JRC TECHNICAL REPORT

# A security analysis of email communications

Ignacio Sanchez
Apostolos Malatras
Iwen Coisel

*Reviewed by: Jean Pierre Nordvik*

2015

EUR 28509 EN

Joint Research Centre

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Ignacio Sanchez
Address: Joint Research Centre, Via Enrico Fermi 2749, I - 21027 Ispra (VA), Italia
E-mail: ignacio.sanchez@ec.europa.eu

JRC Science Hub
*https://ec.europa.eu/jrc*

JRC 99372

EUR 28509 EN

ISSN 1831-9424

ISBN 978-92-79-66503-5

doi:10.2760/319735

Luxembourg: Publications Office of the European Union, 2015

Abstract

The objective of this report is to analyse the security and privacy risks of email communications and identify technical countermeasures capable of mitigating them effectively. In order to do so, the report analyses from a technical point of view the core set of communication protocols and standards that support email communications in order to identify and understand the existing security and privacy vulnerabilities. On the basis of this analysis, the report identifies and analyses technical countermeasures, in the form of newer standards, protocols and tools, aimed at ensuring a better protection of the security and privacy of email communications. The practical implementation of each countermeasure is evaluated in order to understand its limitations and identify potential technical and organisational constrains that could limit its effectiveness in practice. The outcome of the above mentioned analysis is a set of recommendations regarding technical and organisational measures that when combined properly have the potential of more effectively mitigating the privacy and security risks of today's email communications.

# Contents

# 1 Executive Summary

The objective of this report is to analyse the security and privacy risks of email communications and identify technical countermeasures capable of mitigating them effectively. In order to do so, the report analyses from a technical point of view the core set of communication protocols and standards that support email communications in order to identify and understand the existing security and privacy vulnerabilities. On the basis of this analysis, the report identifies and analyses technical countermeasures, in the form of newer standards, protocols and tools, aimed at ensuring a better protection of the security and privacy of email communications. The practical implementation of each countermeasure is evaluated in order to understand its limitations and identify potential technical and organisational constrains that could limit its effectiveness in practice. The outcome of the above mentioned analysis is a set of recommendations regarding technical and organisational measures that when combined properly have the potential of more effectively mitigating the privacy and security risks of today's email communications.

Email is the electronic communication protocol par excellence used on a daily basis by hundreds of millions of European citizens, as well as by most governments and businesses. The email ecosystem is a highly interoperable one and relies on a core set of protocols initially designed more than three decades ago, in an early digital context much different from the one found today in terms of digital privacy and security risks. Consequently, this core set was not originally designed with privacy and security requirements in mind, but under the assumption that the several actors involved in email communications could trust each other and that the digital communication links were secure.

With the massive adoption of Internet and email communications, a new rich set of complementary standards and tools were created in order to tackle the growing security and privacy concerns. However, these enhanced protocols and tools have failed in practice to deliver an effective protection. As a result, world-wide email communications remain largely vulnerable to security and privacy threats.

The main findings of this report are summarised as follows:

- **Email communications are in general not sufficiently protected.** The results of the evaluation suggest that the majority of world-wide email communications are subject to serious privacy and security risks. In most of the cases, content transmitted by email can be intercepted by third parties putting at risk the confidentiality, integrity and availability of the information exchanged, such as the text of the message and the files attached to it.
- **There are standards, protocols and techniques capable of enhancing the security of email communications but they are not always used or implemented properly in practice**. Although there is no single countermeasure that has proven to be effective against all security and privacy risks, there are mature technological solutions that when combined and implemented properly can mitigate more effectively the email risks identified in this report.
- **Mature and interoperable end-to-end email security solutions exist but are rarely used in practice.** Mature end-to-end email security solutions, namely SMIME and OpenPGP (e.g. PGP/GPG), are already readily available but unfortunately rarely used in practice. The main barrier that has been identified for their adoption by European

citizens is the lack of support by commercial providers that do not integrate them into their web-based email clients and mobile applications. One hypothesis for this lack of support and integration is linked to the fact that end-to-end security solutions would impact their current business models which currently involve the usage of the data transmitted and received by email. As a result of this lack of support and integration, currently the usage of end-to-end security solutions presents usability issues and requires certain IT skills that the average citizen does not possess.

- **Email communication channels (SMTP to SMTP) are not sufficiently protected in practice.** Security of email communication channels can be provided by employing SSL in the form of the STARTTLS protocol. However, we have observed that in practice the implementation of STARTTLS does not offer sufficient protection due to the following factors:
  - *Fall back to unencrypted communications.* When the usage of STARTTLS between two servers fails, the communication downgrades to an unencrypted communication in order to preserve the interoperability. Therefore, STARTTLS can only be currently seen in practice as a sort of opportunistic encryption, vulnerable to easy to perform "active downgrade" attacks.
  - *Lack of validation of server certificates.* Self-signed server certificates are accepted in practice in order to preserve interoperability, opening the door to trivial "Man-In-The-Middle" attacks.
- **Lack of security in DNS has a direct impact on the security of email communications.** The public DNS system plays a central role in email communications as it is used to glue the several email actors together. As a result of this dependence, DNS vulnerabilities can be exploited in order to attack email communications. Therefore, in order to create secure email communications it is required to secure the DNS communications as well. Existing deployment of DNSSEC should be carefully analysed to determine the difficulty of deploying such solutions and identify their overhead on the DNS traffic. In addition to providing reliable and secure resolution of MX, SPF and A records, DNS can also help with the management of the public keys employed in STARTTLS. To that end, the implementation of DNSSEC with DANE should be strongly considered. An alternative solution called DNSCurve using elliptic curve cryptography was recently introduced and could be considered as well.
- **Email identity spoofing is still a major risk in email communications.** Email identity spoofing can be easily performed despite the specific countermeasures deployed to fight SPAM, which indirectly help mitigate the threat (i.e. SPF records). Given the design of the email protocols, only end-to-end security (i.e. SMIME or PGP/GPG) can effectively mitigate this risk.

The following recommendations have been identified in order to address the above mentioned issues.

**Incentivise industry to support end-to-end solutions.** We recommend that email service providers, in particular the big industry players that provide webmail services, are incentivised to provide support for interoperable end-to-end email security solutions (i.e. SMIME or OpenPGP) and integrate them into their products and services.

It is our hypothesis that the usage of end-to-end security solutions could be currently perceived by the industry as an impact to the existing business models based on the compilation and analysis of the personal data exchanged by email (i.e. for marketing purposes). Due to this fact, industry players following these practices will rarely support such end-to-end security solutions in email. Interoperable end-to-end email security solutions such as SMIME and OpenPGP have proven to be efficient in the protection of the privacy and security of email communications and should be promoted. Currently, the main impediment to their effective deployment is concentrated in the following aspects:

- *Usability issues.* Major email providers (such as Gmail or Hotmail) don't offer support for SMIME or OpenPGP. Currently, the vast majority of citizens use webmail based systems or mobile apps developed by the email providers, which in most cases lack support for these technologies. Even though many email providers support the usage of standalone email clients, the set-up of this solution not only requires extra effort and specialised knowledge from the citizens that will use the service, but also presents serious usability issues compared to the convenience of the web based interface.
- *Key management.* In the case of SMIME, the process required to obtain an email certificate from a trusted provider is still too complex for users without specialised IT skills. The process is also cumbersome and usually only the most determined users would be willing to follow it. Even though there are some providers that offer such services free of charge, in many cases the user will have to pay for the service. In the case of OpenPGP, there is no global trusted key repository for the storage and sharing of keys and the system is based on a more distributed model which is more difficult to be used transparently.

**Promote the integration of end-to-end solutions into existing products and services.** We recommend that email service providers and developers of email client software (including webmail systems) are incentivised to provide integration with interoperable end-to-end solutions (ie: SMIME and PGP/GPG) in a transparent and usable way.

End-to-end solutions could be promoted if support for SMIME and OpenPGP would be provided by major email providers in their web-based services and mobile applications. In addition, the implementation of such solutions should be as transparent as possible, while still maintaining interoperability and keeping the user in control of the process. The provision of SMIME certificates could be integrated as part of the procedure followed to create an account and the management of keys could be integrated in the contact list already provided in a transparent way by email providers. A mutual trust system between email providers could be envisaged in order to facilitate the transparent recovery of the public key for a given recipient who operates on another email provider.

**Promote the security of the email communication channels.** We recommend that the usage of STARTTLS for the protection of the SMTP communication channels is promoted and required by default following a security by default approach.

There is a big percentage of the global email traffic that does not use STARTTLS at all. This fact is related to the interoperability of the email system. A SMTP server will still be interoperable even if STARTTLS is not supported at all. Given that this feature is completely hidden to the users, there is no actual pressure for the service provider to enable it at all. The usage of STARTTLS could be promoted in the following ways:

- Raise citizens' awareness regarding the dangers of unencrypted email communications.
- Make public information about the usage of STARTTLS per provider (such as the Google transparency report).
- Set of minimum requirements for a system to become interoperable or be considered secure

(see next point).

**Development of a minimum set of security requirements supported by an "Email Privacy Seal".** We recommend the creation of a minimum set of requirements for an email system to be secure and interoperable (including full STARTTLS support) and accordingly consider the creation of an "email privacy seal" to highlight those email providers complying with this security and privacy requirements. The usage of this "EU Email Privacy Seal" could help the user understand the level of commitment of this particular provider with the security and privacy of email communications and give a level of confidence in using their services.

Moreover, it will implicitly instigate email service providers to optimize their services in terms of security to maintain their competitiveness. In particular, the following requirements could be considered:

- Full SMIME support using certificates signed by a trusted CA
- DNSSEC support with DANE
- SPF records
- SMIME and OpenPGP support in proprietary web interfaces, desktop and mobile applications.

# 2 Introduction

The first network electronic mail was sent in 1971 by Ray Tomlinson through the Advanced Research Projects Agency Network (ARPANET), almost two decades before Internet was born and the first World Wide Web website appeared online. This was the first time an electronic message was sent over a network to a remote user located on another server. It was also the first time that the @ symbol was used as part of an email address to split the username and the destination domain fields.

The email system as we know it today started to be formalized in 1982 with the standardization of the SMTP protocol and it began to become massively adopted in the mid 90's with the popularization of the Internet. Today, it is estimated that there are 2.5 billion users world-wide with over 190 billion emails being sent and received every single day[1].

One key element of the world-wide email ecosystem is its openness and interoperability. There are no restrictions regarding who is allowed to become part of the system or which specific implementation shall be used. Any host of the network can become a fully functional email server for the delivery and reception of email to/from any other system, provided it complies with the set of standard protocols.

The set of protocols used by the email system were originally designed in a context where all the network nodes assumed that they could trust each other. In the very early days of the Internet, there were not many security and privacy threats and most of the users of the network and the email system were researchers or engineers. The massive adoption of Internet by both industry and citizens in the mid 90's completely transformed the security threat landscape. In response to the growing security and privacy needs, a new set of protocols as well as extensions to the current ones were proposed as possible countermeasures to mitigate the risk.

Today, the biggest strength of email, its openness and interoperability, has contributed to form its biggest weakness, the lack of protection for the privacy and security of the communications. Even though an extensive set of protocol extensions and techniques aimed at increasing the security and privacy of email communications exist, the primary goal of interoperability greatly limits their effective application in practice.

As a result of the latter observation, in practical terms the protection of the security and privacy of email communications is very limited. In the absence of specific security measures such as end-to-end encryption and signature (e.g. using PGP/GPG or SMIME), emails can be easily eavesdropped and falsified. When a new email is received in the inbox of a user, there is no implicit guarantee that the email was actually sent by the advertised sender (based on the information provided in the "from address" field). There is also no certainty that the subject or email contents (including any attached file) has not been intentionally modified in transit or even completely replaced by new content. Furthermore, it is possible that at some point in the path between the sender and the recipient, the email and all of its contents have been eavesdropped, read and stored by an attacker.

[1]Statistics provided by the Radicati Group for the year 2014. http://www.radicati.com/wp/wp-content/uploads/2014/01/Email-Statistics-Report-2014-2018-Executive-Summary.pdf

Hundreds of millions of European citizens use email on a daily basis to transmit personal information. Citizens often rely on email for the reception of bank documents, invoices and many other types of sensitive documents and information. The email identity of a citizen has become a commonly trusted soft identity, often used by third parties for authentication reasons, such as for example the password recovery mechanisms of websites that are based on email. In corporate environments email is also heavily used for the transmission of all sort of information including personal information of employees and customers.

In this context, the effective protection of the security and privacy of email communications is of paramount importance. To that end, this report will analyse the core set of communication protocols involved in world-wide email communications and evaluate the existing security and privacy threats and vulnerabilities. Moreover, the report will identify the main standards proposed so far to enhance the security of the communications and evaluate their practical capacity to protect the communications. Finally, on the basis of this analysis, the report will provide a set of conclusions and recommendations.

## 2.1 Objective of the report

The objective of this report is to analyse the security and privacy risks of email communications and identify technical countermeasures capable of mitigating them effectively. In order to do so, the report analyses from a technical point of view the core set of communication protocols and standards that support email communications in order to identify and understand the existing security and privacy vulnerabilities. On the basis of this analysis, the report identifies and analyses technical countermeasures, in the form of newer standards, protocols and tools, aimed at ensuring a better protection of the security and privacy of email communications. The practical implementation of each countermeasure is evaluated in order to understand its limitations and identify potential technical and organisational constrains that could limit its effectiveness in practice. The outcome of the above mentioned analysis is a set of recommendations regarding technical and organisational measures that when combined properly have the potential of more effectively mitigating the privacy and security risks of today's email communications.

## 2.2 Scope and structure of the report

This report analyses the current threats and vulnerabilities of the world-wide email system both from a theoretical and a practical point of view. In order to do so, the study focuses in the analysis of the rich set of standard protocols and techniques used to deliver an email from the sender to the recipient, including those that are meant to provide an additional layer of security and privacy protection.

The analysis of the report is focused exclusively in such set of standard protocols and techniques and does not consider the security of specific software implementations (e.g. buffer overflow bugs in SMTP servers) or network links that are assumed to be insecure.

The report begins by analysing in chapter 3 the email ecosystem, describing its structure as well as the several actors, roles and main protocols involved in the transfer of an email from the sender to the intended recipient. The chapter also identifies and presents a technical analysis of the core set of standard protocols involved in email communications.

On the basis of the analysis performed, chapter 4 elaborates a threat and vulnerability analysis of the security of email communications. The chapter begins by summarising the existing threats against the security and privacy of email communications. Based on the threats identified, it identifies and analyses the existing vulnerabilities that are exploited by these threats. Finally, the chapter describes in detail from a technical point of view the attack vectors that are commonly employed by the existing threats in order to exploit the identified vulnerabilities and impact the security and privacy of the communications.

Chapter 5 identifies and provides an analysis of the existing standards that have been proposed as countermeasures to address the threats and vulnerabilities described in the previous chapter. Each countermeasure is carefully analysed with respect to its ability to address the security and privacy risk and the potential limitations that exist in its practical application.

The conclusions of the report are presented in chapter 6 where a comprehensive list of recommendations is also provided on the basis of the results obtained.

# 3 Email systems overview

To understand the security requirements and challenges in current electronic mail systems, we first examine the general architecture of such systems as defined in related Request For Comments (RFCs). Figure 1 illustrates the evolution of RFC standards concerning email systems. We additionally study the functional components of email systems, the protocols used for communication and information exchange and the various communication patterns between the components of email systems.

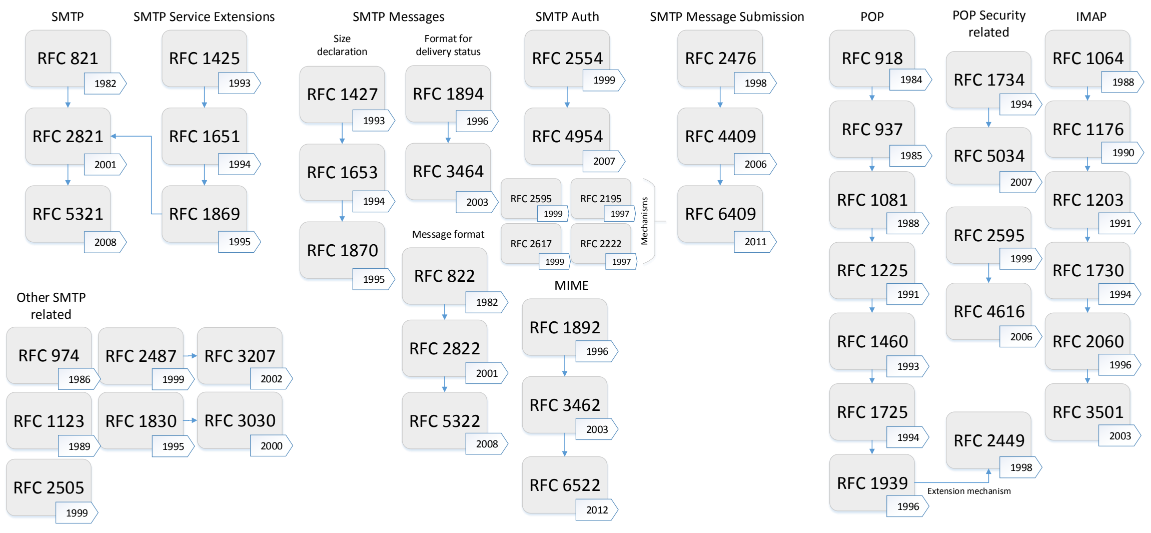


Figure 1: List of most widely utilised RFCs regarding email systems.

## 3.1 Architecture of email systems

The functionality of email systems refers to the exchange of information between interested parties in a 1 to N manner, namely one sender communicating with one or more receivers. A broad view of the email systems' architecture can be seen in Figure 2 . The communication involves a series of functional components, such as the sender, the receiver, and the mail server, and takes place over a local network or the Internet. In the following, we elaborate on the elements of the architecture of email systems.

Figure 3 presents the overall architecture of email systems in the more general form and with all considered components.

### 3.1.1 Client (Sender)

The client, also known as the sender, is the initiator of the email communication exchange. Essentially, a user decides to send an email to another user or a number of other users and to do so utilises a dedicated component, i.e. the *User Agent*. The User Agent is a software program with the following functionality:

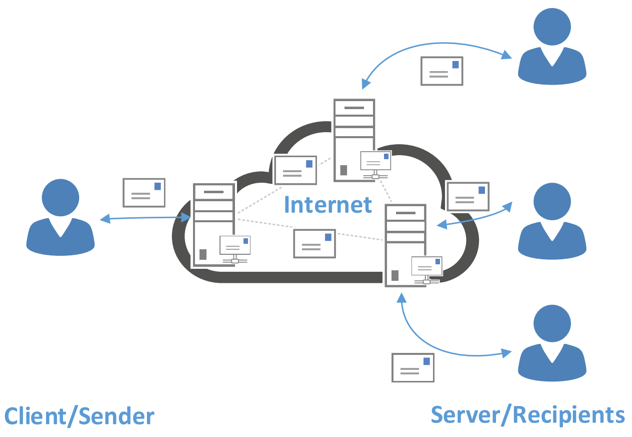


Figure 2: General overview of email systems architecture.

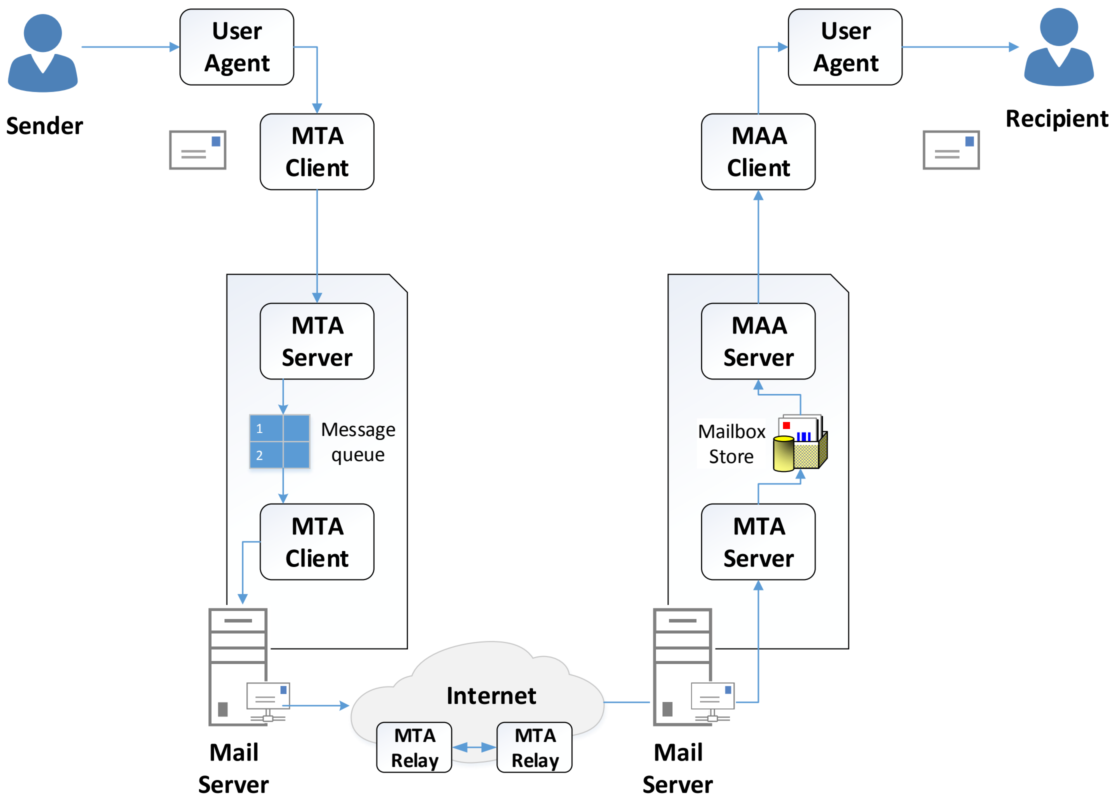


Figure 3: End-to-end architecture of email systems.

- Read emails.
- Write emails.
- Reply to emails.
- Forward emails.
- Manage local mailboxes on the users' computers.

In the first years of electronic mail systems, user agents were basically command-line driven, whereas in the last years they have evolved to powerful, user-friendly GUI-based systems, e.g. MS Outlook, Mozilla Thunderbird (see Figure 4 for example screenshots). Messages have a standard address scheme and format that will be described in the following sections.

In Figure 4 a web-based email system is also illustrated, namely Gmail. Such web-based systems effectively behave as web-based User Agents allowing users to send and receive email messages over HTTP or HTTPS. It needs nonetheless to be clarified that web-based mail systems utilize the HTTP protocol for communications, but behind the scenes traditional protocols of email systems are being used, i.e. SMTP, POP3 and IMAP as will be discussed later.

