http://terraim.cvs.sourceforge.net/viewvc/terraim/terraim/src/toc/TOC1.txt>. Retrieved 08-25-2010.
- [28] Federal Information Processing Standards. Advanced Encryption Standard (AES). (2001). <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

- [29] Federal Information Processing Standards. Secure Hash Standard.(1995).  
<http://www.itl.nist.gov/fipspubs/fip180-1.htm>. Retrieved 08-25-2010.
- [30] Simson L.Garfinkel. VoIP and Skype Security. (2005).  
<http://skypetips.internetvisitation.org/files/VoIP%20and%20Skype.pdf>