Czech Technical University – Faculty of Electrical Engineering

X36SWT – Software Technology

Project Report

Spam and how to fight it

Author: Bc. Marek Handl

Date: May 2009

Table of Contents

Email

Bits from history

History of electronic email has started years before birth of Internet as we know it today. The invention and the first construction of computers can be dated to the times of the Second World War. This is the beginning of the mainframe era, which lasted for several decades. In the 1960’s scientists and computer enthusiasts sitting at terminals connected to a mainframe were able to send messages to one another. These are usually considered the first electronic messages send via computers. However, people were only able to communicate to users using the same mainframe.

During the 1960’s the United States Department of Defense started to work on a project which evolved into the ARPANET (Advanced Research Projects Agency Network). The evolution was slow and complicated. The first connection between two nodes took place in 1969. There is no date when the Internet started to exist. The fact is that the work Internet was first used in 1974. Although, it took several other years before it became a well known term.

It was a natural requirement to be able to send messages among individual machines. It is usually acknowledged that the first messages similar to nowadays email was sent in 1971. At those time there was no standard describing message formats, but since there was only a very limited number of users working with the computers they always found a consensus. It is worth noting, that they used the “at sign” (@) to delimit username and machine name, which is what it is used for even today.

Common computers as we know them today were developed in the second half of 1970’s. The first personal computer was the Apple II developed by Apple Computers in 1977.

The email communication was standardized by the RFC822 in 1982. The RFC’s (Request for Comments) are documents used for describing how things work or should work on the Internet. There is an organization called the IETF (Internet Engineering Task Force) that controls the RFC’s and when there’s time they are released as Internet Standards. The evolution of the email started with the mentioned RFC822, but it lasts until today. The most recent addition is RFC5322 from year 2008.

Protocols

There are number of protocols that are used for email transport. The most common are SMTP, POP and IMAP. Large software companies have developed their own protocols that they use on their email servers. An example is MAPI protocol used by the Microsoft Exchange Server. These non-standard protocols often offer some extended functionality, but are mostly proprietary and closed. They will not be covered in this paper.

Protocol	Version	Latest RFC	TCP port	Functionality
POP	3	RFC 1939	110	Read emails
IMAP	4rev1	RFC 3501	143	Read and manage emails on the server
SMTP	---	RFC 5321	25	Send emails

Table 1: List of major email protocols

POP, the Post Office Protocol, is used when user wants to read emails that arrived into their mailbox. It allows fetching an email from the server and storing it locally in user's desktop application.

IMAP, the Internet Message Access Protocol, offers similar functionality. It is also usually used to read messages. However, one can control their mailbox via IMAP.

When a messages is fetched using POP, a new copy of the message is created locally and from that moment this copy exists independent of the server. The server can keep the original message or can delete it based on the setup. On the other hand IMAP requires a constant connection to the server. When user operates on the messages through the desktop application, he/she is actually working with the message on the server. When they move, modify or delete the message, the changes are propagated to the server.

It is obvious that POP was the original protocol to use in the old days. During that time only a very limited connection existed between computers. Most of personal computers were connected via a dial up link to the Internet, which was charged for based on the connection time. But as the time flew more and more computers got a permanent access to the Internet. With a time unrestricted connection it is possible to use the IMAP, which brings more functionality to the user. Nowadays both protocols are used. Practically all email providers support POP and most of them also support the IMAP. Current versions of the protocols are POP3 and IMAP4 revision 1.

The SMTP, Simple Mail Transfer Protocol, is contrarily used for sending emails and will be covered in the Smtp chapter.