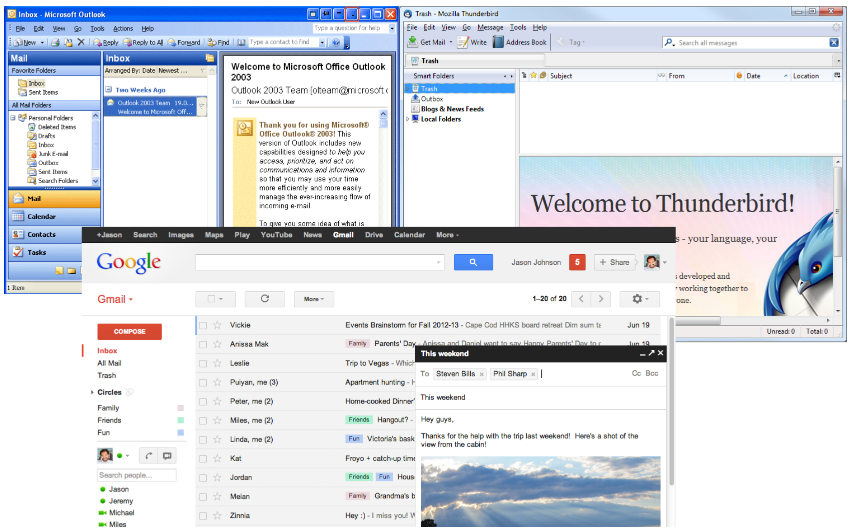


Figure 4: Example screenshots of Mozilla Thunderbird, Gmail and MS Outlook User Agents.

## Email address

Unique addressing is extremely important for an email system to be able to deliver email messages. There exist two parts for standard email addresses separated by the "@" symbol, namely the local part (defines the name of the mailbox of a user where all his/her email messages are stored) and the domain name (DNS[1] name for the mail server that is in charge of a user's mailbox). For example, *user@mail.com* is an email address with a local part having a value of "user" and a domain name "mail.com".

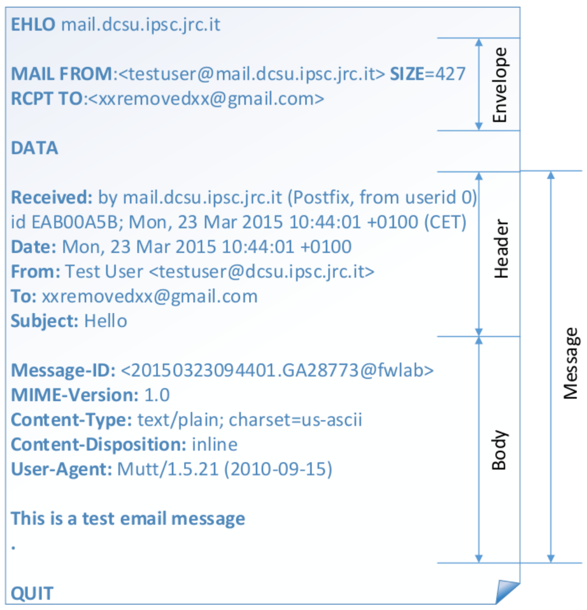
## Email format

Every email is composed of two parts, namely the envelope and the message. The envelope contains information that facilitate the processing of the email. In particular, in the envelope information about the sender and the receiver addresses is present, as well as information regarding the date that the email was sent, when it was received, the content of the message (plaintext or HTML), the reply path, etc.

The message part of the email contains the header and the body. The header declares the sender and the intended receiver of the email, the subject of the message and other information

[1]The Domain Name System (DNS) is a distributed, hierarchical naming system used to identify computing devices over the Internet. It provides support to translate domain names to IP (Internet Protocol) addresses.

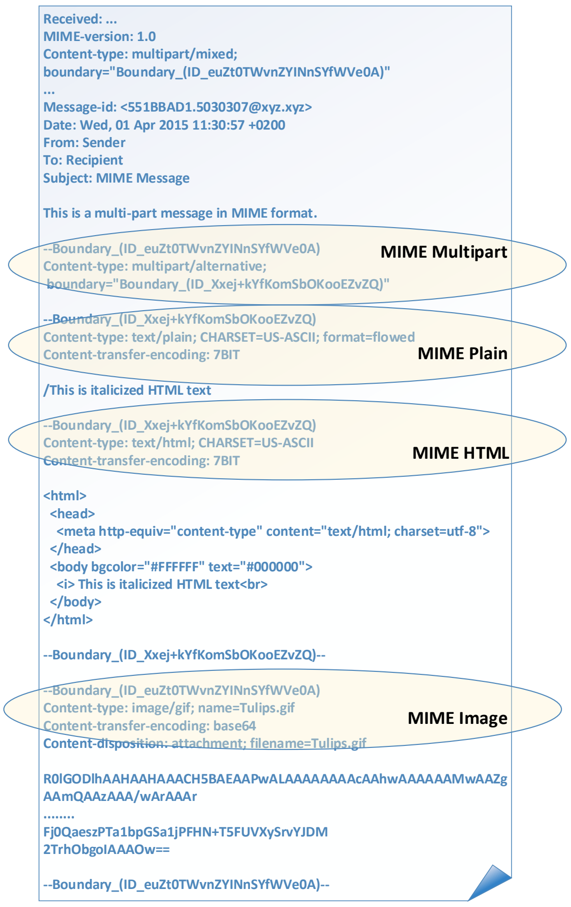
regarding timing. The body contains the actual information that is to be read by the recipient. It is evident that there are repeating fields between the email envelope and the header. Whereas this might seem to be a design flaw, it needs to be specified that the email envelope is meant to be automatically handled by other components of the email systems, namely the Message Transfer Agents, while the email header is meant to be read and processed by the human users. The format of standard email messages is illustrated in Figure 5 .



```
EHLO mail.dcsu.ipsc.jrc.it

MAIL FROM:<testuser@mail.dcsu.ipsc.jrc.it> SIZE=427
RCPT TO:<xxremovedxx@gmail.com>

DATA

Received: by mail.dcsu.ipsc.jrc.it (Postfix, from userid 0)
id EAB00A5B; Mon, 23 Mar 2015 10:44:01 +0100 (CET)
Date: Mon, 23 Mar 2015 10:44:01 +0100
From: Test User <testuser@dcsu.ipsc.jrc.it>
To: xxremovedxx@gmail.com
Subject: Hello

Message-ID: <20150323094401.GA28773@fwlab>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
User-Agent: Mutt/1.5.21 (2010-09-15)

This is a test email message
.

QUIT
```

Figure 5: Format of a typical email message.

This simple, yet expressive, structure of emails spurred its widespread deployment. However, messages formatted as described above can only be 7-bit ASCII format and therefore email content is rather limited. Binary files or attachments of other sort and multi-lingual content are therefore impossible to be sent using this standard email format. For these reasons, MIME (Multipurpose Internet Mail Extensions) was introduced to allow for non-ASCII data to be sent through email systems. MIME is specified in a series of RFCs, i.e. RFC 2045 [29], RFC 2046 [30], RFC 2047 [59], RFC 4288 [26], RFC 4289 [27] and RFC 2049 [28]. Essentially, MIME converts non-ASCII data at the client's side to ASCII data that is sent via email and then converted back to the original non-ASCII data at the server's side. Therefore, MIME facilitates the sending of text that is not in ASCII character sets, attachments (files, images, audio, video, etc.), as well as messages that contain multiple parts, i.e. combination of the latter types. These types are defined in the MIME header, by using the "Content-Type" notation. Of particular interest are multi-part messages that allow for the presentation of the sent data in different ways according to the capabilities of the recipient's User Agent. For example, a message can be sent in both plaintext and HTML and it can be displayed in HTML at the User Agent of the recipient, or in plaintext if the latter Agent does not support HTML content. In terms of security, it is worth noting that there exist two content types referring to digital signature of an email message and its encryption. These content types are defined in a dedicated RFC, i.e. RFC 1847 [32], with an additional noteworthy implementation regarding OpenPGP in RFC 3156 [20]. In the example depicted in Figure 6 , MIME version 1.0 is used to sent a multipart MIME message that includes plain text, HTML and an image attachment to the recipient of the email. The different parts of the MIME multipart are highlighted accordingly, while it is interesting to note the boundaries (i.e. unique identifiers) between them that are used to distinguish between these distinct parts.



```
Received: ...
MIME-version: 1.0
Content-type: multipart/mixed;
boundary="Boundary_(ID_euZt0TWvnZYINnSYfWVe0A)"
...
Message-id: <551BBAD1.5030307@xyz.xyz>
Date: Wed, 01 Apr 2015 11:30:57 +0200
From: Sender
To: Recipient
Subject: MIME Message

This is a multi-part message in MIME format.

--Boundary_(ID_euZt0TWvnZYINnSYfWVe0A)
Content-type: multipart/alternative;
 boundary="Boundary_(ID_Xxej+kYfKomSbOKooEZvZQ)"

--Boundary_(ID_Xxej+kYfKomSbOKooEZvZQ)
Content-type: text/plain; CHARSET=US-ASCII; format=flowed
Content-transfer-encoding: 7BIT

/This is italicized HTML text

--Boundary_(ID_Xxej+kYfKomSbOKooEZvZQ)
Content-type: text/html; CHARSET=US-ASCII
Content-transfer-encoding: 7BIT

<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
  </head>
  <body bgcolor="#FFFFFF" text="#000000">
    <i> This is italicized HTML text<br>
  </body>
</html>

--Boundary_(ID_Xxej+kYfKomSbOKooEZvZQ)--

--Boundary_(ID_euZt0TWvnZYINnSYfWVe0A)
Content-type: image/gif; name=Tulips.gif
Content-transfer-encoding: base64
Content-disposition: attachment; filename=Tulips.gif

R0lGODlhAAHAAHAAACH5BAEAAPwALAAAAAAAAcAAhwAAAAAAMwAAZg
AAmQAAzAAA/wArAAAr
........
Fj0QaeszPTa1bpGSa1jPFHN+T5FUVXySrvYJDM
2TrhObgoIAAAOw==

--Boundary_(ID_euZt0TWvnZYINnSYfWVe0A)--
```

Figure 6: Format of a typical MIME Multipart email message.

### 3.1.2 Server (Receiver)

The receiver or recipient of the email is the person or persons with whom the client wishes to communicate. On this side of the architecture, a User Agent is also utilised to receive emails while additionally having the same functionality as the client's User Agent, namely editing email messages. While the sender of an email message is always one user, the receivers might be more than one. The notion of alias (also known as mailing list) was thus conceived to refer to several email addresses with the use of a pseudo-address. The email system will check each time a message is sent whether it refers to an alias or not. In the former case, the alias will be parsed and the original email message will be sent individually to every email address that is included in the alias.

### 3.1.3 Mail Server

The mail server resides at the core of the email system's architecture. Its functionality comprises two main activities, namely managing users' mailboxes and implementing the delivery of email messages between users. A user's mailbox is a section of a local hard drive at the mail server, where the received email messages for that particular user are stored. A strict permission authorisation system is in place to ensure that only the intended, authenticated user is the one who will have access to its own mailbox. In terms of delivering email messages, we need to classify the procedure according to whether the sender and the receiver are both registered to the same mail server or different ones. In the former case, the workflow is really simple, since upon compiling the email at the sender's side, it is automatically stored at the mailbox referring to the receiver. Since they are both registered in the same mail server, no communication over the network takes place.

Conversely, when the sender and the receiver are registered in diverse, remotely located mail servers, then the communication and delivery of email messages employs a series of additional components other than the User Agents. Specifically, a component titled MTA (Mail Transfer Agent) is employed. The MTA is an entity located at the mail server of both the server and the receiver. When the sender compiles an email to be sent, it is stored in a message queue in its mail server, which parses the queue and sends the email over the Internet to the mail server of the receiver. In practice, the email is delivered to the MTA of the receiver's mail server. The MTA at the client side does not need to be running all the time, merely when a new email appears in the queue. On the other hand, the MTA at the server side needs to be online all the time, listening for new emails that might be delivered. Once the MTA at the server side receives a new email it forwards it to the mailbox of the intended receiver in order for the receiver to read it using its User Agent.

The latter procedure is followed if we assume that the sender and the receiver are directly connected to their respective mail servers. However, this is rarely the case in realistic conditions, and especially when taking into account web-based email systems that promote remote connectivity to mail servers. The User Agent is once again the entry point for the users to send their emails. In this case, the User Agent contacts an MTA client located in his/her local machine[2] and it is this MTA client that establishes a connection to an MTA server entity that is located in the remote mail server. This MTA server in the sender's mail server is running all the time listening for new emails to be forwarded to remote mail servers. When one such email arrives, it is stored in the local queue and similarly to the aforementioned procedure this queue is processed by an MTA client and transported over the Internet to the MTA server at the receiver's side. The latter stores the email to the intended receiver's mailbox. The receiver (assuming that he/she is also not directly connected to his/her mail server) employs an additional set of mail agents, namely the MAA (Message Access Agents). An MAA client is used by the receiver's User Agent to collect email that have been stored in his/her mailbox[3]. To do so, the MAA client contacts the always online MAA server that resides in the mail server of the receiver.

The distinction between MTA and MAA agents lies in the pattern of communication between them. Whereas the MTA operates under the basis of a push architecture (client pushes mails to the server), the MAA functions in a pull-based manner (client contacts server to pull email messages). Furthermore, the notion of relay MTAs has been introduced to cater for multihop, delay-tolerant mail delivery. When the mail servers of the client and the recipient do not reside close to one another, messages need to be transported over multiple hops. MTA relays are

[2]In most cases, the User Agent and the MTA Client at the sender's side are part of the same implementation, i.e. they are functional components of the same application.

[3]The MAA Client and the User Agent at the receiver's side are in most cases part of the same implementation, i.e. they are functional components of the same application.

utilised to store email messages and forward them towards the intended destination, whereas even in the case that the recipient's mail server is temporarily offline such a mechanism can be used to enforce robustness in mail delivery by making repeated attempts to deliver the mail on behalf of the server. In addition, MTA gateways are used to relay mail connections between client and server that reside over distinct networks, even when TCP/IP connectivity is not available in one or more of these networks.

Some of the most well-known implementations of Mail Servers include Postfix, Microsoft Exchange Server, Apache James, Eudora Internet Mail Server, qmail, etc.

## 3.2 Communication Protocols

As mentioned before when describing the architecture of email systems, communication of email messages between senders and recipients takes place with the use of special types of mail agents, namely MTA and MAA. These agents communicate using standard protocols that define the messages that need to be exchanged to establish connections between the agents, initiate communication and email message transfer and finalise connections. The foremost protocol used to define the communication between MTA clients and servers is SMTP (Simple Mail Transfer Protocol). In addition, protocols such as POP3 (Post Office Protocol version 3) or IMAP (Internet Mail Access Protocol) are utilised for the communication between MAA agents. These interactions are depicted in Figure 7 .

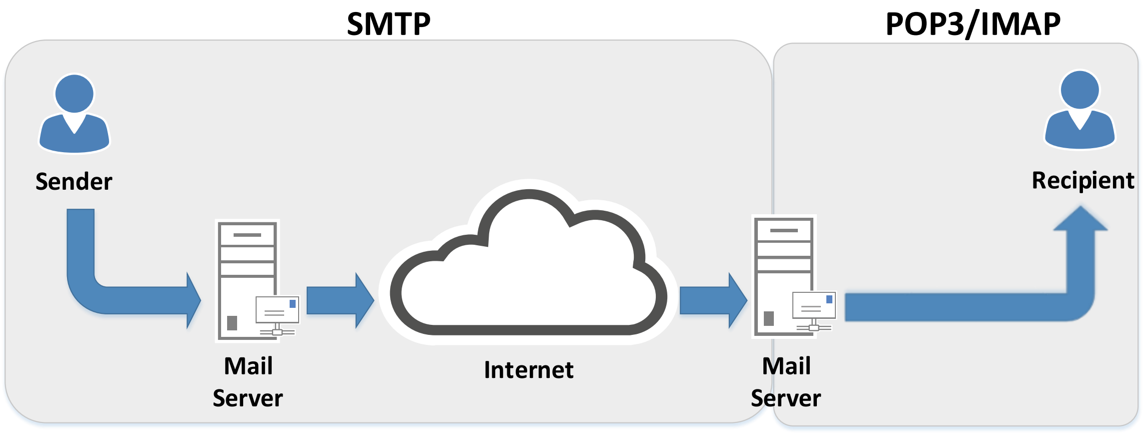


Figure 7: Application domain of different email communication protocols (SMTP, POP3, IMAP).

In what follows, we review the communication protocols involved in email systems in a brief yet comprehensive manner.

### 3.2.1 SMTP

In regard to the email system architecture that we presented in a previous section, SMTP is used during the communication between the sender and the mail server, as well as during the communication between the sender's and the recipient's mail servers. SMTP is a communications protocol that defines the exchange of messages between these entities in order for email messages to be sent and received. It is defined in detail in RFCs and there are many available implementations, the interoperability of which is attributed to the standard nature of the protocol. TCP is used for SMTP message exchanges, with port number 25 used by default, with the exception of email message submissions that commonly utilize port 587. SMTP uses the notion of DNS MX (Mail Exchange) Records to route email messages to their recipients. An MX record holds the DNS settings associated with a user's mailbox, essentially storing the DNS location of the server and instructing on how emails should be delivered. MX records can store information for more than one mail server, to increase resilience. In such a case, priorities are assigned to the mail servers to establish which one will be contacted first. MX records are defined in RFC 1035 [57]. For example, in the case of Gmail, the MX record is as follows (lowest number has highest priority):

*gmail.com mail exchanger = 30 alt3.gmail-smtp-in.l.google.com.*
*gmail.com mail exchanger = 20 alt2.gmail-smtp-in.l.google.com.*
*gmail.com mail exchanger = 40 alt4.gmail-smtp-in.l.google.com.*
*gmail.com mail exchanger = 5 gmail-smtp-in.l.google.com.*
*gmail.com mail exchanger = 10 alt1.gmail-smtp-in.l.google.com.*

In SMTP terminology, commands and responses are the means of communication between two entities. Each command issued by a client needs to be responded to by a server. Commands and responses are formally defined in the RFCs that define SMTP and each one of them is terminated by the use of carriage return and line feed (combination reflecting end of line character). An indicative listing of the most used SMTP commands is illustrated in Table 3.1. They are used to convey SMTP email exchange functionality, such as sending email messages, defining their parameters, and managing connections between client and server. Responses are 3-digit codes with an optional textual explanation; a non-exhaustive list of SMTP responses can be seen in Table 3.2. A 3-digit code starting with 2 denotes a positive reply, whereas if it starts with 4 or 5 it denotes a transient negative and permanent negative reply, respectively.

SMTP is used to define the functionality to transport emails messages among hosts, and not to define the contents of the message. Supporting Internet standards have been defined to deal with the SMTP header and SMTP mail body, namely RFC 5321 [48] and RFC 5322 [72] respectively. Originally, SMTP was ASCII-based and thus did not facilitate the sending/receiving of messages with binary or internationalised content. MIME (Multipurpose Internet Mail Extensions) was therefore introduced to cater for this shortcoming. MIME essentially allows the encoding of binary content to ASCII so that it can be transmitted using SMTP. Further relevant extensions to SMTP have been proposed over the years, with 8BITMIME (RFC 1653 [49]) being one of the most prominent ones used nowadays.

SMTP was first introduced in 1982 (RFC 821 [68]) and since then many updates and extensions have been put forward. The current version of the protocol's implementation is defined in RFC 5321 [48] (2008), with notable updates inbetween including RFC 2821 [47] (2001) and RFC 1869 [50] (1995). The email message submission functionalities are defined in RFC 6409 [33] (2011) and extensible message formats in RFC 3464 [58] (2003).

Support for authentication was introduced in SMTP because the original version did not require clients to authenticate themselves to servers prior to sending email messages. This loose security policy was evidently exploited by spammers and led to widespread propagation of worms.

| Keyword | Description |
|---|---|
| DATA | It is used to send the content of the actual email message. The message finishes with a new line containing one period. |
| AUTH | Authentication as defined in RFC 2554 [63]. |
| HELO | Used by the client to identify itself to the mail server upon initialization. |
| EXPN | Asks the recipient to expand the mailing list that was sent as an argument and return the corresponding unique mailbox addresses. |
| HELP | Requests the recipient to send information about a command that was passed as an argument. |
| MAIL FROM | Used by the client to identify the sender of the email message (passed as argument). |
| NOOP | No operation: sent from the client to the recipient to check its status. Expecting acknowledgement. |
| QUIT | Finalise the email message. |
| RCPT TO | Used by the client to denote the recipient of an email message. For multiple recipients, the command is repeated. |
| RSET | Stops and exits the current email operation, resetting all information and connection to the mail server. |
| SEND FROM | Send directly to the terminal of the recipient and not its mailbox. If the recipient is not online the email bounces back. The email address of the sender is the argument. |
| SOML FROM | Send directly to the mailbox or the terminal of the recipient. The email address of the sender is the argument. |
| SAML FROM | Send directly to the terminal and the mailbox of the recipient. If the recipient is not online the email is delivered only to the mailbox. The email address of the sender is the argument. |
| TURN FROM | Allows the sender and the recipient to change places. Not supported in modern SMTP implementations. |
| VRFY FROM | It is used to verify the address of the recipient (argument passed to the command). |

Table 3.1: List of main SMTP commands with their respective expected arguments.

Building on an initial Internet Draft that was published in 1995, RFC 2554 [63] on "SMTP Service Extension for Authentication" was proposed in 1999. In this respect, authentication was introduced as a service extension as defined in RFC 1869 [50] ("SMTP Service Extensions") enabling the server to inform the client on the authentication mechanisms that it supports. This is done in the response to the HELO message by the client and the supported authentication mechanisms are listed as arguments to the 250 response code of SMTP. Typical authentication mechanisms include (non-exhaustive list): LOGIN (Base64 challenge-based, not formally defined), PLAIN (RFC 2595 [64]), CRAM-MD5 (RFC 2195 [51]), DIGEST-MD5 (RFC 2617 [25]),

| Code | Description |
|---|---|
| **Positive completion reply** | |
| 211 | System status or system help reply. |
| 214 | Help message. |
| 220 | Domain service ready. |
| 221 | Domain service closing transmission channel. |
| 235 | Authentication successful. |
| 250 | Requested command completed. |
| 251 | User not local, so the command will be forwarded. |
| 252 | Cannot verify user, but will attempt delivery. |
| **Positive intermediate reply** | |
| 334 | Server authentication challenge. |
| 354 | Start mail input. |
| **Transitive negative completion reply** | |
| 421 | Domain service not available, closing transmission channel. |
| 450 | Mailbox unavailable. |
| 451 | Local error in processing: command aborted. |
| 452 | Insufficient system storage: command aborted. |
| 453 | No mail available. |
| **Permanent negative completion reply** | |
| 500 | Command not recognised: syntax error. |
| 501 | Syntax error in parameters or arguments. |
| 502 | Command not implemented. |
| 503 | Bad sequence of commands. |
| 504 | Command parameter temporarily not implemented. |
| 534 | Authentication mechanism is too weak. |
| 538 | Encryption required for requested authentication mechanism. |
| 550 | Requested action not taken: mailbox unavailable. |
| 551 | User not local |
| 552 | Requested mail action aborted: exceeded storage allocation. |
| 553 | Requested mail action not taken: mailbox name not allowed. |
| 554 | Transaction failed. |

Table 3.2: List of most common SMTP responses.

Kerberos (RFC 2222 [61]), and GSSAPI (RFC 2222 [61]).

### 3.2.2 POP3

The Post Office Protocol (POP), conversely to the push-style of SMTP, is a pull-based protocol. It is used for the communication between the mail server of the recipient of an email message and the recipient. The protocol's current version is 3, namely POP3. The functionality of POP3 is quite simple and occurs over TCP (standard port used for communication is 110).

POP3 operations are initiated by the POP3 client (recipient of email messages) that contacts the POP3 server (installed on the mail server of the recipient) in order to authenticate itself using the given username/password combination for its corresponding mailbox. Upon proper authentication, authorised access to the mailbox is provided and the client is given the option to list the available email messages in the mailbox, to read and to delete them.

POP3 allows users to either download and delete the message from the mailbox or to download the email message and also keep a copy of it on the mailbox in the mail server. In the former case, the email messages are usually stored in a mail program installed in the recipient's personal computer. Mail is deleted on the server following the client's disconnection. The simplicity of POP3 spurred its growth and popularity among ISPs and mail service providers. It is very lightweight in terms of message exchanges, as well as in terms of the resources required by the mail server, e.g. storage. However, this simplicity comes at a cost, in particular regarding complex operations on the user's mailbox, such as implementation of organisation schemes, e.g. folders. IMAP addresses such concerns, albeit at an increased processing cost and need for additional computational resources on behalf of the mail server.

POP1 was introduced in 1984 (RFC 918 [73]) and POP2 in 1985 (RFC 937 [9]). The current version 3 was first defined in RFC 1081 [76] in 1988, and this has been updated in RFC 1939 [62] in 1996. An extension mechanism was defined in RFC 2449 [34] (1998) and an authentication mechanism in RFC 1734 [60] (1994). To date, version 4 of the protocol has not been published, despite the fact that there have been proposals to alleviate shortcomings such as folder management and multipart message support in future versions of the POP protocol.

### 3.2.3 IMAP

Aiming at addressing notable shortcomings of the POP3 protocol, IMAP (Internet Message Access Protocol) was introduced in 1986 and its second version was defined as an RFC in 1998 (RFC 1064 [14]) and updated in 1990 (RFC 1176 [16]). The current version is IMAPv4rev1 and it has been defined in RFC 3501 [15] (2003). IMAP operates on top of TCP as an application layer protocol, with an established port number 143 and similarly to POP3 is used to retrieve email messages from the mail server to a client (recipient). Contrary to POP3, IMAP allows multiple users to connect to the same mailbox and manage it.

There are two modes of operation in IMAP, i.e. online and offline. Offline operations are synchronized when the client reconnects to the mail server. IMAP has a much more extended functionality, and hence complexity, compared to POP. In particular, IMAP supports folder management on the server, partial email message downloads (based on MIME extensions), advanced mailbox management on the server (e.g. hierarchies, renaming), examining the header of email messages prior to downloading (partial fetch), message state information (using different flags for reading, replying, or deleting email messages).

## 3.3 Communication patterns

In the mail transfer-delivery process one can conceptually consider 3 distinct communication patterns depending on the particular protocols involved and the components that are involved. Related to the standard email systems that we described before we can distinguish between communication among the client and the mail server, as well as the mail server to the recipient.

Moreover, another different communication pattern is that between mail servers when the email message is transmitted over the Internet. As will be discussed later, there is the option for the client to use an end-to-end secure email system (encrypting all traffic), in which case we can identify another communication pattern that is of great importance when examining solutions for secure email.

### 3.3.1 Client to mail server

Communications between clients and the mail server, namely the MTA Client at the client's side and the MTA Server at the mail server's side, is regulated by SMTP, since this is the underlying communication protocol. There are three phases involved during this communication exchange, i.e. connection establishment, email transfer and connection termination.

#### Connection establishment

The client connects over TCP/IP port 25 to the SMTP server and the server responds with SMTP response code 220 (service ready) subject to successful connection or returns code 421 (service unavailable) in case the server is for some reason not operational. The client subsequently sends a HELO SMTP command with its domain name address as an argument to identify itself and its domain to the server. The server responds with an appropriate code, typically 250 for successful completion of a request command. The client then chooses a preferred authentication mechanism out of the list of available mechanisms that was appended to the 250 response and issues an AUTH command to the server, waiting for a server authentication challenge that should arrive as a parameter of a 334 response code. The client upon successful authentication should get a confirmation from the server with a response code of 235. Figure 8 illustrates this exchange of messages. It should be noted that in the exchange of messages that was described, any one of the supported (by the mail server) authentication mechanisms could be utilised. In the example shown in Figure 8 , the CRAM-MD5 authentication mechanism is illustrated. Additional security mechanisms that are employed by the mail server to prevent identity spoofing and spam campaigns include the verification of the MAIL FROM address of the sender, as well as the validation of the IP address of the sender. Other workarounds include approaches like POP before SMTP or IMAP before SMTP, both of which require the user to first download some mail messages via POP or IMAP respectively and thus authenticate himself/herself and then proceed with sending email messages.

#### Email transfer

Subject to successful connection establishment, email messages can be send from the client to the server and then on to their intended recipients. The client commences by sending a MAIL FROM command to indicate the sender of the email message (mailbox and domain name address) to be used as a return email address in case of error or reporting messages. The server will respond with an appropriate code, typically 250 for successful reception of the command. The client then sends the RCPT TO command with the email address of the recipient as an argument. Similarly, the server will respond with 250 and wait for the client to send the DATA command with the actual email message to be transmitted. The server's response code for successful execution of the DATA command is 354, which denotes that it is ready to listen to the mail input. The client will then proceed by sending the message in consecutive lines, each one terminated by an end-of-line token (in the case of SMTP this is carriage return followed

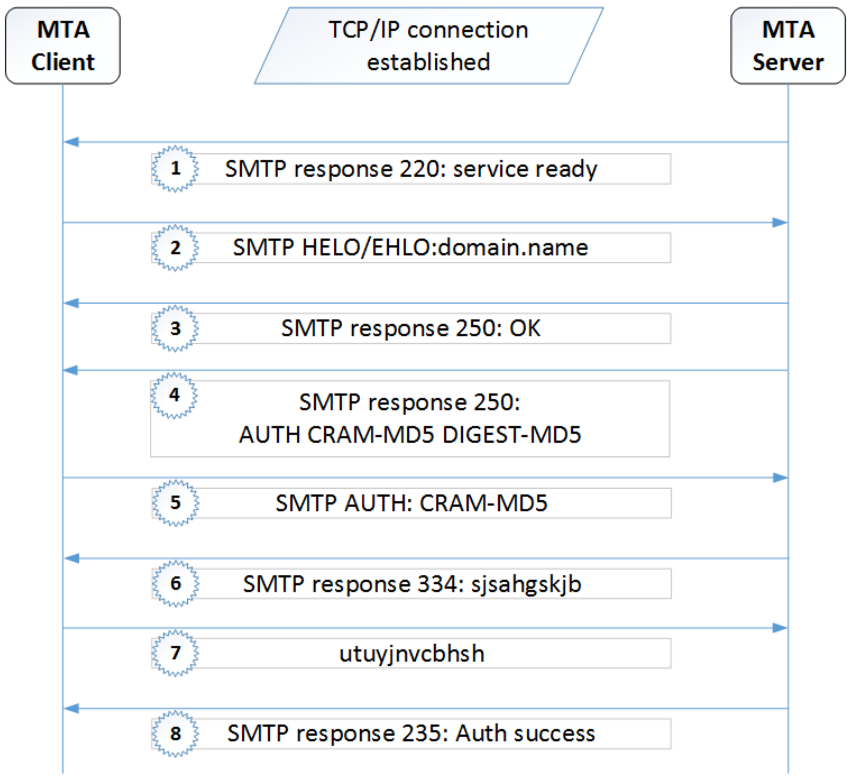

MTA Client
TCP/IP connection established
MTA Server
1
SMTP response 220: service ready
2
SMTP HELO/EHLO:domain.name
3
SMTP response 250: OK
4
SMTP response 250: AUTH CRAM-MD5 DIGEST-MD5
5
SMTP AUTH: CRAM-MD5
6
SMTP response 334: sjsahgskjb
7
utuyjnvcbhsh
8
SMTP response 235: Auth success

Figure 8: Communication exchange between MTA Client and Server during connection establishment.

by line feed). To terminate the message the client will need to send to the server one line containing one period. The server will acknowledge the latter by a 250 response code in the case of successful reception of the mail. Figure 9 illustrates this exchange of messages.

#### Connection termination

To terminate the connection to the server, e.g. because no more email messages need to be sent, the client simply submits to the server the QUIT command of SMTP, which needs to be acknowledged by the server. Typical response code for successful finalization of the connection is 221 (service closed). Figure 10 illustrates this exchange of messages.

### 3.3.2 Mail server to mail server

As seen in Figure 3 , communication between mail servers is essentially communication between the MTA client of one server with the MTA Server of the other mail server. Therefore, it is evident that the communication exchange between mail servers follows the exact same pattern as the one described in the previous section.

A noteworthy particularity is that concerning the role of DNS in the routing of email messages between mail servers, as well as in promoting security in email transmissions. When emails are being forwarded from one mail server to another, the former needs to locate the latter and validate its identity. This is essential, because identity spoofing is a high security risk. In this respect, in order to avoid spamming campaigns and protect against identity spoofing, several

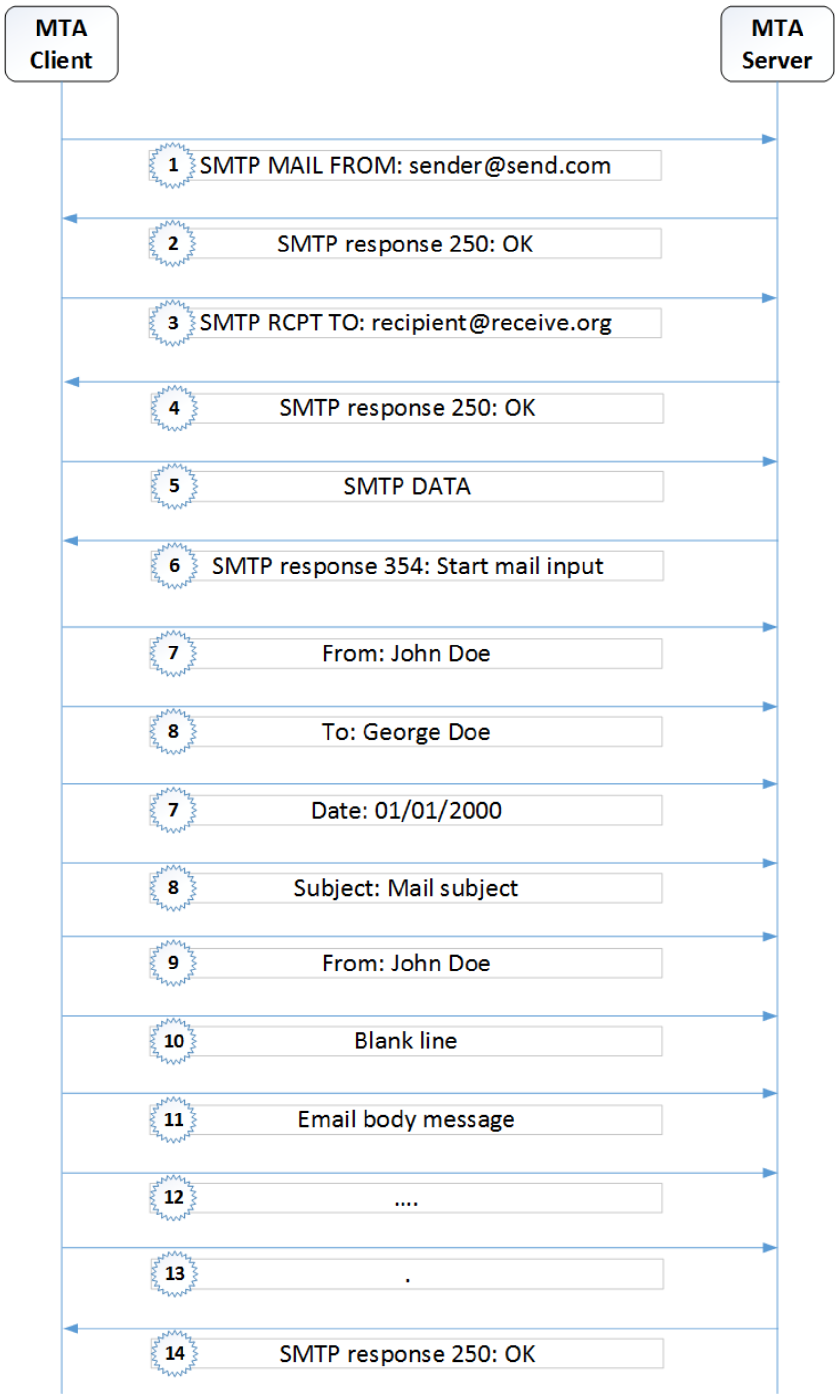


Figure 9: Simplified example of communication exchange between MTA Client and Server during email transfer.

DNS checks have been proposed. The recipient mail server can first of all examine the IP address of the sender (through the TCP connection that SMTP is based on) and validate it against the MX record of the mail server of the sender (the recipient is aware of the domain name of the sender via his/her mailing address and can therefore perform an *nslookup* operation to retrieve the corresponding MX record). The IP address corresponding to the mail server in the MX

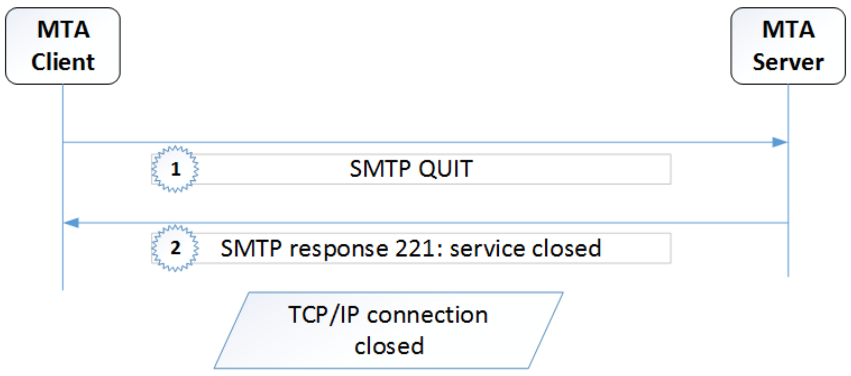


Figure 10: Communication exchange between MTA Client and Server during connection termination.

record should match that of the receiver request, otherwise an identity spoofing attack might be in place. Similarly, reverse PTR records could facilitate protection against spammers and identity spoofing attacks. Malicious users commonly try to imitate legitimate sites by faking their mailing address. When a mail server receives an email message it can check whether the IP address from which it was sent is the same one as that of the PTR (DNS pointer) record of the domain to which the mailing address belongs. The resolution of a PTR record is the process of resolving an IP address to its associated hostname and it is essentially the inverse process of a DNS lookup. If there is no PTR record in the DNS server associated to the sender's mail server, then the process will fail and the email message will not be delivered, e.g. it can be safely considered as spam. Furthermore, recently the notion of SPF (Sender Policy Framework) DNS Resource Records was introduced to detect email identity spoofing. It is defined in RFC 7208 [46] (2014) and it defines for a particular domain the hostnames of its computers that are allowed to send emails. The receiver of an email can then quickly check against the listing to establish whether to accept an email or not, based on whether the sending host is in the SPF record or not. SPF records are stored in TXT files (for backward compatibility) or SPF-formatted ones. Evidently, all these protection measures against identity spoofing are subject to the open, unprotected nature of DNS itself. It is therefore, highly recommended that DNSSEC[4] be employed to ensure the integrity of the transmitted data, including MX, PTR and SPF records.

### 3.3.3 Mail server to server (recipient)

Communication between the recipient of an email message and its mail server occurs using POP3 or IMAP. In the former case, the recipient of the email message will connect to its assigned POP3 server via TCP/IP port 110. Following successful completion, the recipient sends the username for its mailbox on the POP3 server and its password, waiting in both cases for positive acknowledgement. The recipient is then connected to its mailbox and can list all the email messages in it by issuing the LIST command or retrieve an email message by issuing the RETRIEVE command using the email identifier as an argument. A similar communication exchange is followed in the case of IMAP. Figures 11 , 12 illustrate these exchanges of messages for the POP3 and IMAP protocols respectively.

[4]Domain Name System Security Extensions (DNSSEC) involves a series of RFCs that were proposed to secure DNS, i.e. RFC 4033 [3], RFC 4034 [5], and RFC 4035 [4]. It will be discussed in more detail in Section 5.

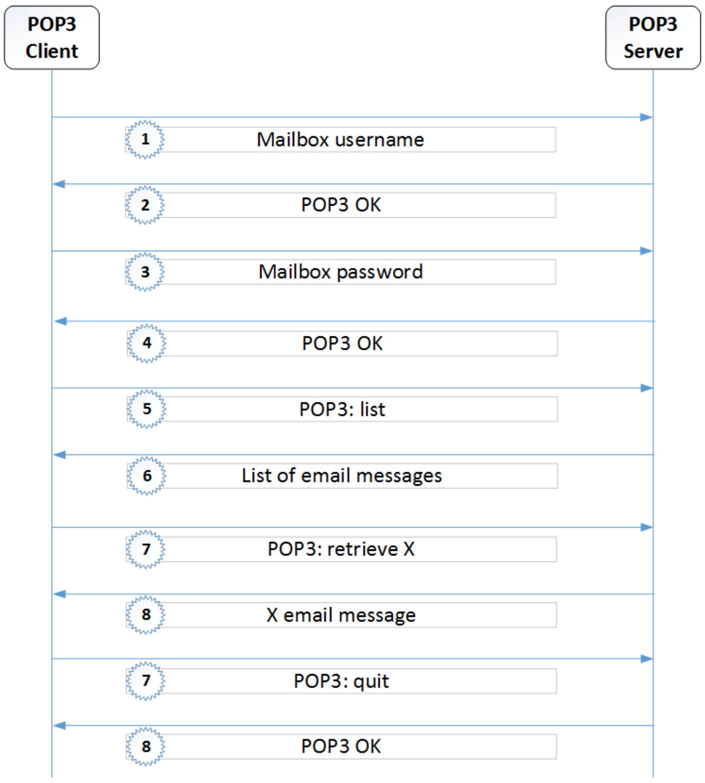


Figure 11: Communication exchange between POP3 Mail Server and POP3 client.

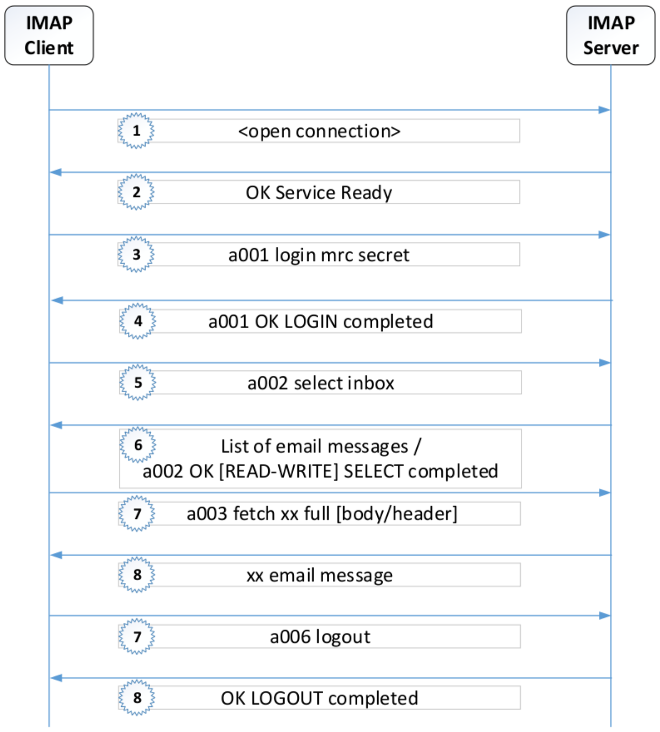


Figure 12: Communication exchange between IMAP Mail Server and IMAP client.

# 4 Threat and Vulnerability analysis of the email system

The aim of this report is to identify and map the existing threats against the citizens' right to privacy in the context of current email systems. Having presented an overview of the operation and functionalities offered by such systems, we detail in this chapter the different threats against them, as well as their vulnerabilities that are being exploited by malicious users and attackers, while we also discuss the most common attack vectors that are being utilised by malicious entities. By means of this comprehensive review and analysis of threats and vulnerabilities, we intend to highlight existing security-related shortcomings of email systems and instigate the discussion on corresponding countermeasures, which is the focus of Section 5.

## 4.1 Threats

The wide deployment and popularity of email systems, combined with the lack of consideration for security at the initial stages of their conception (e.g. open SMTP mail relays that allowed non-authenticated mail submission), have spurred the growth of numerous threats against email communications. In the following, we review such threats and examine different types, such as malware, spam, social engineering, massive eavesdropping and targeted attacks.

### 4.1.1 Malware

The term is derived from malicious software and as indicated by its name describes any piece of software that is malicious in nature and can have a detrimental impact on the security and privacy of computers, data or users. There are a variety of malicious software that can fall under the scope of this definition, e.g. viruses, worms, adwares, Trojans. Malware exploits vulnerabilities of a system to infect it and subsequently it usually operates in the background affecting the operation of the system, monitoring its behaviour and that of its users and therefore directly impacting their privacy. Email systems are highly susceptible to be exploited by malware in two ways. First, email systems themselves can be infected by malware, which could potentially expose all communications to malicious entities. Second, email can be used as an effective method to remotely propagate malware by means of malicious content and attachments.