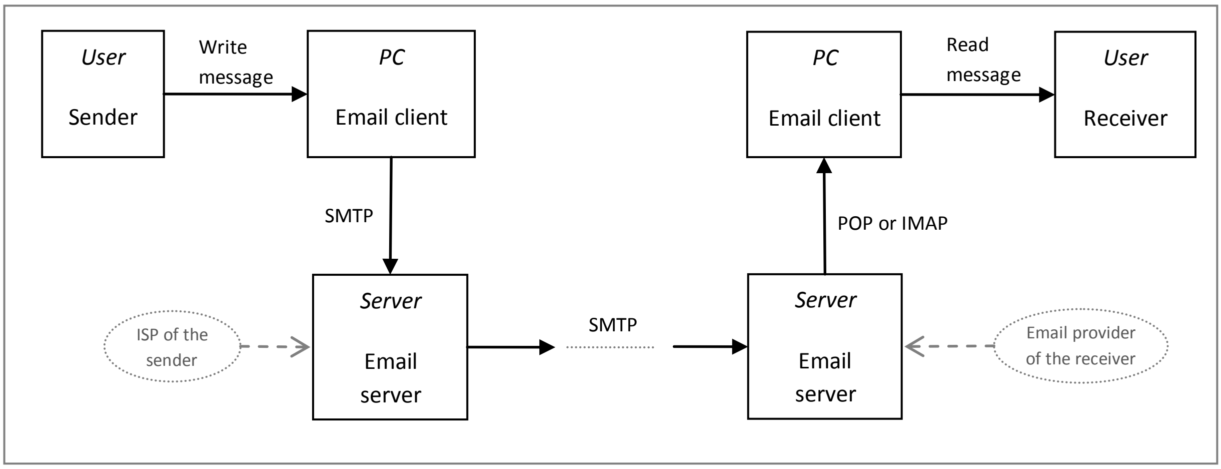


Figure 1: Overview of a typical email communication

There are two kinds of software needed for email transport. The first one is the email client, i.e. the application that the end user communicates with. This is usually represented by a desktop application, e.g. Microsoft Outlook, or a web application, often called web mail. This software allows for writing and sending new emails as well as for reading and managing received emails. Email client is sometimes called the MUA – Mail User Agent.

The second kind of software needed for email communication is the email server. A more precise term is the MTA – Mail Transfer Agent. This application is usually not visible to the users. They do not need to know about its existence. All they have to do is to setup their clients to communicate with some server at some port. The usual setup is to use the email provider server for receiving emails (via POP or IMAP) and internet service provider (ISP) server for sending emails (via SMTP). The email server is where the user email account actually exists. It makes sure that all emails arrive to the receiver mailbox on their dedicated server.

Software	Examples
MUA - Email client	MS Outlook (Express) Mozilla Thunderbird Mutt IBM Lotus Notes The Bat!
MTA - Email server	MS Exchange Server Sendmail Postfix Exim

Table 2: Examples of commonly used email communication software

The SMTP is a push protocol. It means that when sending email the sender is the one who pushes the email further – from the sender client to the first server and then between other servers. On the other hand, POP and IMAP are pull protocols. I.e. the client has to ask for the email.

Email format

As mentioned above, there are Internet Standards defining email communication. Unfortunately most of the software does not fully adhere to the standards. This is one of the reasons why there are so many opportunities for spammers to misuse existing infrastructure, which will be discussed in the Spam chapter of this paper. Most of email software is built so that it follows the standard from the most part, but is also able to deal with slightly non-standard structures.

Format of an email address is defined in RFC2821 and RFC2822. The fundamental scheme is

```
local_part @ domain_part
```

Where the local part can consist from a dot-string or a quoted string. A dot-string consists of strings separated by a period character. The strings can contain various characters, see Table 3, but mostly just alphanumerical characters, hyphens and underscores are used. It is also possible to put comments into a dot-string, but that is rarely used. A comment is a string enclosed in a pair of round brackets. The second option is a quoted string, i.e. a string enclosed in a pair of quotes; it can contain all US-ASCII characters excluding the backslash and the quote character. However, usage of quoted strings is discouraged. The maximum length of the local part is 64 characters.

The domain part defines an Internet domain name, an address of the email server. Its prescribed format is a dot-string again. The maximum length of the domain part is 255 characters. Practically, the length of the whole address cannot be more than 256 characters, because that is the limit in MAIL and RCPT SMTP commands.

Alphanumeric ! # $ % & ‘ * + - / = ? ^ _ ` { | } ~

Table 3: List of symbols allowed in a dot-string. More or less, it represents symbols that may be used in an email address.

Emails get transported over a TCP/IP connection in plain text format. That was the original idea. Nowadays the messages are often encoded or encrypted, but their encoded/encrypted version is still a composition of printable characters. Early RFC's required lines to be maximally 78 characters long. Today the boundary is 998 characters per line. Since there is no guarantee that the message will only go through modern email servers, it is a good practice to reformat the message so that it adheres to the old standards. This is usually done by the email clients people use to send emails. Each line in a message ends with two characters – CR LF (decimal ASCII codes 13 and 10). Including terminating characters a line can be 1000 characters long.

Each email message consists of two parts – a header and a body. This is then wrapped in an SMTP envelope. The envelope is often omitted when talking about messages and only the header and the body is considered an email. The reason for this is that the SMTP envelope is not visible to the end user, although it plays a very important role in the transport process. Email header and body together are often called message content or mail data.

The envelope defines the sender and the receiver of the message. The envelope and the header are quite independent of one another. This is actually another reason why there are so many spam emails around, which will be shown later in the paper.