In the early 90s, the malware ecosystem was primarily composed of computer viruses that used to replicate by attaching themselves to other programs or storage media. The spread of the virus from one personal computer to another would typically take place through the sharing of storage media between users, mainly floppy drives and CDs. As a result of this, viruses were spread locally within distinct geographical regions. With the birth of Internet a new generation of malware emerged, able to propagate remotely at an unprecedented rate through data communication networks and infect computers located thousands of kilometres away. With email being one of the earliest protocols widely used in Internet, it does not come as a surprise that one of the first and most famous Internet worms [74] used it as one of its infection vectors. The so called Morris worm targeted specific vulnerabilities present in the systems involved in email communications in order to spread to new targets.

The Happy99 worm [78] and Melissa virus [79] would follow it in January and March 1999 respectively. The Happy99 worm would become the first modern malware to propagate by email in the form of an attachment. The Melissa virus followed a similar approach and was estimated to have infected up to 20% of computers worldwide. Both malwares ushered in a new era where email communications would become the primary infection vector for a new generation of computer viruses and worms.

In May 2000, a new worm [80] designed to spread over email hit the Internet and became the most virulent malware up to date. The ILOVEYOU worm, named as such due to the subject of the email messages used to spread it, used social engineering to trick the victims into opening a malicious attachment. These viruses and worms would become the first of many other email worms that would follow them in the next years, such as the Nimda worm in 2001 [11] and MyDoom worm [12] in 2004.

Nowadays, there is a wide variety of malware that exploit email in order to find and remotely infect its victims, including Trojans (programs that while behaving in an inconspicuous manner, conceal their malicious activities), rootkits (malware that embeds itself in the operating system to prevent detection), adwares and spywares (malware that aims at pushing advertisements to users or to spy on their behaviour respectively), etc. Malware is not limited to desktop devices, but has recently found its way onto mobile devices by taking advantage of the many security loopholes that exist in the corresponding platforms. Evidently, the threats on email systems become even more prominent in such cases due to the fact that mobile platforms are highly personalized and tightly integrated with user accounts, including their email accounts. In many cases, especially when the malware is linked to a botnet, the attack has a direct impact on the security and privacy of the citizen's email communications.

### 4.1.2 Spam

Spam is defined as the delivery of unsolicited bulk email. Despite the fact that spam might not explicitly appear to be considered as a privacy or security threat, both its implications and its potential impact constitute it a noteworthy threat against email systems. Firstly, the spam threat is well known to impact the functionality of the email service, mainly in terms of availability and usability of the service. Many Denial of Service (DoS) attacks have been mounted based on massive spam campaigns, aiming at overloading email servers so that their proper operations become disturbed or even permanently interrupted. Secondly, spam is closely related to other specific privacy and security threats also described in this section, such as malware and social engineering. Spam messages often contain links to websites that host malware or are part of a phishing campaign.

Furthermore, due to the fact that spam has a very high probability of occurrence and it directly impacts the functionality of the service, it is considered to be one of the main risks of the worldwide email system. For this reason, specific countermeasures are often put in place with the primarily objective of mitigating the spam threat. Since the vulnerabilities and attack vectors used by the spam threat are also the ones used by other more specific privacy and security threats, the specific countermeasures put in place to mitigate spam also help mitigate other related security risks, e.g. authentication and checking of the validity of the sender using various SMTP techniques as mentioned in Section 3.3.2.

In its basic form, a spam message takes the form of a standard email sent to the recipient as part of a massive mailing campaign in order to advertise some products. The number of worldwide spam messages grew exponentially in the 90s and nowadays they are estimated to represent about 85% of the global email traffic [54]. From the perspective of the attacker, spam is a highly profitable business due to the low cost of operations and the monetary return of the investment, even with response rates as low as 0.001%. A comprehensive analysis of the spam business model can be found in [67]. Massive spam campaigns are one of the most frequent uses of botnets, which are a specific type of malware that infects numerous hosts and forces them to perform actions on behalf of a central entity called botmaster.

In an attempt to counteract this threat, a new generation of anti-spam filters were developed at the beginning of 2000. These filters employ machine learning techniques and follow collaborative approaches [7] in order to automatically analyse the emails sent and received by the SMTP servers to detect spam messages. The massive deployment of spam filters triggered a change in the tactics used by the spammers who started to obfuscate the content of the spam messages [52], to perform email identity spoofing and to abuse the vulnerabilities of the SMTP-to-SMTP communications. The usage of these attack vectors by the spammers urged the security community to deploy specific countermeasures to mitigate them.

One of the most recent and effective countermeasures designed to fight spam is the usage of Sender Policy Framework (SPF) records [46]. As described in chapter 3, by using SPF DNS records allow mail exchange servers to detect email spoofing by checking that the SMTP server that delivers the email is in fact a legitimate one. Although the main motivation for the SPF standard is the fight against the spam threat, the protection it offers against email identity spoofing makes it also a versatile countermeasure against other threats such as malware, social engineering and several types of targeted attacks based on email identity spoofing.

### 4.1.3 Social Engineering (phishing, targeted attacks)

The notion of Social Engineering refers to the exploitation of users in order to perform some action that will diminish their security and privacy, e.g. divulge their password or install malicious programs. In the context of email systems, social engineering efforts focus on convincing users to explicitly or implicitly reveal their authentication credentials, i.e. phishing, or to take advantage of users by gaining access to their computing devices where users are already authenticated. The latter case refers to targeted attacks, whereby the attackers are not aiming at affecting random users' email services, but instead they focus their efforts on specific users because they have particular characteristics of interest, e.g. they are employees of a specific company or organization. The fundamental exploit based on which social engineering threats have proved to be widely successful is the inherent trust placed on email systems by their users. Users typically trust their email and therefore are convinced of their validity.

Phishing is the main threat in terms of social engineering in the realm of email systems. Phishing involves a malicious entity sending email messages to users in order to induce them to reveal their credentials [43]. In the most common modus operandi of phishing, these emails appear to be from legitimate entities, e.g. banks, Internet Service Providers, organisations, etc., which request some sort of action on behalf of the users, for example to verify their account by logging in using a provided link. That link would redirect users to a spoofed website that will be used to grab their credentials. In other cases, users are intimidated to provide their credentials by responding to the email message, because if they do not do so they will incur some sort of penalty, e.g. their account will be suspended. There have been a lot of media reports on phishing campaigns and therefore such malicious actions have limited scope, since many users are aware of them and manage to avoid being victims [75]. However, even a very small number of deceived users would yield significant, lucrative gains for the attackers.

Targeted attacks are inherently more advanced in comparison to phishing, since the attackers need to be aware of particularities regarding the potential victims, for example their place of work. This threat is commonly referred to as spear phishing. One of the most notable targeted attacks of this nature is the one that took place in 2011 against RSA Security. During that attack, some groups of RSA Security employees received phishing emails with a malware attachment that allowed attackers to gain access to the corporate network and performed theft of the token seeds thus cracking the security provided by the tokens themselves.

### 4.1.4 Massive eavesdropping

The allegations of mass electronic surveillance formulated by the former NSA subcontractor Edward Snowden [66] constitute a good example of the massive eavesdropping threat. This type of threat is not necessarily linked to governmental surveillance but can also be found in the context of industrial espionage or criminal activities.

Email communications take place in the form of TCP connections over the Internet. Depending on where the sender's and the recipient's email servers are located, the communications can travel thousands of kilometres crossing countries and continents. In doing so, the email will travel through several intermediate systems and communication links. An attacker able to passively monitor any of these systems or communication links will be in a position to massively eavesdrop all the email communications that travel through it. The compromise of a single critical element such as a transatlantic cable or a busy mail server could allow the massive eavesdropping of the emails sent and received by hundreds of millions of users.

There are several scenarios that might fall under the scope of the massive eavesdropping definition. An attacker with an adequate level of resources could eavesdrop a relevant percentage of all the emails sent and received worldwide every day. On the other side of the spectrum, an attacker able to compromise an email server will be able to passively eavesdrop all the emails sent and received by the users of that server.

In both cases, the massive eavesdropping threat will directly impact the confidentiality of the email communications and will do it typically in a passive and transparent way that is difficult to be detected. Section 4.3 will offer some details about the several attack vectors that could be exploited by this type of threat.

### 4.1.5 Other targeted criminal acts

Email has become the de-facto standard to address and identify an individual online. As such, it is also often used as the main vector to conduct all sort of targeted online criminal activities. An interesting example of this type of threat can be found in targeted cyber-attacks aimed at penetrating a corporation by injecting malware delivered by email to an employee [42].

Another notable threat against email systems is the one that aims at collecting contextual information regarding a particular user and then utilizing this information to try to derive that particular user's credentials. The latter targeted type of attack can prove to be quite effective, since users typically devise passwords that they can remember easily and hence they associate them with other personal information, e.g. location or personal preferences.

Moreover, Man-in-The-Middle attacks pose a significant threat to email systems. In such cases, malicious entities intercept email messages between sender and recipient and impersonate one to the other. In this manner, attackers can gain access to sensitive, e.g. financial, information exchanged between users and can therefore have significant gains[1].

An interesting version of this threat involves the actual operation of many email systems, when a user has forgotten his credentials. The user is then normally led to a website that sends back to the user the forgotten password or a link to reset it. Assuming a man-in-the-middle is listening in on this exchange, it is evident that the threat to the protection of users' credentials is great.

## 4.2 Vulnerabilities

### 4.2.1 Integrity of email communications

The lack of protection of integrity of email communications is a major vulnerability that can be exploited by several of the threats described in the previous section. The following vulnerabilities are specific examples of what an attacker could achieve by tampering with the unprotected email communications.

#### Identity spoofing

Identify spoofing is one of the main vulnerabilities inherent to the design of the email system. In an identify spoofing attack an adversary is able to impersonate the identity of a legitimate email user and send emails to third parties on his behalf. In most of the attack scenarios the legitimate user will never realise that someone has impersonated his/her identity.

Even a basic identity spoofing attack could be very hard to detect by the average user, since the email appears to be indistinguishable from what could be a legitimate one. However, very advanced users and security professionals could determine that the source email address was spoofed by analysing the email headers inserted by the SMTP servers involved in the communication.

Detecting more elaborate email spoofing attacks is often infeasible unless specific security measures, such as the ones that will be described in the next chapter, are put in place.

#### Alteration of email content

Similarly to the scenario of the identity spoofing, an attacker could also abuse the lack of protection of the integrity of the data to modify the content of legitimate emails that are sent or received by users.

Even if no identity spoofing is performed and the email received by the user was actually sent by the recipient, it is still possible for its content to have been modified by an attacker. Moreover, this holds true both for the content of the email, as well as for any possible attachments that the email might contain, such as a PDF file.

[1]FBI informed the public on a series of such cases in 2013: http://www.fbi.gov/seattle/press-releases/2013/man-in-the-e-mail-fraud-could-victimize-area-businesses

### 4.2.2 Confidentiality of email communications

The email system assumes that all the actors involved in the communications, as well as the communication links can be trusted and are secure. In reality this is hardly the case. For example, the communication between SMTP servers for email delivery takes place through the public Internet and it is susceptible to be intercepted by third parties.

In practice, this means that in the absence of very specific security measures, such as end-to-end encryption with PGP or SMIME[2], emails sent and received, including the files attached, could be read and copied by third parties. There is no certainty that the communication is private.

## 4.3 Attack vectors

### 4.3.1 SMTP to SMTP server communications

The SMTP to SMTP server communication is considered to be the most vulnerable component of the email system. The original SMTP protocol was built under the assumption that SMTP servers trust each other, so no additional security features were initially built-in into the design of the protocol. Consequently, when a SMTP server contacts another one to deliver a given message, there is an implicit assumption that none of the parties involved in the communication will act in a malicious manner and that the network communication channel is secure.

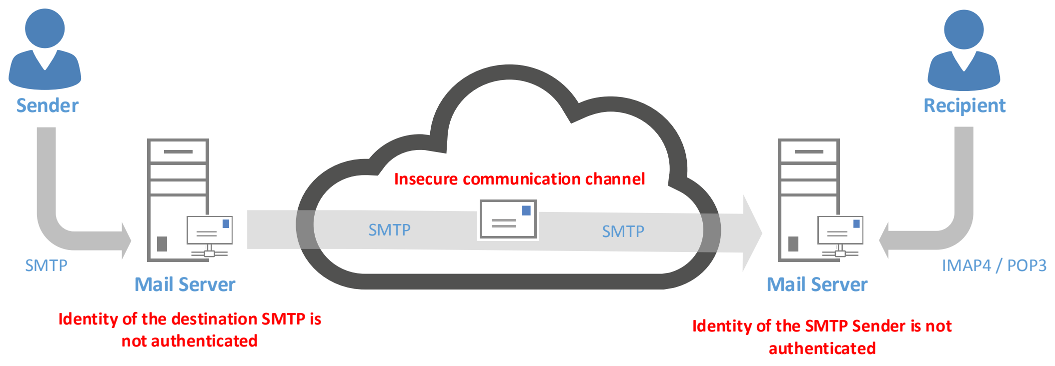

Sender
Recipient
Insecure communication channel
SMTP
SMTP
SMTP
IMAP4 / POP3
Mail Server
Mail Server
Identity of the destination SMTP is not authenticated
Identity of the SMTP Sender is not authenticated

Figure 13: Main vulnerabilities in the SMTP to SMTP communications

In practice, neither of these 2 assumptions is correct. On the one hand, the communication between SMTP servers doesn't take place through secure dedicated channels but through the Internet. On the other hand, the identity of the SMTP servers is not mutually authenticated and the emails requested to be delivered, both for the content and associated metadata, are assumed to be legitimate. Figure 13 depicts the SMTP to SMTP communications and the location of the main vulnerabilities previously described.

The lack of a secure communication channel allows the passive eavesdropping of all the SMTP to SMTP communications provided the attacker is able to retrieve the data while it is in transit. There are several ways the attacker could achieve this, such as passively monitoring data as it flows through intermediate routers or communication links and reconstructing the emails

[2]PGP and SMIME are analysed in detail in chapter 5.

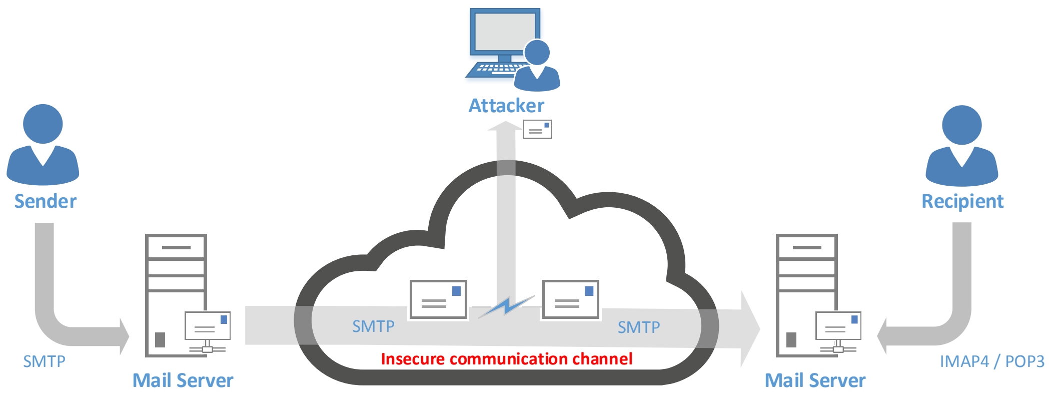


Figure 14: Passive eavesdropping of SMTP to SMTP communications

transmitted following the SMTP protocol. This attack vector is typically the one employed by the massive eavesdropping threat described in section 4.1.4.

An attacker could also employ active means in order to perform a Man-in-The-Middle attack at network level to change the flow of communications to his/her advantage and be able to monitor the communication. An example of this scenario would be the abuse of the BGP (Border Gateway Protocol) to change the routing of IP packets effectively creating a network level Man-in-The-Middle attack [65]. Such an attack vector is also known as BGP hijacking.

Figure 14 depicts how this passive attack is conducted. Although at network level active means might be employed, at SMTP level the attack would be completely transparent since the attacker would simply reassemble the IP fragments (if any), reconstruct the TCP connection and rebuild the emails exchanged on the basis of the captured SMTP transaction in a fully passive manner.

The attack previously described would directly impact the confidentiality of the email communication but not the integrity of the data. However, since the integrity of the SMTP to SMTP communications is not protected, an attacker could also tamper the communication in order to modify the emails that are delivered. In this scenario, an attacker could manipulate the email while it is in transit and change not only the content but also the associated metadata, such as the sender and the recipient.

Figure 15 depicts the active attack previously described. Following this approach, the attacker could modify the contents of an email that has been sent in a way that the recipient would not notice the alteration. In fact, the attack leaves almost no traces at SMTP level making it impossible to detect at the user side. This scenario is depicted in Figure 16 .

In Figure 16 , the attacker manages to receive the outbound SMTP TCP connection of the sender mail server and impersonates the identity of the legitimate SMTP server. The attacker can employ several means to achieve that, such as mounting a network based attack. One effective way to perform such an attack would be the abuse of the DNS protocol which is used by the sender mail server to find the address of the SMTP server in charge of receiving emails for the domain name of the email address to be delivered to, as it was described in chapter 3.

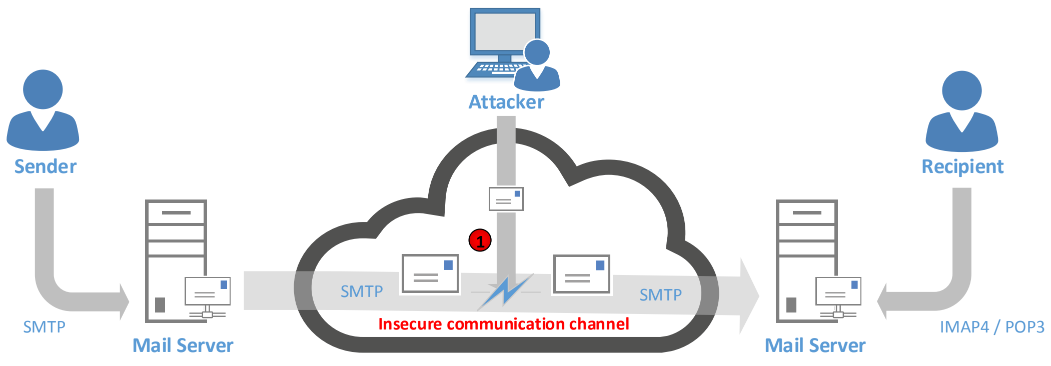


Figure 15: On-the-fly tampering of SMTP to SMTP communications

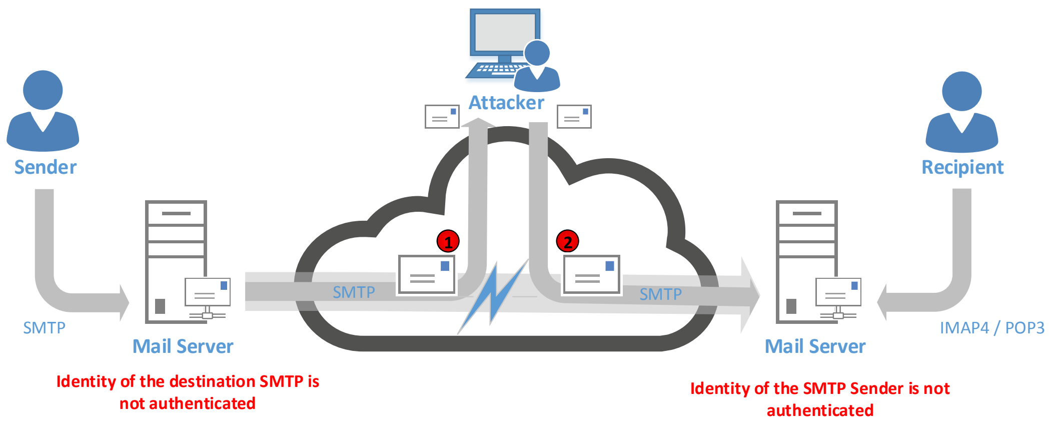

Attacker
Sender
Recipient
SMTP
Mail Server
1
2
SMTP
SMTP
Mail Server
IMAP4 / POP3
Identity of the destination SMTP is not authenticated
Identity of the SMTP Sender is not authenticated

Figure 16: Active interception of email in SMTP to SMTP communications

The attacker could replace the MX record of the destination with its own address, for example performing a DNS cache poisoning attack, effectively convincing the sender that it is the SMTP server in charge. In this case, the attacker would receive the inbound SMTP connection from the sender, who would deliver to him/her the email(s) intended for the legitimate destination.

Similarly, in order to make the attack transparent to the recipient, the attack would impersonate the identity of the sender and connect to the destination SMTP server to deliver the email that was wrongly relayed to him/her. In this process, the attacker would not only compromise the confidentiality of the email communication but potentially also its integrity, since it could modify the contents of the email that was sent, even completely replacing it with another one. The attacker might also decide not to relay at all the intercepted email(s) to the destination. In that case, the legitimate recipient would never receive the email that felt into the hands of the attacker.

Unlike the attack previously depicted in Figure 15 , this attack would leave visible traces in the logs of the SMTP server. Typically the SMTP servers will add some logging information into the email headers that will ultimately be retrieved by the email client of the user. Although this information will not be by default visible to the user, it is possible to visualize it using special options in the email client software. Part of this information will inform about the route followed by the email travelling from the source to the destination. Although the attacker can falsify this information, the final destination SMTP server would still register the IP address of the SMTP server that delivered the email to it. By retrieving this information at the client side and geolocating the IP address of the last SMTP server, it is possible to detect this type of attack.

However, it is worth noticing that the analysis of the SMTP headers at the client side is not considered to be a preventive measure, but just a reactive one. Indeed, this analysis is usually performed manually by trained personnel as part of a security incident response procedure as part of the investigation of a security incident.

The lack of authentication of the sender SMTP server can also be abused in a different manner in order to spoof email identities. In the SMTP protocol there is an implicit trust on the identity of the sender SMTP and all the emails to be delivered that are assumed by the recipient to be legitimate. Consequently, an attacker can easily pretend to be a SMTP server, connect to any other SMTP server and deliver a false email message. Since the attacker controls the complete content of the message to be delivered including the associated metadata, it can easily modify it to spoof the identity of the sender, writing an arbitrary email address and sender's name.

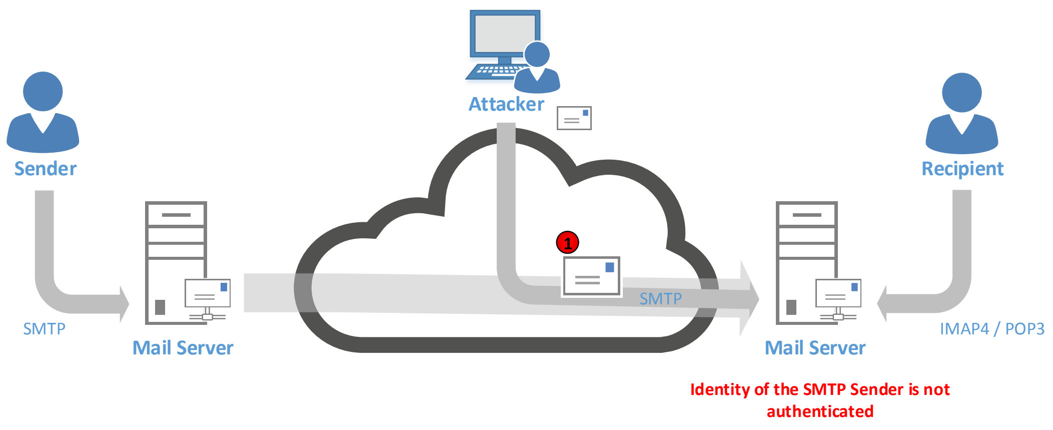


Figure 17: Email identity spoofing attack abusing email in SMTP to SMTP communications

This identity spoofing attack is depicted in Figure 17 . In order to carry out this attack, the attacker is not required to tamper with the network communications of a legitimate email delivery of another SMTP server. Internet IP connectivity and a simple TCP client are sufficient to execute the attack, provided no additional measures are used by the recipient's SMTP. Therefore email identity spoofing is quite easy to perform.

### 4.3.2 User (email client) to server communications

The user to server communication can also be attacked in order to send emails with a spoofed identity, eavesdrop or alter the content of sent and received emails.

In the absence of implicit or explicit SSL, the emails retrieved over POP3 and IMAP protocols can be passively eavesdropped. An attacker able to monitor the IP communication between the user email client and the POP3/IMAP4 server will be able to retrieve the entire content of

```
helo spoofed.com
220 Distinct ESMTP
250 mx3.xxx.xx Hello [XXX.XXX.XXX.XXX], pleased to meet you
mail from: johndoe@spoofed.com
250 2.1.0 johndoe@spoofed.com... Sender ok
rcpt to: destinationuser@xxx.xxx
250 2.1.5 destinationuser@xxx.xxx... Recipient ok
data
354 Enter mail, end with "." on a line by itself
From: John Doe <johndoe@spoofed.com>
To: Destination User <destinationuser@xxx.xxx>
Subject: This is a test of a spoofed email

I actually never sent this email...
.
250 2.0.0 t2KA24qm009803 Message accepted for delivery
quit
221 2.0.0 mx3.xxx.xx closing connection
```

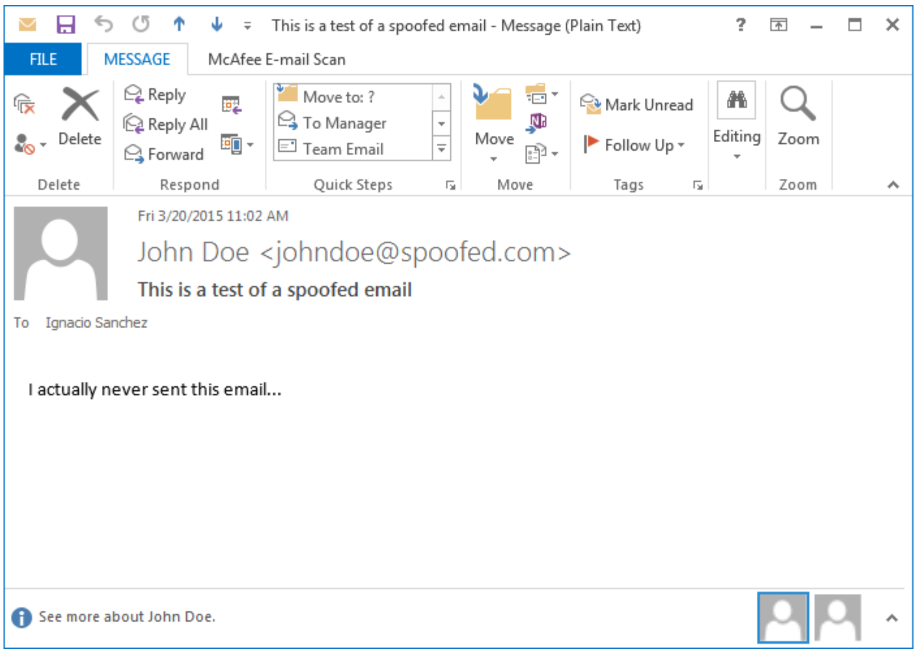
Figure 18: Example of a TCP session used to send a spoofed email

the emails retrieved by the user. Furthermore, the attacker would also be able to retrieve the username and password of the user that is transmitted over the network as part of the POP3 and IMAP authentication process. Once in possession of this information the attacker could directly connect himself to the server and fully impersonate the user. In those cases, such as IMAP4, where a copy of the emails is always stored in the server, the attacker would be able to remotely retrieve all the emails ever received by the victim.

Figure 20 illustrates this attack depicting a scenario where the legitimate user accesses his/her inbox from a mobile device connected to the Internet through a public Wi-Fi hotspot. The protocol used for the communication is IMAP4 with login/password authentication. An attacker located nearby is able to monitor all the IP communication of the legitimate user as it flows through the public Wi-Fi hotspot. Using a sniffer able to parse the IMAP4 protocol, the attacker can retrieve the username and password of the victim, as well as the emails retrieved from the server. In a second step, the attacker can fully impersonate the victim reusing the username and password and remotely dump all the emails stored in the inbox of the server.

The same attack can be conducted against the SMTP protocol used by the email client of the user to send email, provided no SSL is used to secure the communication. If the email clients transmits username and password information as part of the SMTP server authentication process, the attacker will, similarly to the previous case, be able to eavesdrop them passively and fully impersonate the user sending email from his identity.

In addition to the attack vectors previously described, the lack of security in the SMTP and POP3/IMAP communications can also be exploited in more exotic ways, such as transparent replacement of email content or spoofed email injection directly to the client application as it retrieves the emails from the server's inbox.



Fri 3/20/2015 11:02 AM

John Doe <johndoe@spoofed.com>

This is a test of a spoofed email

To Ignacio Sanchez

I actually never sent this email...

See more about John Doe.

Figure 19: Spoofed email visualised by the recipient using Microsoft Outlook 2013

### 4.3.3 Email data storage

The security of the servers, including the respective SMTP, POP3 and IMAP daemons, plays an important role in ensuring the security and privacy of the email communications. This is particularly true for the POP3/IMAP email server that stores the users' inbox. That server contains the emails sent and received by all the users of the domain. If the server gets compromised, either by a remote attack or a local attack (e.g. malicious intent of one of the administrators), the attacker can get a copy of all the users' emails and he/she would also be in a position to modify them at the server.

Furthermore, if the username/password database of the server gets compromised, the attacker would be able to fully impersonate legitimate users, sending emails from their identities and reading all their emails.

In addition to the POP3/IMAP server, the emails are also stored at the client's device, which could either be a computer or a mobile device such as a smartphone. In both cases, if the client's device security gets compromised, the attacker could retrieve all the emails, as well as modify their content. There are several possible ways this can happen. For example, the client device could be infected by malware and the data can be retrieved remotely. It may also happen that the device gets lost or is physically stolen.

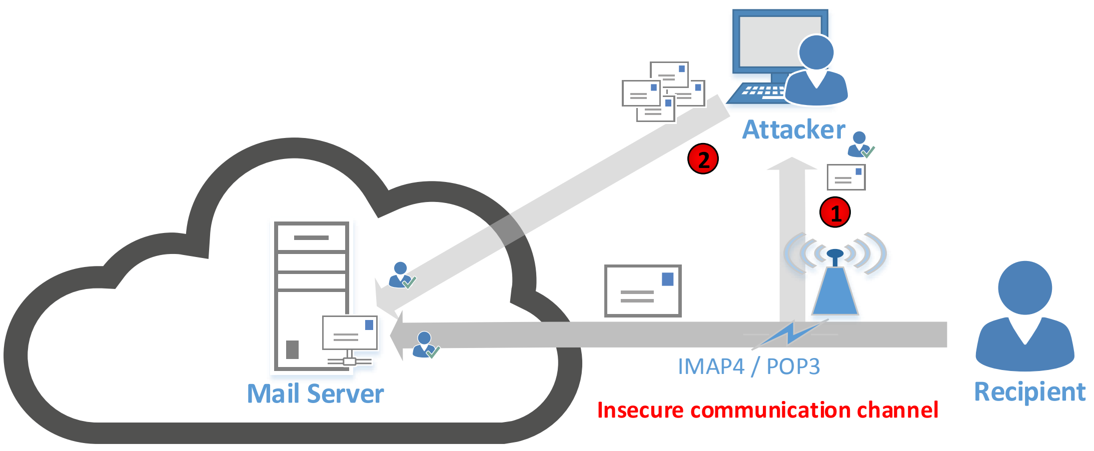


Figure 20: Passive eavesdropping of email client's POP3/IMAP communications over a public Wi-Fi hotspot

# 5 Privacy and security countermeasures

Several cryptographic protocols and tools have been proposed to address the threats and vulnerabilities described in Chapter 4. The purpose of these tools is to ensure both the authenticity of the sender and of the mail relays, as well as the confidentiality and integrity of the message. The existing countermeasures can be split in two categories. The first family provides security and authenticity to the communication by putting in place a secure channel to transmit the messages. The second family focuses on the protection of the messages themselves achieving what is called *end-to-end encryption*. In both cases, cryptographic algorithms are used to provide the different security properties. We briefly introduce in the next section some basics about cryptography before presenting in detail the specific countermeasures.

## 5.1 Cryptography Overview

Historically ensuring the confidentiality of messages has always been of prime importance, especially in the military domain. With the advent of new means of communication, confidentiality becomes crucial to protect the privacy of citizens. This property can be achieved by encrypting the communications effectively transforming them into non-intelligible messages to all but the intended recipient. In addition, the protection of the integrity of the communications and the identity of the sender is equally important. Similarly to manuscript signatures in the physical world, digital signature algorithms provide these two security properties. In this section, we briefly present how these cryptographic protocols work. Interested readers should refer to the "Handbook of Applied Cryptography" [53] to obtain more technical details.

### 5.1.1 Encryption Algorithms

As all cryptographic algorithms, encryption protocols can be split in two main families, namely symmetric and asymmetric algorithms. In symmetric cryptography, a secret value (or a set of values), known as key, has to be known by all the participants that want to share messages. This key is given as input to the encryption algorithm together with the clear text, also called *plaintext*, in order to obtain the encrypted message, also called *ciphertext*. The recipient(s) of the message uses the same secret key to retrieve the original message from the ciphertext. While being generally computationally efficient, symmetric algorithms require that each user generates and exchanges through secure channels a unique secret key with every person he/she wants to communicate with. As communication channels for email communications are generally not considered to be secure, a secure exchange of keys is not feasible in practice.

Until 1976, no solutions other than symmetric cryptography were available. Keys were most of the times exchanged during physical meetings, a practice that is not compatible with the modern digital world. Diffie and Hellman introduced a solution in 1976 [18] presenting the concept of public-key cryptography, also called asymmetric cryptography. Instead of a single shared key, each user is associated with a pair of keys, a public one that has to be distributed to all the persons the user wishes to communicate with and a private key that must be only known by this user. As depicted in Figure 21 , a message is encrypted using the recipient's public key. The recipient can retrieve the original message using his private key.

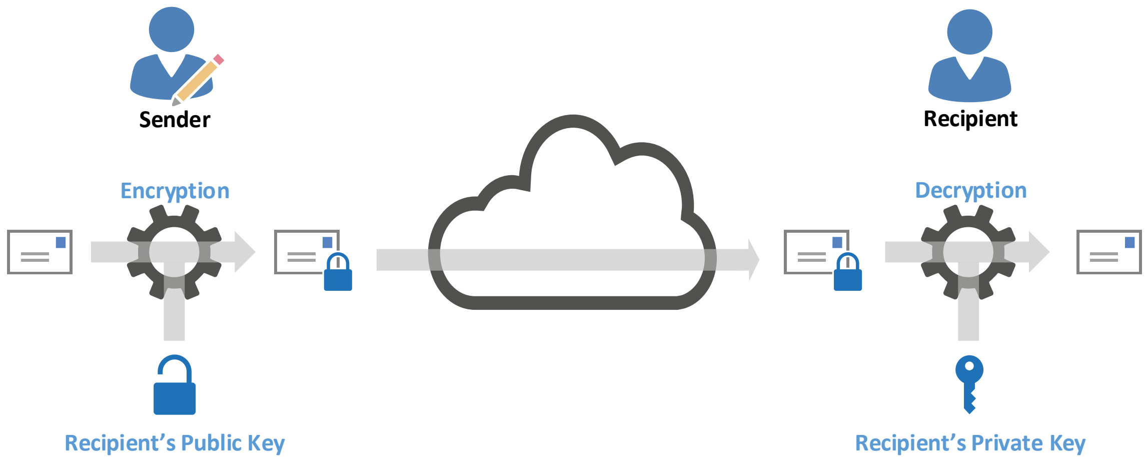


Figure 21: Asymmetric Encryption Algorithm Principle

The security of the protocol relies on the impossibility to retrieve the private key of a given user. Consequently, the public key can be sent through an insecure channel without compromising the confidentiality of the messages. The price to pay is that asymmetric algorithms require more computation than symmetric ones and are thus less efficient.

One way to address this performance issue is to use hybrid encryption. This principle that is depicted in Figure 22 combines the efficiency of symmetric cryptography with the convenience of asymmetric cryptography. To encrypt a message, a sender generates a disposable symmetric session key (step 1 in Figure 22 ) and uses it to symmetrically encrypt the message to be sent (step 2 in Figure 22 ). Afterwards, using the public key of the recipient, the sender encrypts the session key with the asymmetric algorithm (step 2 in Figure 22 ). Upon reception of the messages, the recipient can retrieve the session key thanks to his private key (step 3 in Figure 22 ) and finally decrypts the message using the symmetric algorithm (step 4 in Figure 22 ). Obviously, such solution is only meaningful if the message to encrypt is bigger than the symmetric key used[1] otherwise the message could be directly encrypted using the asymmetric algorithm.

### 5.1.2 Key Exchange Algorithms

Another widespread protocol enabling the usage of symmetric cryptography without a pre-established secret key is the key exchange protocol firstly introduced by Diffie and Hellman in [18]. The aim of this protocol is to allow two actors to agree on a common secret communicating over an insecure channel. In a nutshell, both participants exchange a portion of secret hidden in a *mathematical container* that cannot be extracted by anyone eavesdropping over the channel. Each actor is able to insert its own portion of secret in the received container leading to the same result on both sides, namely the shared secret key. Once the shared key has been established, both peers can use it with the symmetric encryption algorithm to encrypt their communication.

[1]Nowadays, symmetric keys are recommended to be at least 128-bit long [83].

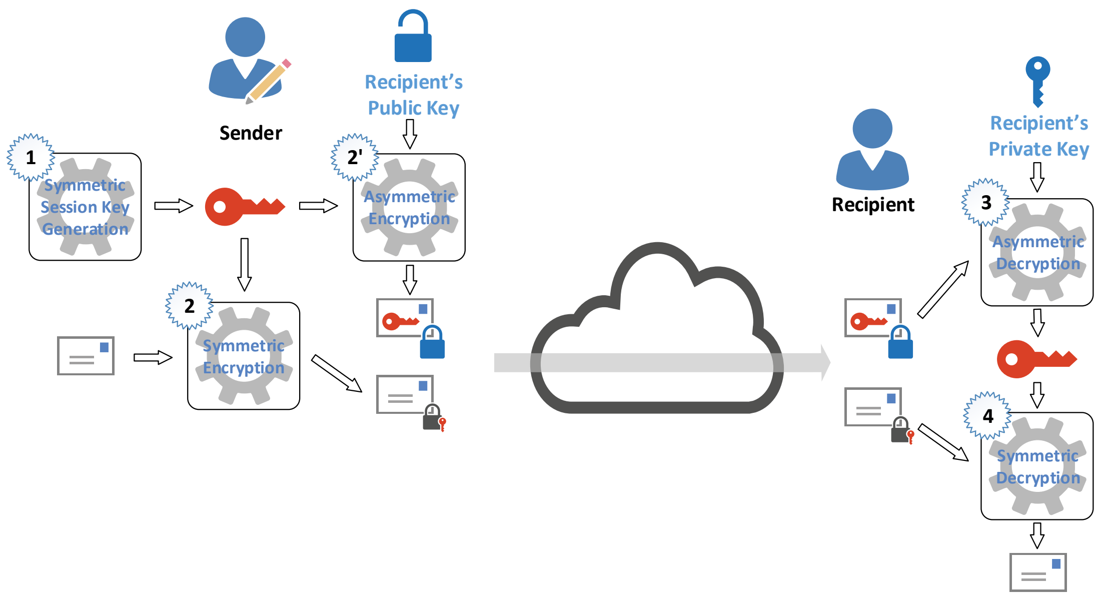


Figure 22: Hybrid Encryption Algorithm Principle

### 5.1.3 Signature Algorithms

While encryption algorithms provide confidentiality of the messages, they do not guarantee that the sender is the person he/she pretends to be. Furthermore, they do not protect from a Man-in-The-Middle attack aiming at modifying the message, regardless whether it is encrypted or not. These two properties are ensured by signature algorithms that can be seen as the digital equivalent of the manuscript signature. In addition to authenticating the author of a message, signature protocols aim at ensuring the non-repudiation property, meaning that the holder of a signing key cannot claim that a signed message was not issued by himself/herself. Such property cannot be reached in symmetric cryptography as all the secret key holders could issue such "signature". These symmetric protocols, called Message Authentication Code (MAC), only protect the integrity of the messages.

The basic principle of signature is depicted in Figure 23 and is briefly described in the following. The signer uses his/her private key to sign the message and sends both the message and the signature. The recipient uses the signer's public key to verify the validity of the couple message - signature. It is considered infeasible to forge a signature on behalf of someone without knowing the corresponding private key. As a consequence, modifying a single bit of a signed message invalidates the signature, ensuring at the same time both authenticity of the author and integrity of the message.

### 5.1.4 Certificates

The rise of asymmetric cryptography helped to solve many practical problems and ensure many security properties without requiring the prior establishment of a secret key through a secure channel. However, the security of the whole communication system relies on the fact that the users are sure of the identity associated to a public key. If such link is not secured, one can

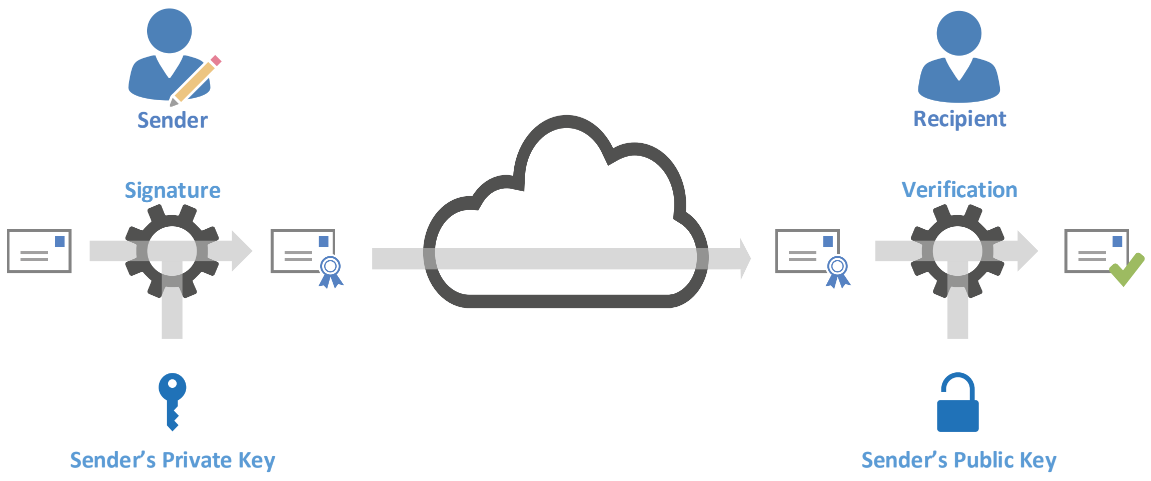


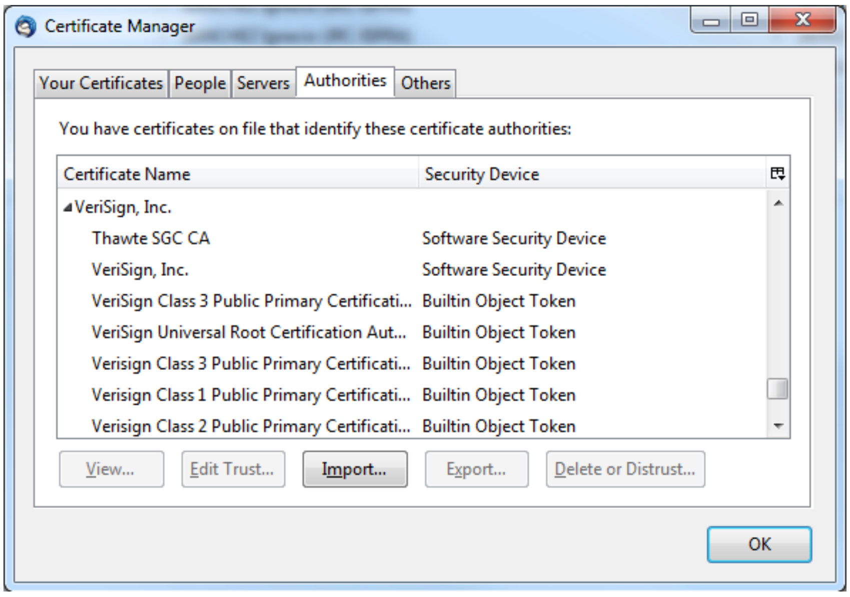
Figure 23: Signature Algorithm Principle

still trust that a message has been signed by the holder of the corresponding public key but not that the holder is necessarily who he/she claims to be.

The most obvious way to trust the identity of a public-key holder is to physically exchange such keys, which unfortunately brings us back to the same limitations of symmetric cryptography. To address this issue, public key infrastructures have been put in place where users trust a single entity or a group of entities who certify the link between the identity and the corresponding public key. Such proof as well as other relevant information are packaged together in a *certificate*. A widespread format is the X.509 certificate defined in the RFC 2459 standard [41] (updated in RFC 5280 [13]). The certificates are composed of the following fields:

- Version and serial number of the certificate;
- Algorithm used to sign the certificate;
- Distinguished Name of the Certification Authority that has issued the certificate;
- Validity period (starting and ending date);
- Distinguished Name of the holder;
- Public Key details, namely the public key algorithm and the public key of the holder;
- Issuer Unique Identifier (only in X.509 v2);
- Subject Unique Identifier (only in X.509 v2);
- Extensions (only in X.509 v3).

Evidently, the certificate by itself does not provide any additional evidence about the link between identity and public key. To secure this link, the certificate has to be signed by an entity trusted by the user. There are many ways that such trust systems can be implemented. The most transparent one for a user is the one deployed in Internet browsers (Internet Explorer, Firefox, Chrome, etc) and mailer systems (Microsoft Outlook, Thunderbird, etc) that is based on a *chain of trust*. A chain of trust is a multiple tree-based solution, where each root is a trusted Certification Authority (CA). The certificates of these authorities are locally stored on the computer in such a way that the software will consider as legitimate any certificate signed by them. An example of some trusted CAs stored locally by the mailer system Thunderbird is depicted in Figure 24 .


Certificate Manager

Your Certificates | People | Servers | Authorities | Others

You have certificates on file that identify these certificate authorities:

| Certificate Name | Security Device |
|---|---|
| ◢VeriSign, Inc. | |
| Thawte SGC CA | Software Security Device |
| VeriSign, Inc. | Software Security Device |
| VeriSign Class 3 Public Primary Certificati... | Builtin Object Token |
| VeriSign Universal Root Certification Aut... | Builtin Object Token |
| Verisign Class 3 Public Primary Certificati... | Builtin Object Token |
| Verisign Class 1 Public Primary Certificati... | Builtin Object Token |
| Verisign Class 2 Public Primary Certificati... | Builtin Object Token |

Figure 24: Example of Certification Authorities Trusted by Thunderbird

These authorities can delegate their trust to intermediate CAs allowing them to also issue end-users certificates, as well as perform other trust delegation. Such hierarchical structure is depicted in Figure 25 . Each node of the trees is associated to a certificate signed by the parent node.

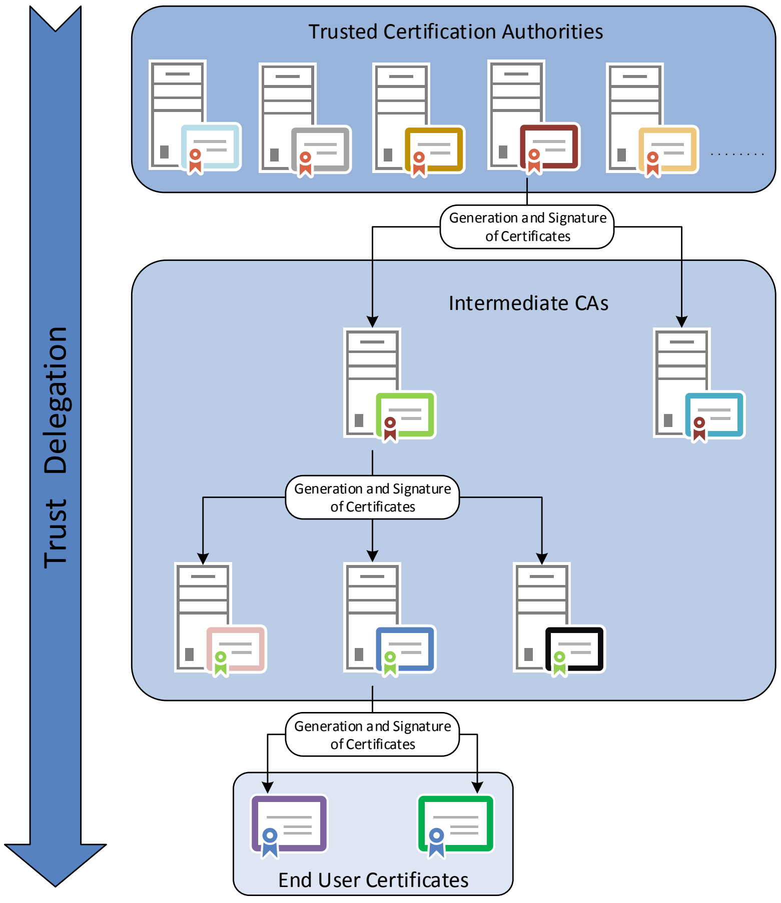
Thanks to the chain of trust, a user can link an end-user certificate to a trusted root CA verifying the validity of each certificate going up to the root of the tree in a stepwise process. Such verification process is represented in Figure 26 for the verification of a signed mail. We assume in this example that the certificate associated to the public key was sent together with the message. The user verifies the validity of the message's signature and checks sequentially all the certificates starting from the signer one up to the trusted root CAs. If a single signature is found to be invalid or if there is a problem with one of the certificates (e.g. an outdated certificate or a non-trusted root CA) the signature of the message will be considered as invalid.

## 5.2 Securing the Transport Layer

The SMTP protocol was designed to provide an efficient and interoperable way to exchange electronic mails. The confidentiality and authenticity of the communications were not initially considered to be a priority and consequently the core set of email protocols were not designed with strong security requirements in mind. With the rise of awareness of the email security and privacy risks several solutions were proposed in order to enhance the security and privacy of email communications. In this section, we describe several solutions that have been proposed to secure the communication channel, also called the transport layer, between the several actors involved in email communications.

### 5.2.1 Secure Sockets Layer and Transport Layer Security

The Secure Sockets Layer (SSL) protocol, defined in the RFC 6101 [31], and its successor the Transport Layer Security (TLS) protocol, defined in the RFC 5246 [17] (and updated in



Trust Delegation
Trusted Certification Authorities
........
Generation and Signature of Certificates
Intermediate CAs
Generation and Signature of Certificates
Generation and Signature of Certificates
End User Certificates

Figure 25: Hierarchical Public-Key Infrastructure

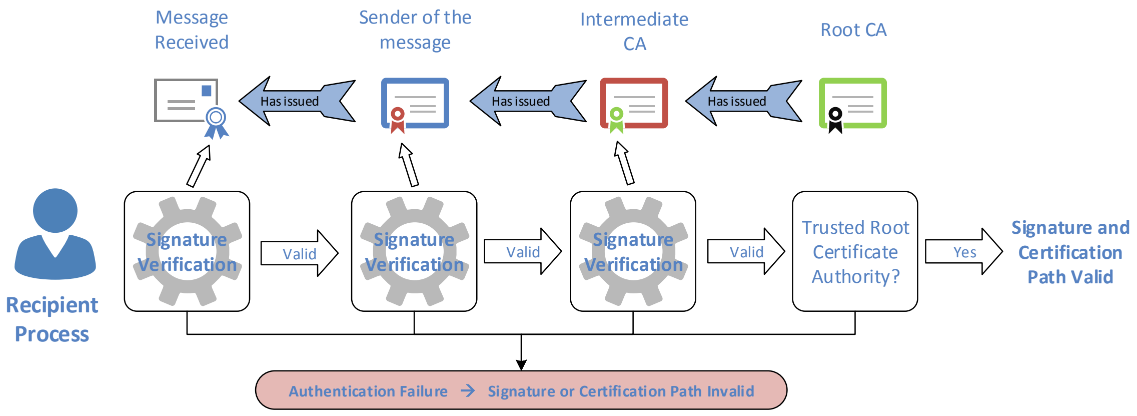


Figure 26: Verification Process of a Signature in a Hierarchical PKI

the RFC 6176 [81]), are two methods to encrypt the communication channel using symmetric encryption. The required symmetric key is determined during the start of the session following a key exchange algorithm based on asymmetric cryptography. The great advantage of TLS based solutions is that they are independent of the protocol used at the application layer. They can be applied in two different manners. One possible approach, known as implicit TLS, is to directly start TLS as soon as the TCP connection between two clients has been established. The other alternative way, known as explicit TLS, consists of establishing the secure channel at a later stage upon reception of a special command at the application level.

In both cases, at the initialisation of the protocol the client and the server engage in a handshake protocol whose purpose is the agreement of the encryption algorithm and of the symmetric key that will be used to encrypt the rest of the session. To do so, the client sends the list of algorithms supported and the server will select the one that it decides to use. The list of algorithms provided by the client shall be a subset of the global list of encryption algorithms supported by the standard. For the sake of backwards compatibility, the client or server may support algorithms defined in previous versions of the standard, as long as they are not explicitly forbidden by the current standard. For example, for security reasons TLS v1.2 is not any more backwards compatible with the standard SSL v2.0. In the next phase of the TLS initialisation process the server certificate is received and verified by the client.

### 5.2.2 Implicit SSL/TLS

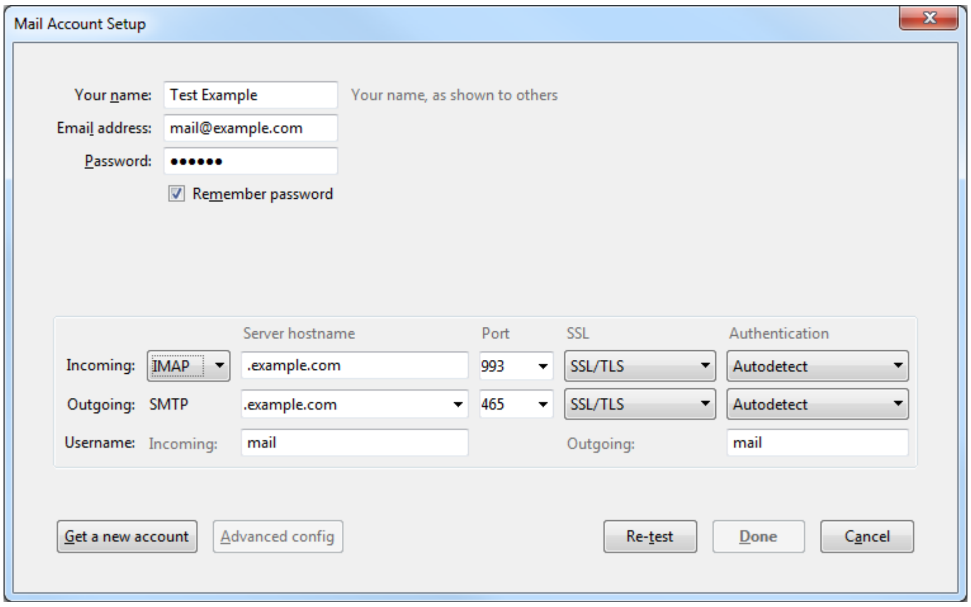
In the implicit TLS configuration, a dedicated port is specifically allocated for TLS/SSL secure communications. Non TLS based communications will not be allowed on that port. This approach is similar to the one followed in HTTP with the definition of HTTPS in the RFC 7230 [21]. In 1997, the Internet Assigned Numbers Authority (IANA) registered the port 465 for SMTPS in order to provide implicit TLS for the SMTP protocol. However, the protocol as such has never been published as an official SMTP transmission channel by the IETF and it was deprecated in 1998 when STARTTLS was published. Ports 993 and 995 were respectively assigned to IMAPS and POP3S for similar purposes but like SMTPS they are not supported by any standard.

Nevertheless, implicit TLS is a widespread protocol commonly used to secure the SMTP, POP3 and IMAP communications between the email client software running on the user's devices and the respective email servers. Due to this fact, email software often include support for SMTPS, POP3S and IMAPS. For example, when creating a new account in Thunderbird the user can choose to use implicit SSL/TLS connection as depicted in Figure  27 .

In any case, after the TLS channel has been established, the server will authenticate the user. This authentication can be performed in several ways, the one based on username and password being the most used one. Since the secure channel has already been established, TLS will protect the transmission of the username and password information to the server.

### 5.2.3 Explicit SSL/TLS

The other approach to transport layer security refers to the establishment of a secure channel upon reception of a specific command at the application level, in SMTP, IMAP or POP3 protocols. In the case of email communication this is implemented in the STARTTLS protocol, as defined in the RFC 2595 [64] for the IMAP and POP3 protocols, and in the RFC 3207 [36]


Mail Account Setup

| | | |
|---|---|---|
| Your name: | Test Example | Your name, as shown to others |
| Email address: | mail@example.com | |
| Password: | •••••• | |
| | [✓] Remember password | |

| | | Server hostname | Port | SSL | Authentication |
|---|---|---|---|---|---|
| Incoming: | IMAP | .example.com | 993 | SSL/TLS | Autodetect |
| Outgoing: | SMTP | .example.com | 465 | SSL/TLS | Autodetect |
| Username: | Incoming: | mail | | Outgoing: | mail |

Figure 27: Implicit SSL/TLS Settings with Thunderbird

for the SMTP protocol. Contrary to the implicit approach, explicit TLS/SSL uses the same port to support the secure communication. If both peers support SSL or TLS a secure channel will be set up over the same port to secure the rest of the communication.

The STARTTLS protocol works as an extension of existing protocols. In the case of the SMTP protocol for example, a mail server adds its encryption capabilities upon reception of an EHLO command. As displayed in Figure 28, the line “250 STARTTLS” of the answer to the EHLO command is meant to inform the sender that the server is capable of setting up a TLS channel. If the client also supports TLS, he will reply to the server with the STARTTLS command effectively starting the TLS protocol initialisation.

### 5.2.4 Limitations

Although at first glance it might seem that the TLS protocol addresses the requirements of confidentiality, integrity and authenticity of the communications, the effective mitigation of these risks is limited by the compromises made in the practical implementation of the protocol in order to maintain interoperability between email systems. The gaps opened by this trade-off between interoperability and security lead to several important weaknesses that will be addressed in this section.

The usage of implicit TLS implies the adoption of two mutually exclusive security policies, namely ”use TLS” or ”don’t use TLS”. According to RFC 2595 [64], the desirable security policy must be ”use TLS when available” in order to maintain interoperability. Consequently, if only one of the two actors involved in the protocol (IMAP, POP3 or SMTP) supports TLS, the strict application of the ”use TLS” policy would fail. In practical terms, implicit TLS is only used for the SMTP, POP3 and IMAP communications between the email software that runs on the user’s device and the respective email servers. In this scenario the decision to support implicit TLS is taken by the administrator in charge of the management of the servers and it becomes a requirement for all its user community. Alternatively, the server can also be

```
Server                                                  Client

220 mail.dcsu.ipsc.jrc.it ESMTP
                                 EHLO mail-wg0-f47.google.com
250-mail.dcsu.ipsc.jrc.it
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
                                                     STARTTLS
220 2.0.0 Ready to start TLS
                                                [Client Hello]
[Server Hello containing the certificate
 and the cipher capabilities]
[Initalisation of the key exchange]
                                         [Client Key exchange]
[Encrypted handshake message]

[Encrypted SMTP Data]
```

Figure 28: Negotiation of a STARTTLS Communication

configured to simultaneously support other protocols in addition to implicit TLS, so that the decision to use it will lie on the user who configures the client software.

For implicit TLS to be effective, the server certificate shall be ultimately signed by a root CA which is trusted by the client software in charge of validating the signature. The client will also verify that the common name of the certificate matches the Full Qualified Domain Name (FQDN) that was used to configure the connection and that the certificate has not expired or has not been revoked. In the scenario where the server certificate is self-signed or has been issued by an unknown root CA, the client will refuse to connect unless it is specifically configured to ignore the server certificate validation. In this case, although the connection will be encrypted, it will be vulnerable to a Man-in-The-Middle attack where the certificate of the server will be replaced on the fly by one generated by the attacker which will be ultimately accepted by the client. Figure 29 shows how the official Android email client can be configured to ignore the certificate validation.

In the case of explicit TLS, since the beginning of the SMTP protocol communication is not yet secured by TLS, an active Man-in-The-Middle attack can target the the EHLO request sent by the client and strip off from the answer of the server the line where the support for TLS is declared. As the client cannot check the integrity of the answer received by the server, the missing information will go unnoticed and the client will continue in clear text assuming that TLS support was not available. This downgrade attack is feasible given the requirement for interoperability and the usage of an insecure channel for the exchange of the server capabilities information. Downgrade attacks against STARTTLS can also be conducted in other ways against SMTP-to-SMTP communications, such as injecting RST TCP segments to SMTP connections [24] thus exploiting the wrong requirement for interoperability between email systems.

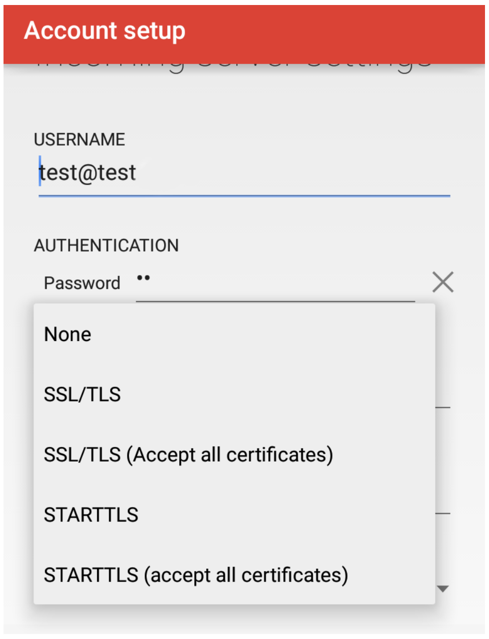


Figure 29: Security Choices in Android Accounts Settings

Similarly to implicit TLS, explicit TLS is also vulnerable to active Man-in-The-Middle attacks in those scenarios where the server certificate is not signed by a root CA trusted by the client. Given the strong requirement for interoperability in SMTP-to-SMTP communications, the client is likely to proceed with a STARTTLS connection even if the server certificate was a self-signed one or the certificate exhibits other problems. In this scenario, although the STARTTLS connection will provide some degree of protection against passive eavesdroppers, following an opportunistic encryption approach, the communication will be vulnerable to an active Man-in-The-Middle attack where the certificate of the server is swapped on the fly to another one owned by the attacker.

Evaluating the usage of STARTTLS or the proportion of strict certificate verification is not an easy task considering the huge amount of SMTP servers available on the Internet. Michael Adkins from the Facebook company published an interesting study [2] of STARTTLS usage by analysing the email communications made to deliver the large quantity of mail sent everyday by Facebook. The results of this study are thus mainly related to personal mail server and not corporate nor governmental one. We underline some highlights of this study.

In 40% of the cases, STARTTLS was not advertised by the SMTP server and the mail was consequently sent not encrypted. Out of the remaining 60% that was encrypted, 30% successfully achieved a strict validation of the certificate. As for the other 30%, the certificate verification failed due to several reasons, mostly because the certificate was self-signed or because there was a mismatch between the host name and the certificate.

In their transparency report [35], Google provides information about the proportion of encrypted email sent and received by their email service, known as Gmail. According to the report 80% of the outbound SMTP traffic is encrypted in contrast to a 55% of the inbound one. However, the report does not include information about the strict verification of the validity of the certificates. Figure 30 shows the evolution of the STARTTLS adoption for SMTP-to-SMTP

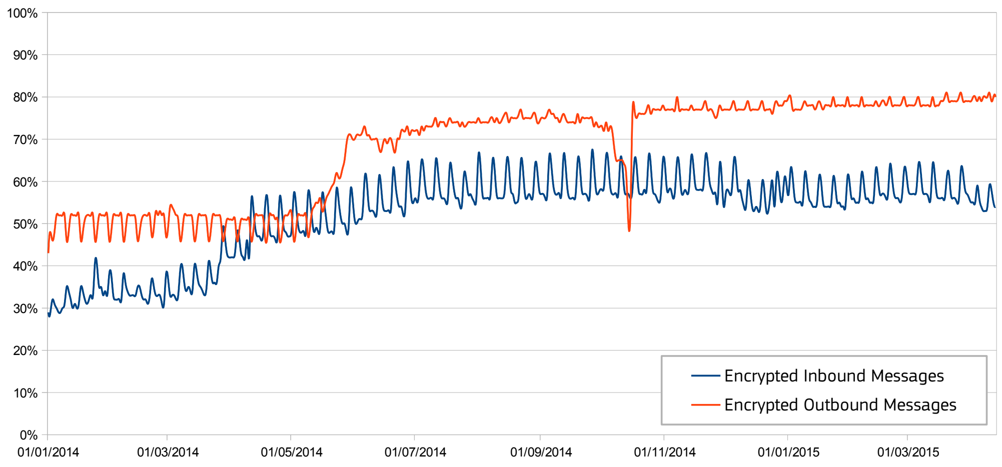


Figure 30: Evolution of STARTTLS adoption based on the raw data published by Google

communications for the last 2 years.

It is worth noting that even though the analysis of the percentage of emails delivered through SMTP connections using STARTTLS and strict validation might provide assurance about the confidentiality of these past communications, it is certainly difficult to extrapolate these results for future communications. Even in the scenario of STARTTLS with strict validation, the communication might still be vulnerable to active Man-in-The-Middle attacks able to downgrade the connection to clear text, as described previously. Moreover, there are also some risks related to the usage of old versions of SSL or TLS and certain encryption algorithms. For example, TLS 1.0 and 1.1 maintain the compatibility with SSL v2.0 that should not be any more considered as secure. The updated version 1.2 of TLS was defined in RFC 6176 [81] and it removes this backward compatibility with SSL v2.0. Nevertheless, TLS v1.2 is still not widespread enough leaving the door open to exploit these weaknesses. There are also other related risks (e.g. the usage of weak keys like the "export key" as described in [8], or the existence of "non-trustable trusted CA") worth analysing when evaluating the security of email communications as a whole.

Finally, it is important to highlight that both implicit and explicit SSL/TLS are meant to protect the communication channel between the several actors involved in email communications. In practical terms, this countermeasure helps in mitigating security and privacy risks related to the confidentiality of email while it is in transit over the Internet.

As such, SSL/TLS offers no protection against the other risks identified in chapter 4 that were not related to the confidentiality of the email communications, such as identity spoofing. It is also worth noting that the email can also be eavesdropped (confidentiality) and altered (integrity) in places other than the communication links. For example, emails could be eavesdropped while they are processed by the SMTP server or stored in the IMAP server of the destination in the event the security of these systems was compromised or the administration were acting in a malicious manner.

### 5.2.5 Possible Solutions

The biggest challenge to address some of the vulnerabilities described in the previously section for SMTP-to-SMTP communications, lies in the provision of authentication and integrity since the beginning of the communication, including the early phases of the protocol. This is already accomplished by the implicit TLS connection but it could also be achieved in the explicit one through the signature of the answer to EHLO request. The two solutions become an effective countermeasure in mitigating eavesdropping attacks against the email communication channel if both the validity of the TLS certificate is strictly verified and there is a proper trust delegation system in place.

A hierarchical PKI as defined in Section 5.1.4 could be used for this purpose. Another solution is provided by DNS-based Authentication of Named Entities (DANE) described in the proposed standard RFC 6698 [37] by the IETF. DANE offers the possibility to store TLS certificates directly as DNS records in the DNS server that handles the domain where the SMTP server operates, effectively binding the server certificate to its FQDN. Thanks to the Domain Name System Security Extension (DNSSEC) infrastructure described below, senders could collect the associated TLS certificate at the same time they resolve the MX records of the recipient's domain. An underlying chain of trust ensures the user of the link between the FQDN and the TLS certificate.

DNS is a hierarchical distributed system that associates information within a domain name. This protocol defined in the early 80s in RFC 882 [56] (and updated many time since then, the last version being RFC 6895 [19]) is a crucial protocol ensuring the interconnection of all Internet nodes and services. The most straightforward usage of DNS is the resolution of domain names into numerical IP addresses. DNS responses are traditionally not cryptographically signed, leading to many vulnerabilities that can be exploited in several ways in order to conduct Man-in-The-Middle attacks and Denial of Service.

The Domain Name System Security Extensions (DNSSEC) modifies DNS to add support for cryptographically signed responses making it infeasible to modify or withdraw a DNS record without invalidating its signature. New types of records have been introduced in the first standard RFC 2535 [1] (and updated in the latest version [4]) with DNSSEC to include these new elements. The DNSKEY record contains the public key needed to verify all the records of the corresponding zone. The RRSIG field contains the signature of the corresponding record using the private key associated to the one in the DNSKEY record. The public keys of the child zones of a DNS server are also stored and signed in a DNS record. This last field builds a full chain of trust from the root DNS server to the leaves of this hierarchical structure.

A simplified version of the resolution of a DNSSEC request is depicted in Figure 31 and it works as follows. Let us assume that a user needs to obtain the MX records of the SMTP server test.example. The DNS resolver first contacts the DNS root to obtain the DNS records of .example. The DNS root returns the corresponding records together with the public key of the Top Level Domain (TLD) .example. All these fields are signed by the root DNS. As the DNS resolver knows and trusts the public key of the root, it can validate the authenticity of all the fields it received. At this point the DNS resolver contacts the TLD to obtain the DNS records of the second level domain test.example. Again it receives the records together with the public key of this parent domain with all these fields being signed by the TLD. Even though the DNS resolver does not trust by default the public key of this DNS server, it knowns and

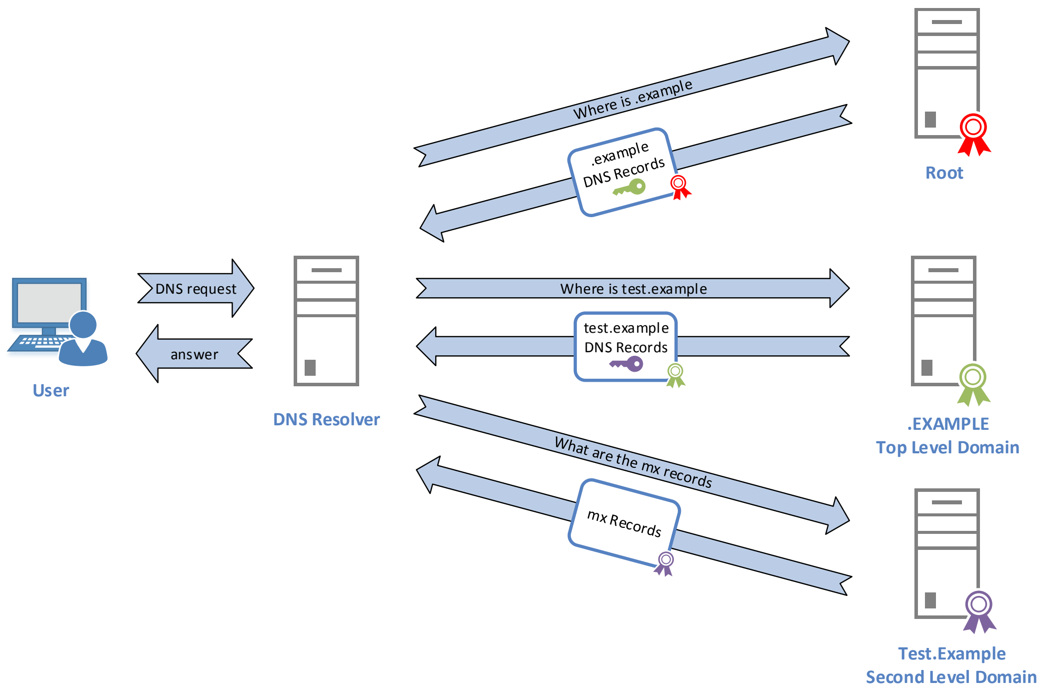


Figure 31: Resolution of a DNSSEC Request

trusts the key received from the root server and can use it to verify the authenticity of the other records. Finally, it contacts the last DNS server to request the MX records that are sent back signed by the second level domain. Based on the same chain of trust the DNS resolver can verify the validity of the answer and then forward it to the user.

The DNSSEC solution provides many advantages and prevents untrustworthy signers from compromising anyone's key except those in their own sub-domains. The feasibility of a wide deployment of this solution will be studied in the next steps of the project. A similar approach called DNSCurve has been developed by Daniel Bernstein [6] and will be studied as well.

Securing SMTP-to-SMTP communications at the transport layer could become an effective protection of the confidentiality and authenticity of emails while they are in transit over IP networks. It also has the great advantage of being fully transparent from the point of view of the users. In combination with DNSSEC, the solution would not only be robust but could also become an effective protection against email identity spoofing attacks.

A trade-off between interoperability and security is inevitable at this point. An effective implementation of this solution designed to prevent active Man-in-The-Middle attacks that are able to downgrade the security of the transport layer, would also cause interoperability problems for those systems not fulfilling these minimum security requirements. For this reason, new solutions have been proposed in an attempt to find a balance between security and interoperability, such as [24].

## 5.3 End-to-End Countermeasures

The security of email communications can also be ensured directly at the protocol layer that manages the message instead of the specific communication channel. In end-to-end security, the email message is encrypted and/or signed at the sender's machine before it is delivered over the network and it will only be unencrypted and/or verified by the recipient once it has arrived to the email client software. Therefore, even though the email might be intercepted while travelling through communication links or when it is stored by an intermediate server, the encryption and signature will protect its confidentiality and integrity. Only the recipient will be able to decrypt it and detect any malicious alteration or identity spoofing by verifying its signature.

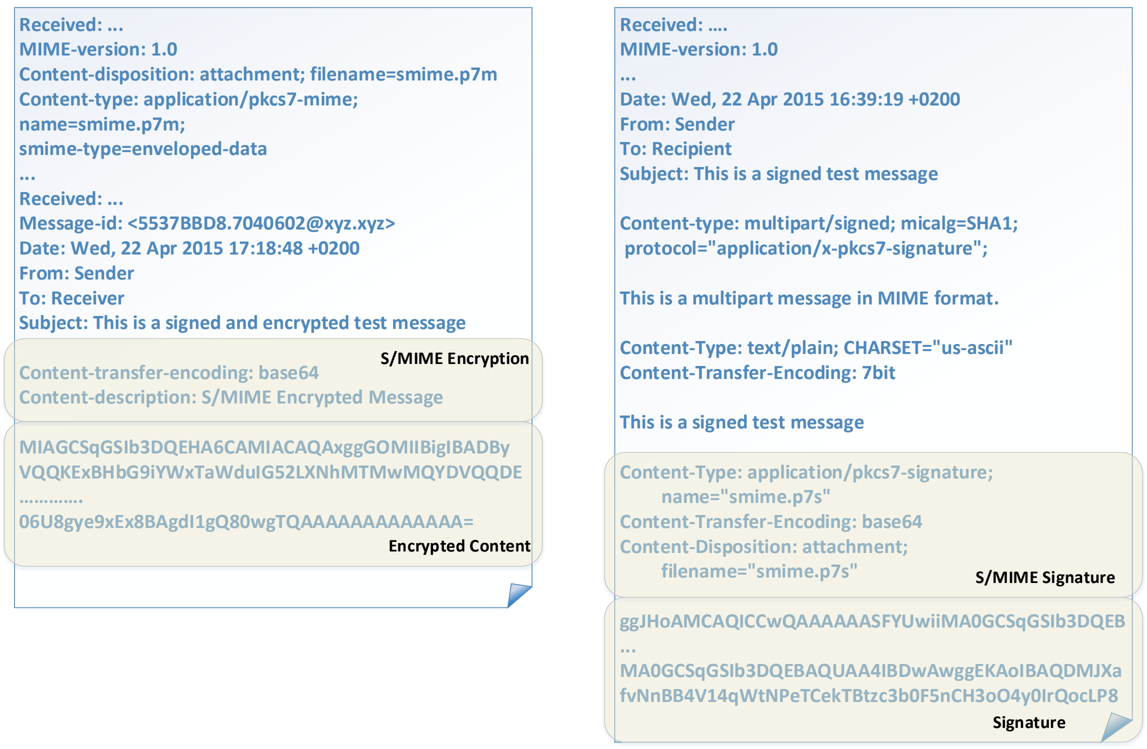
End-to-end email security has a broader scope than transport layer security, effectively mitigating a wide variety of threats. However, unlike transport layer security, end-to-end solutions are not completely transparent and require the intervention of the user or his/her system administrator for corporate environments. The usage of this type of solutions involves the installation and configuration of special software and the deployment of cryptographic keys.

### 5.3.1 S/MIME

As described in section 3.1.1, Multipurpose Internet Mail Extensions (MIME) is an Internet standard extending the mail format to handle character sets other than ASCII. It proposes a uniform structure of messages to include non-text content, as well as digital signature and encrypted content. However, the standard does not recommend any specific security requirements on how to protect the confidentiality, the integrity and the authenticity of the communications. RSA Data Security Inc has extended the MIME standard by including such type of properties using cryptographic mechanisms from the PKCS#7 standard RFC 2315 [45]. This solution has been standardised under the name of Secure/Multipurpose Internet Mail Extensions (S/MIME) leading to a track of standards and discussions from the IETF, most importantly the RFCs 2634 [38], 3369 [39], 3370 [40], 3850 [69], 3851 [70] and 5751 [71].

The standard specifies the two fields *multipart/signed* and *multipart/encrypted* and gives recommendations about the cryptographic algorithms that should be used. The encrypted content and the signature are included in the email as separate parts, as it can be seen in the examples displayed in Figure 32 . The right part of the figure corresponds to a signed email while the left part shows an email that is both encrypted and signed.

The underlying Public-Key Infrastructure in S/MIME is a hierarchical PKI as defined in Section 5.1.4 that makes use of X.509 certificates. To be able to encrypt a message to someone, the sender must know in advance the certificate of the recipient or to be able to retrieve it from a trusted source (e.g. in a public address book). In the case of a signature, the signer can attach its certificate to the signed mail. The chain of trust of the hierarchical PKI provides to the users the capacity to verify the validity of the certificate upon reception of a signature or before sending an encrypted mail. Client email software are generally pre-installed with a predefined list of trusted CAs that can be manually extended by the user or the system administrator managing the workstation of the user. If the root CA that has issued a certificate is not considered as a trusted one, the software raises a security exception asking the user whether the certificate can be trusted or not.



```
Received: ...
MIME-version: 1.0
Content-disposition: attachment; filename=smime.p7m
Content-type: application/pkcs7-mime;
name=smime.p7m;
smime-type=enveloped-data
...
Received: ...
Message-id: <5537BBD8.7040602@xyz.xyz>
Date: Wed, 22 Apr 2015 17:18:48 +0200
From: Sender
To: Receiver
Subject: This is a signed and encrypted test message
Content-transfer-encoding: base64
Content-description: S/MIME Encrypted Message
MIAGCSqGSIb3DQEHA6CAMIACAQAxggGOMIIBigIBADBy
VQQKExBHbG9iYWxTaWduIG52LXNhMTMwMQYDVQQDE
............
06U8gye9xEx8BAgdI1gQ80wgTQAAAAAAAAAAAAA=
```

```
Received: ....
MIME-version: 1.0
...
Date: Wed, 22 Apr 2015 16:39:19 +0200
From: Sender
To: Recipient
Subject: This is a signed test message

Content-type: multipart/signed; micalg=SHA1;
 protocol="application/x-pkcs7-signature";

This is a multipart message in MIME format.

Content-Type: text/plain; CHARSET="us-ascii"
Content-Transfer-Encoding: 7bit

This is a signed test message

Content-Type: application/pkcs7-signature;
      name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
      filename="smime.p7s"
ggJHoAMCAQICCwQAAAAAASFYUwiiMA0GCSqGSIb3DQEB
...
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDMJXa
fvNnBB4V14qWtNPeTCekTBtzc3b0F5nCH3oO4y0IrQocLP8
```

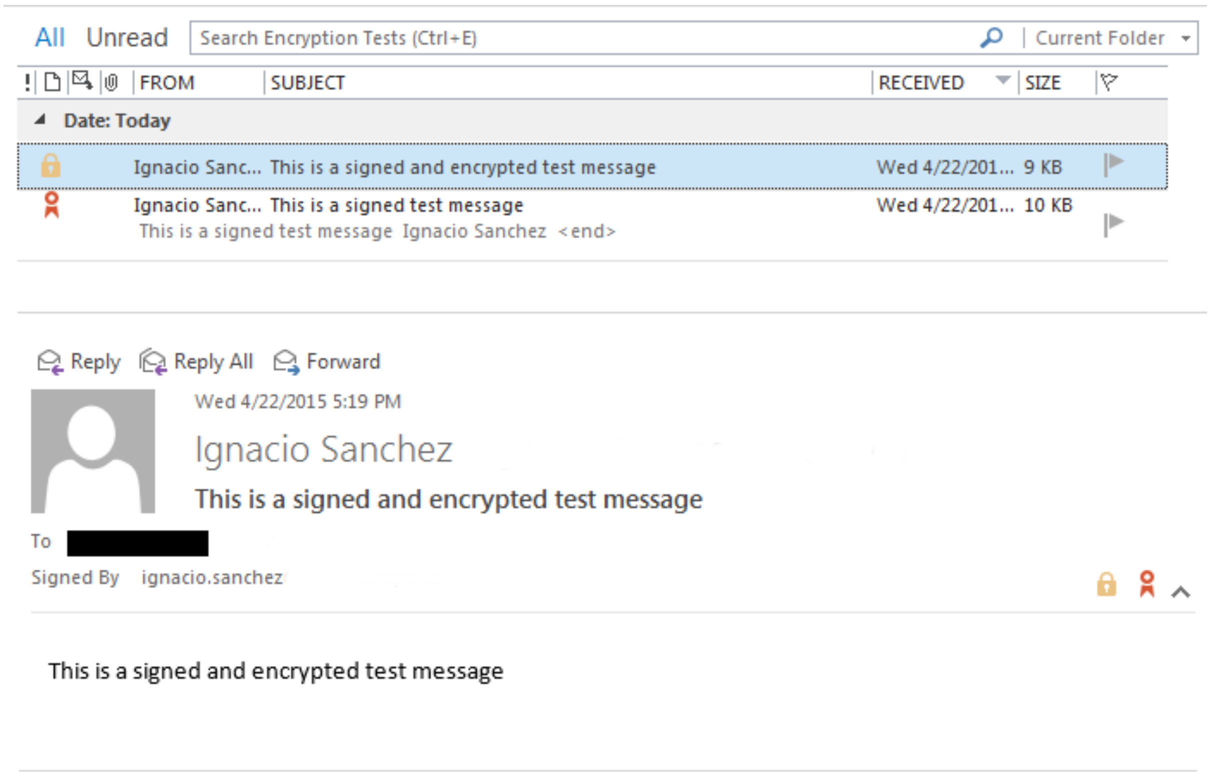
Figure 32: Encrypted and Signed S/MIME Parts in an Email

Typically, many of the cryptographic operations are transparent from the user's point of view. When a signed mail has been received by the user, the signature and the certificate are automatically verified by the mail client and subject to certificate verification, a symbol attesting that the email was properly signed is displayed next to the mail, as it can be seen in Figure 33 in the case of the Microsoft Outlook mail client.

The decryption process is also automatically performed by the client's software assuming that the respective cryptographic keys are properly installed. Usually, outbound email is not encrypted or signed by default. The user will have to specifically enable signature or encryption per email. However, most of the email clients allow easily changing these default settings so that every email is automatically signed and/or encrypted if the recipient's certificate is known or can be retrieved from a trusted source.

The S/MIME solution offers a good level of usability provided the required tools are properly installed and the user has access to a trusted directory where the public keys of his/her contacts are located. Conversely, the compromise of a single node of the PKI will result in the compromise of the entire system until this particular node is revoked.

Similarly to the HTTPS certificates, this solution for providing end-to-end email security requires that users obtain/buy a valid certificate from a provider whose root CA is trusted.


All Unread Search Encryption Tests (Ctrl+E) Current Folder

| FROM | SUBJECT | RECEIVED | SIZE |
|---|---|---|---|
| Date: Today | | | |
| Ignacio Sanc... | This is a signed and encrypted test message | Wed 4/22/201... | 9 KB |
| Ignacio Sanc... | This is a signed test message This is a signed test message Ignacio Sanchez <end> | Wed 4/22/201... | 10 KB |

Reply Reply All Forward

Wed 4/22/2015 5:19 PM

Ignacio Sanchez

This is a signed and encrypted test message

To [redacted]

Signed By ignacio.sanchez

This is a signed and encrypted test message

Figure 33: Examples of Signed and Encrypted Email Notifications with Microsoft Outlook

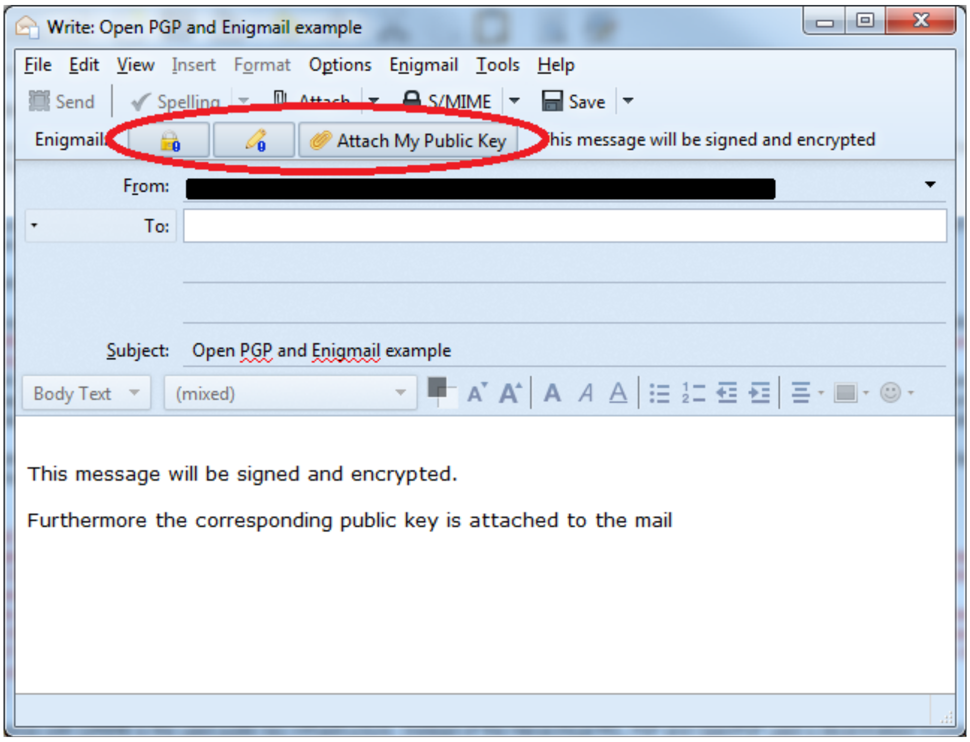
### 5.3.2 Pretty Good Privacy (PGP)

Pretty Good Privacy, also known as PGP, is a software tool developed in 1991 by Philip Zimmermann aimed at ensuring the confidentiality and integrity of email communications, also providing authentication for the sender.

In 1997, Network Associates Inc. bought the company founded by Zimmerman and the PGP software was extended to include other security features such as disk encryption and IPSec VPNs. Strongly inspired by the PGP tool, the Internet Engineering Task Force released in 1998 the standard RFC 2440 [10] called "OpenPGP Message Format" describing a common format for encrypted and/or signed messages as well as defining a list of agreed cryptographic algorithms. The standard RFC 3156 [20] was released in 2001 in order to define how to use the OpenPGP standard in compliance with the MIME standard. The standard was updated in 2007 in the RFC 4480 [77]. Since then, other standards or information notes have been released to extend the standard, as for example the standard RFC 6637 [44] that introduces Elliptic Curve Cryptography.

Whilst PGP is a commercial product sold by Symantec Corp., there are many free and open source solutions compliant with the OpenPGP format based on the GNU Privacy Guard (called GnuPG or GPG) software provided and maintained by the Free Software Foundation. There are many examples of email solutions which integrate with it, such as Thunderbird, through the Enigmail plug-in, or Mail.app of Mac OS X, through the GPGMail plug-in.

The main difference with S/MIME lies in the PKI that is used. Instead of a Hierarchical PKI, PGP and OpenPGP use a decentralised trusted model called *web of trust*. Users are responsible


Write: Open PGP and Enigmail example
File Edit View Insert Format Options Enigmail Tools Help
Send Spelling Attach S/MIME Save
Enigmail: Attach My Public Key This message will be signed and encrypted
Body Text (mixed)
From: [redacted]

To:

Subject: Open PGP and Enigmail example

This message will be signed and encrypted.

Furthermore the corresponding public key is attached to the mail

Figure 34: Signing and Encrypting Buttons in the Add-on Enigmail for Thunderbird

for the generation and dissemination of their certificates. The most secure way to hand over a certificate to another user is by delivering it physically, something that is not feasible when applied to online communications. In the web of trust concept, users publish their certificates on a key server publicly available such as the MIT server [55] or the pool of servers sks-keyservers [22]. Once a user has verified the legitimacy of the holder of a certificate using a secure channel (e.g. during a phone conversation), the user uses its signing key to attest that he/she trusts this person. By doing so, all the users already trusting the signing user will also trust the signed certificate.

The compromise of a single node in such a decentralised structure is definitely not critical. However, users should not focus on the number of persons trusting an identity, as forged identities can be self generated to attest that an identity is correct. It only works if a user can generate a chain of trust from its trusted nodes to the identity he/she is communicating with. Furthermore, the key management and the propagation of trust are usually in the hands of the users, a fact which makes PGP and OpenPGP cumbersome to use in practice. Nevertheless, the verification of a signature and the encryption processes are still transparent operations provided the certificates of the targeted user are already trusted. Encrypting an email and signing it is really easy with OpenPGP. As it can be seen in Figure 34 , Enigmail within Thunderbird adds specific buttons to encrypt or sign an email. There is even a button to attach the public key to the mail message.

The two solutions, namely S/MIME and PGP, can be considered as being user-friendly enough, provided the set-up has been done properly and the user has access to a trusted directory containing the public keys of his/her recipients. In the case of corporate environments, such a directory could easily be provided to the employees, as well as the keys and proper installation of the email tools required. Still there would be some drawbacks when accessing the emails from webmail interfaces or mobile devices.

In the case where email is used for personal communications outside corporate environments, obtaining and setting up the keys, the software environment and using it in practice for the day to day email communications can be very challenging.

Properly implemented end-to-end email security can be quite effective as countermeasure to mitigate privacy and security risks in email communications. The existing difficulties around the wide deployment of this type of solutions seem to be concentrated around the lack of some technical elements, such as easy acquisition of email certificates and public trusted key repositories, as well as usable email software which incorporates the required features by default. This type of solutions could benefit from a security and privacy by default approach in order to minimize the actions that shall be taken by the users in order to ensure the privacy of the email communications.

# 6 Conclusions

Email communications, born in the research community more than two decades ago, have evolved into the electronic communication protocol par excellence used on a daily basis by hundreds of millions of European citizens, as well as by most governments and businesses. The email ecosystem is a highly interoperable one and relies on a core set of protocols initially designed more than three decades ago, in an early digital context much different from the one found today in terms of digital privacy and security risks. Consequently, this core set was not originally designed with privacy and security requirements in mind, but under the assumption that the several actors involved in email communications could trust each other and that the digital communication links were secure.

With the massive adoption of Internet and email communications, a new rich set of complementary standards and tools were created in order to tackle the growing security and privacy concerns. However, these enhanced protocols and tools have failed in practice to deliver an effective protection. As a result, world-wide email communications remain largely vulnerable to security and privacy threats.

The main findings of this report are summarised as follows:

- **Email communications are in general not sufficiently protected.** The results of the evaluation suggest that the majority of world-wide email communications are subject to serious privacy and security risks. In most of the cases content transmitted by email can be intercepted by third parties putting at risk the confidentiality, integrity and availability of the information exchanged, such as the text of the message and the files attached to it.
- **There are standards, protocols and techniques capable of enhancing the security of email communications but they are not always used or implemented properly in practice**. Although there is no single countermeasure that has proven to be effective against all security and privacy risks, there are mature technological solutions that when combined and implemented properly can more effectively mitigate the email risks identified in this report.
- **Mature and interoperable end-to-end email security solutions exist but are rarely used in practice.** Mature end-to-end email security solutions, namely SMIME and OpenPGP (e.g. PGP/GPG), are already readily available but unfortunately rarely used in practice. The main barrier that has been identified for their adoption by European citizens is the lack of support by commercial providers that do not integrate them into their web-based email clients and mobile applications. One hypothesis for this lack of support and integration is linked to the fact that end-to-end security solutions would impact their current business models which currently involve the usage of the data transmitted and received by email. As a result of this lack of support and integration, currently the usage of end-to-end security solutions presents usability issues and requires certain IT skills that the average citizen does not possess.
- **Email communication channels (SMTP to SMTP) are not sufficiently protected in practice.** Security of email communication channels can be provided by employing SSL in the form of the STARTTLS protocol. However, we have observed that in practice the implementation of STARTTLS does not offer sufficient protection due to the following factors:

  - *Fall back to unencrypted communications.* When the usage of STARTTLS between two servers fails, the communication downgrades to an unencrypted communication in order to preserve the interoperability. Therefore, STARTTLS can only be currently seen in practice as a sort of opportunistic encryption, vulnerable to easy to perform "active downgrade" attacks.
  - *Lack of validation of server certificates.* Self-signed server certificates are accepted in practice in order to preserve interoperability, opening the door to trivial "Man-In-The-Middle" attacks.
- **Lack of security in DNS has a direct impact on the security of email communications.** The public DNS system plays a central role in email communications as it is used to glue the several email actors together. As a result of this dependence, DNS vulnerabilities can be exploited in order to attack email communications. Therefore, in order to create secure email communications it is required to secure the DNS communications as well. Existing deployment of DNSSEC should be carefully analysed to determine the difficulty of deploying such solutions and identify their overhead on the DNS traffic. In addition of providing reliable and secure resolution of MX, SPF and A records, DNS can also help with the management of the public keys employed in STARTTLS. To that end, the implementation of DNSSEC with DANE should be strongly considered. An alternative solution called DNSCurve using elliptic curve cryptography was recently introduced and could be considered as well.
- **Email identity spoofing is still a major risk in email communications.** Email identity spoofing can be easily performed despite the specific countermeasures deployed to fight SPAM, which indirectly help mitigate the threat (i.e. SPF records). Given the design of the email protocols, only end-to-end security (i.e. SMIME or PGP/GPG) can effectively mitigate this risk.

The following recommendations have been identified in order to address the above mentioned issues.

**Incentivise industry to support end-to-end solutions.** We recommend that email service providers, in particular the big industry players that provide webmail services, are incentivised to provide support for interoperable end-to-end email security solutions (i.e. SMIME or OpenPGP) and integrate them into their products and services.

It is our hypothesis that the usage of end-to-end security solutions could be currently perceived by the industry as an impact to the existing business models based on the compilation and analysis of the personal data exchanged by email (i.e. for marketing purposes). Due to this fact, industry players following these practices will rarely support such end-to-end security solutions in email. Interoperable end-to-end email security solutions such as SMIME and OpenPGP have proven to be efficient in the protection of the privacy and security of email communications and should be promoted. Currently, the main impediment to their effective deployment is concentrated in the following aspects:

- *Usability issues.* Major email providers (such as Gmail or Hotmail) don't offer support for SMIME or OpenPGP. Currently, the vast majority of citizens use webmail based systems or mobile apps developed by the email providers, which in most cases lack support for these technologies. Even though many email providers support the usage of standalone email clients, the set-up of this solution not only requires extra effort and specialised knowledge from the citizens that will use the service, but also presents serious usability issues compared to the convenience of the web based interface.

- *Key management.* In the case of SMIME, the process required to obtain an email certificate from a trusted provider is still too complex for users without specialised IT skills. The process is also cumbersome and usually only the most determined users would be willing to follow it. Even though there are some providers that offer such services free of charge, in many cases the user will have to pay for the service. In the case of OpenPGP, there is no global trusted key repository for the storage and sharing of keys and the system is based on a more distributed model which is more difficult to be used transparently.

**Promote the integration of end-to-end solutions into existing products and services.** We recommend that email service providers and developers of email client software (including webmail systems) are incentivised to provide integration with interoperable end-to-end solutions (i.e. SMIME and PGP/GPG) in a transparent and usable way.

End-to-end solutions could be promoted if support for SMIME and OpenPGP would be provided by major email providers in their web-based services and mobile applications. In addition, the implementation of such solutions should be as transparent as possible, while still maintaining interoperability and keeping the user in control of the process. The provision of SMIME certificates could be integrated as part of the procedure followed to create an account and the management of keys could be integrated in the contact list already provided in a transparent way by email providers. A mutual trust system between email providers could be envisaged in order to facilitate the transparent recovery of the public key for a given recipient who operates on another email provider.

**Promote the security of the email communication channels.** We recommend that the usage of STARTTLS for the protection of the SMTP communication channels is promoted and required by default following a security by default approach.

There is a big percentage of the global email traffic that does not use STARTTLS at all. This fact is related to the interoperability of the email system. A SMTP server will still be interoperable even if STARTTLS is not supported at all. Given that this feature is completely hidden to the users, there is no actual pressure for the service provider to enable it at all. The usage of STARTTLS could be promoted in the following ways:

- Raise citizens' awareness regarding the dangers of unencrypted email communications.
- Make public information about the usage of STARTTLS per provider (such as the Google transparency report).
- Set of minimum requirements for a system to become interoperable or be considered secure (see next point).

**Development of a minimum set of security requirements supported by an "Email Privacy Seal".** We recommend the creation of a minimum set of requirements for an email system to be secure and interoperable (including full STARTTLS support) and accordingly consider the creation of an "email privacy seal" to highlight those email providers complying with this security and privacy requirements. The usage of this "EU Email Privacy Seal" could help the user understand the level of commitment of this particular provider with the security and privacy of email communications and give a level of confidence in using their services.

Moreover, it will implicitly instigate email service providers to optimize their services in terms of security to maintain their competitiveness. In particular, the following requirements could be considered:

- Full SMIME support using certificates signed by a trusted CA
- DNSSEC support with DANE
- SPF records
- SMIME and OpenPGP support in proprietary web interfaces, desktop and mobile applications.

# Bibliography

[1] Donald E. Eastlake 3rd. Domain name system security extensions. RFC 2535, RFC Editor, March 1999. http://www.rfc-editor.org/rfc/rfc2535.txt.

[2] Michael Adkins. The Current State of SMTP STARTTLS Deployment. www.facebook.com/notes/protect-the-graph/the-current-state-of-smtp-starttls-deployment/1453015901605223, 2014.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Dns security introduction and requirements. RFC 4033, RFC Editor, March 2005. http://www.rfc-editor.org/rfc/rfc4033.txt.

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol modifications for the dns security extensions. RFC 4035, RFC Editor, March 2005. http://www.rfc-editor.org/rfc/rfc4035.txt.

[5] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource records for the dns security extensions. RFC 4034, RFC Editor, March 2005. http://www.rfc-editor.org/rfc/rfc4034.txt.

[6] Daniel Bernstein. DNSCurve: Usable security for DNS. dnscurve.org/, 2009.

[7] Enrico Blanzieri and Anton Bryl. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1):63–92, 2008.

[8] Bill Brener. Akamai Addresses CVE 2015-0204 Vulnerability. blogs.akamai.com/2015/03/cve-2015-0204-getting-out-of-the-export-business.html, 2015.

[9] M. Butler, J. Postel, D. Chase, J. Goldberger, and J. K. Reynolds. Post office protocol: Version 2. RFC 937, RFC Editor, February 1985. http://www.rfc-editor.org/rfc/rfc937.txt.

[10] Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer. Openpgp message format. RFC 2440, RFC Editor, November 1998. http://www.rfc-editor.org/rfc/rfc2440.txt.

[11] CERT. Nimda worm. 2001.

[12] CNN. Security firm: Mydoom worm fastest yet. 1999.

[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008. http://www.rfc-editor.org/rfc/rfc5280.txt.

[14] M. Crispin. Interactive mail access protocol: Version 2. RFC 1064, RFC Editor, July 1988. http://www.rfc-editor.org/rfc/rfc1064.txt.

[15] M. Crispin. Internet message access protocol - version 4rev1. RFC 3501, RFC Editor, March 2003. http://www.rfc-editor.org/rfc/rfc3501.txt.

[16] Mark R. Crispin. Interactive mail access protocol: Version 2. RFC 1176, RFC Editor, August 1990. `http://www.rfc-editor.org/rfc/rfc1176.txt`.

[17] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246, RFC Editor, August 2008. `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[18] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[19] D. Eastlake. Domain name system (dns) iana considerations. BCP 42, RFC Editor, April 2013. `http://www.rfc-editor.org/rfc/rfc6895.txt`.

[20] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. Mime security with openpgp. RFC 3156, RFC Editor, August 2001. `http://www.rfc-editor.org/rfc/rfc3156.txt`.

[21] R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Message syntax and routing. RFC 7230, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7230.txt`.

[22] Kristian Fiskerstrand. SKS Keyservers. `sks-keyservers.net/`.

[23] European Union Agency for Network and Information Security. Algorithms, key size and parameters report. `bitmessage.org/wiki/Main_Page`.

[24] Electronic Frontier Foundation. StartTLS Everywhere. A system for ensuring and authenticating STARTTLS encryption between mail servers. `https://github.com/EFForg/starttls-everywhere`, 2015.

[25] John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach, Ari Luotonen, and Lawrence C. Stewart. Http authentication: Basic and digest access authentication. RFC 2617, RFC Editor, June 1999. `http://www.rfc-editor.org/rfc/rfc2617.txt`.

[26] N. Freed and J. Klensin. Media type specifications and registration procedures. RFC 4288, RFC Editor, December 2005. `http://www.rfc-editor.org/rfc/rfc4288.txt`.

[27] N. Freed and J. Klensin. Multipurpose internet mail extensions (mime) part four: Registration procedures. BCP 13, RFC Editor, December 2005. `http://www.rfc-editor.org/rfc/rfc4289.txt`.

[28] Ned Freed and Nathaniel S. Borenstein. Multipurpose internet mail extensions (mime) part five: Conformance criteria and examples. RFC 2049, RFC Editor, November 1996. `http://www.rfc-editor.org/rfc/rfc2049.txt`.

[29] Ned Freed and Nathaniel S. Borenstein. Multipurpose internet mail extensions (mime) part one: Format of internet message bodies. RFC 2045, RFC Editor, November 1996. `http://www.rfc-editor.org/rfc/rfc2045.txt`.

[30] Ned Freed and Nathaniel S. Borenstein. Multipurpose internet mail extensions (mime) part two: Media types. RFC 2046, RFC Editor, November 1996. `http://www.rfc-editor.org/rfc/rfc2046.txt`.

[31] A. Freier, P. Karlton, and P. Kocher. The secure sockets layer (ssl) protocol version 3.0. RFC 6101, RFC Editor, August 2011. `http://www.rfc-editor.org/rfc/rfc6101.txt`.

[32] Jim Galvin, Sandy Murphy, Steve Crocker, and Ned Freed. Security multiparts for mime: Multipart/signed and multipart/encrypted. RFC 1847, RFC Editor, October 1995. `http://www.rfc-editor.org/rfc/rfc1847.txt`.

[33] R. Gellens and J. Klensin. Message submission for mail. STD 72, RFC Editor, November 2011. `http://www.rfc-editor.org/rfc/rfc6409.txt`.

[34] Randall Gellens, Chris Newman, and Laurence Lundblade. Pop3 extension mechanism. RFC 2449, RFC Editor, November 1998. `http://www.rfc-editor.org/rfc/rfc2449.txt`.

[35] Google. Google Transparency Report - Email Encryption in Transit. `www.google.com/transparencyreport/saferemail`, 2015.

[36] P. Hoffman. Smtp service extension for secure smtp over transport layer security. RFC 3207, RFC Editor, February 2002. `http://www.rfc-editor.org/rfc/rfc3207.txt`.

[37] P. Hoffman and J. Schlyter. The dns-based authentication of named entities (dane) transport layer security (tls) protocol: Tlsa. RFC 6698, RFC Editor, August 2012. `http://www.rfc-editor.org/rfc/rfc6698.txt`.

[38] Paul Hoffman. Enhanced security services for s/mime. RFC 2634, RFC Editor, June 1999. `http://www.rfc-editor.org/rfc/rfc2634.txt`.

[39] R. Housley. Cryptographic message syntax (cms). RFC 3369, RFC Editor, August 2002. `http://www.rfc-editor.org/rfc/rfc3369.txt`.

[40] R. Housley. Cryptographic message syntax (cms) algorithms. RFC 3370, RFC Editor, August 2002. `http://www.rfc-editor.org/rfc/rfc3370.txt`.

[41] Russell Housley, Warwick Ford, Tim Polk, and David Solo. Internet x.509 public key infrastructure certificate and crl profile. RFC 2459, RFC Editor, January 1999. `http://www.rfc-editor.org/rfc/rfc2459.txt`.

[42] Mikko Hypponen. How we found the file that was used to hack rsa. 2011.

[43] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, October 2007.

[44] A. Jivsov. Elliptic curve cryptography (ecc) in openpgp. RFC 6637, RFC Editor, June 2012. `http://www.rfc-editor.org/rfc/rfc6637.txt`.

[45] Burt Kaliski. Pkcs #7: Cryptographic message syntax version 1.5. RFC 2315, RFC Editor, March 1998. `http://www.rfc-editor.org/rfc/rfc2315.txt`.

[46] S. Kitterman. Sender policy framework (spf) for authorizing use of domains in email, version 1. RFC 7208, RFC Editor, April 2014. `http://www.rfc-editor.org/rfc/rfc7208.txt`.

[47] J. Klensin. Simple mail transfer protocol. RFC 2821, RFC Editor, April 2001. `http://www.rfc-editor.org/rfc/rfc2821.txt`.

[48] J. Klensin. Simple mail transfer protocol. RFC 5321, RFC Editor, October 2008. `http://www.rfc-editor.org/rfc/rfc5321.txt`.

[49] John Klensin, Ned Freed, and Keith Moore. Smtp service extension for message size declaration. RFC 1653, RFC Editor, July 1994. http://www.rfc-editor.org/rfc/rfc1653.txt.

[50] John Klensin, Ned Freed, Marshall T. Rose, Einar A. Stefferud, and Dave Crocker. Smtp service extensions. STD 10, RFC Editor, November 1995. http://www.rfc-editor.org/rfc/rfc1869.txt.

[51] John C. Klensin, Randy Catoe, and Paul Krumviede. Imap/pop authorize extension for simple challenge/response. RFC 2195, RFC Editor, September 1997. http://www.rfc-editor.org/rfc/rfc2195.txt.

[52] Changwei Liu and Sid Stamm. Fighting unicode-obfuscated spam. In *Proceedings of the Anti-phishing Working Groups 2Nd Annual eCrime Researchers Summit*, eCrime '07, pages 45–59, New York, NY, USA, 2007. ACM.

[53] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, and R. L. Rivest. Handbook of applied cryptography, 1997.

[54] Malware Messaging and Mobile Anti-Abuse Working Group. Report 16 1st quarter 2012 through 2nd quarter 2014. 2014.

[55] MIT. MIT PGP Keyservers. pgp.mit.edu/, 2015.

[56] P. Mockapetris. Domain names: Concepts and facilities. RFC 882, RFC Editor, November 1983. http://www.rfc-editor.org/rfc/rfc882.txt.

[57] P. Mockapetris. Domain names - implementation and specification. STD 13, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1035.txt.

[58] K. Moore and G. Vaudreuil. An extensible message format for delivery status notifications. RFC 3464, RFC Editor, January 2003. http://www.rfc-editor.org/rfc/rfc3464.txt.

[59] Keith Moore. Mime (multipurpose internet mail extensions) part three: Message header extensions for non-ascii text. RFC 2047, RFC Editor, November 1996. http://www.rfc-editor.org/rfc/rfc2047.txt.

[60] John G. Myers. Pop3 authentication command. RFC 1734, RFC Editor, December 1994. http://www.rfc-editor.org/rfc/rfc1734.txt.

[61] John G. Myers. Simple authentication and security layer (sasl). RFC 2222, RFC Editor, October 1997. http://www.rfc-editor.org/rfc/rfc2222.txt.

[62] John G. Myers and Marshall T. Rose. Post office protocol - version 3. STD 53, RFC Editor, May 1996. http://www.rfc-editor.org/rfc/rfc1939.txt.

[63] John Gardiner Myers. Smtp service extension for authentication. RFC 2554, RFC Editor, March 1999. http://www.rfc-editor.org/rfc/rfc2554.txt.

[64] Chris Newman. Using tls with imap, pop3 and acap. RFC 2595, RFC Editor, June 1999. http://www.rfc-editor.org/rfc/rfc2595.txt.

[65] Ola Nordström and Constantinos Dovrolis. Beware of bgp attacks. *ACM SIGCOMM Computer Communication Review*, 34(2):1–8, 2004.

[66] European Parliament. Libe committee inquiry. electronic mass surveilance of eu citizens. 2014.

[67] PP Paul, P Judge, D Alperovitch, and W Yang. Understanding and reversing the profit model of spam. In *Proceedings of the Workshop on the Economics of Information Security (WEIS)*, pages 1–11, 2005.

[68] Jonathan B. Postel. Simple mail transfer protocol. STD 10, RFC Editor, August 1982. `http://www.rfc-editor.org/rfc/rfc821.txt`.

[69] B. Ramsdell. Secure/multipurpose internet mail extensions (s/mime) version 3.1 certificate handling. RFC 3850, RFC Editor, July 2004. `http://www.rfc-editor.org/rfc/rfc3850.txt`.

[70] B. Ramsdell. Secure/multipurpose internet mail extensions (s/mime) version 3.1 message specification. RFC 3851, RFC Editor, July 2004. `http://www.rfc-editor.org/rfc/rfc3851.txt`.

[71] B. Ramsdell and S. Turner. Secure/multipurpose internet mail extensions (s/mime) version 3.2 message specification. RFC 5751, RFC Editor, January 2010. `http://www.rfc-editor.org/rfc/rfc5751.txt`.

[72] Peter W. Resnick. Internet message format. RFC 5322, RFC Editor, October 2008. `http://www.rfc-editor.org/rfc/rfc5322.txt`.

[73] J. K. Reynolds. Post office protocol. RFC 918, RFC Editor, October 1984. `http://www.rfc-editor.org/rfc/rfc918.txt`.

[74] Joyce K. Reynolds. Helminthiasis of the internet. RFC 1135, RFC Editor, December 1989. `http://www.rfc-editor.org/rfc/rfc1135.txt`.

[75] Stefan A. Robila and James W. Ragucci. Don't be a phish: Steps in user education. *SIGCSE Bull.*, 38(3):237–241, June 2006.

[76] Marshall Rose. Post office protocol: Version 3. RFC 1081, RFC Editor, November 1988. `http://www.rfc-editor.org/rfc/rfc1081.txt`.

[77] H. Schulzrinne, V. Gurbani, P. Kyzivat, and J. Rosenberg. Rpid: Rich presence extensions to the presence information data format (pidf). RFC 4480, RFC Editor, July 2006. `http://www.rfc-editor.org/rfc/rfc4480.txt`.

[78] Symantec. Happy99.worm technical analysis. 1999.

[79] Symantec. W97m.melissa.a advisory. 1999.

[80] Symantec. Vbs.loveletter.var. 2000.

[81] S. Turner and T. Polk. Prohibiting secure sockets layer (ssl) version 2.0. RFC 6176, RFC Editor, March 2011. `http://www.rfc-editor.org/rfc/rfc6176.txt`.

[82] Jonathan Warren. Bitmessage: A PeertoPeer Message Authentication and Delivery System. `bitmessage.org/bitmessage.pdf`, 2012.

[83] Jonathan Warren. BitMessage Wiki Page, 2014.

JRC 99372

EUR 28509 EN

ISSN 1831-9424

ISBN 978-92-79-66503-5

doi:10.2760/319735

European Commission
Joint Research Centre – Institute for the Protection and Security of the Citizen

Title: A security analysis of email communications

Author(s): Ignacio Sanchez, Apostolos Malatras, Iwen Coisel

Luxembourg: Publications Office of the European Union

2015 – 70 pp. – 21.0 x 29.7 cm

KJ-NA-28509-EN-N

## JRC Mission

Publications Office

doi:10.2760/319735

ISBN 978-92-79-66503-5