The header is divided into fields, each composed from a name, a colon, and a value. There are numerous fields that can be present in an email header. Some of them are mandatory, some are optional and some are non-standard headers. Typically a field fits on a single line; in that case each line is considered a new field. When there is need for a longer value, the syntax is that the first line is printed as usual and second and other lines start with a white character – a space or a tabulator. The header ends with an empty line.

The body contains the real contents of the message. For simple messages it is literally what is displayed to the user. For more complicated messages with HTML, attachments and other extensions the structure is slightly more complex.

Text encoding

The original email standards were created in times when the Internet network was very limited. At that time nobody anticipated the future evolution which took place decades after. Almost all computers were located in the United States; therefore everything was created so that it supports English language. They did not care for internalization, which was a major mistake from today's point of view. Computers and networks were expensive and slow then. It was logical to make the communication as simple as possible, though. Any support for non-english characters would make the communication costs larger, which was not acceptable.

Therefore, the email was defined as using only 7-bit ASCII (also known as US-ASCII) characters. Seven bits cover needs for English communication, but are not enough for sending binary data and for communication in languages with other non-english characters, e.g. French, Spanish, German and Czech. To solve these problems email extensions have been devised. They are defined in numerous RFC's again. There is a well supported mechanism that deals with the mentioned problems which is called the MIME, Multipurpose Internet Mail Extension.

One way to encode characters out of 7 bit scope is called the Quoted-Printable scheme. The idea is to use the hexadecimal presentation of the character instead of the character itself. Typically, a character has 8 bits or can be divided into 8 bit groups. Each 8 bit group can be described by a hexadecimal number that has 2 digits. For example, let's have a character with hexadecimal representation of A3 (i.e. 163 decimally). To represent this character in an email, three characters are written out – an equal sign, an A and a 3, i.e. =A3. This way it is possible to encode any character that requires more than 7 bits. The advantage of this approach is that only non-english characters need to be encoded; the rest of the text stays intact. Therefore; Quoted-Printable is usually used when there are only a couple of characters that need encoding.

Original	ASCII decimal	ASCII hexadecimal	Quoted-printable encoded
P	80	50	P
ř	248	F8	=F8
e	101	65	E
d	100	64	D
m	109	6D	M
ě	236	EC	=EC
t	116	74	T
Předmět	---	---	P=F8edm=ECt

Table 4: An example of Quoted-printable encoding (using Windows-1250 character set)

Another mechanism for character encoding is the Base64. This approach is used to encode the whole stream of data instead of single characters. Therefore, it is used for encoding binary data or texts with high frequency of special characters. Core of the algorithm is a 64 character alphabet. The data are taken as a binary stream. Each 24 bits (3 characters of 8 bits) get encoded into 4 seven-bit characters as follows. Take the first 6 bits and use it as an index into the 64 item array, the result is the final character. Take the other 6 bits and do the same thing etc. If the stream is not divisible into 24 bit chunks padding is added at the end (using an equal sign).

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

Table 5: Base64 alphabet consisting of 64 US-ASCII characters

Obviously, using Quoted-printable or Base64 makes the data to be transferred longer than the original. In the case of the Quoted-printable approach the overhead equals to 200%, because one 8-bit character gets encoded as 3 characters. There usually are a small number of encoded characters in the text; therefore the overall overhead of the message is very small. The Base64 encoding introduces an average overhead of 37% of the message length. This is the final result, because Base64 encodes the whole text.

Original	Binary representation	Indices (decimal)	Base64 encoded
Pře	010100 001111 100001 100101	20, 15, 33, 37	UPhl
dmě	011001 000110 110111 101100	25, 6, 55, 44	ZG3s
t	011101 00	29, 0	dA
Předmět	---	---	UPhlZG3sdA==

Table 6: An example of Base64 encoding (using Windows-1250 character set)

There are other techniques how to deal with the 7-bit problem. Modern email software often supports new standards that allow for 8-bit transfers on SMTP level. These include the 8BITMIME and BINARYMIME extensions. If both communicating nodes (that have a direct connection with one another) support the extensions, they can use it and there is no need for complicated data encoding. However the previously mentioned solutions, the Quoted-printable and the Base64, have the advantage that only the end nodes have to understand the encoding. The intermediate nodes do not have to know any MIME and still will be able to transport the message without any loss or error, because result of the encoding is a US-ASCII plain text. Nowadays the Quoted-printable and the Base64 are a de facto standard. Eight bit transfer extensions are spreading but are not as common yet.

Email header

It has been mentioned earlier that the header consists of fields that follow this pattern:

Field_name: field_value CR LF

Values that span over multiple lines start new line with a white character:

Field_name: field_value_start CR LF
    field_value_continue CR LF
    field_value_end CRLF

The header uses standard 7-bit ASCII. To insert non-english characters the so called MIME encoded words are used. The syntax is as follows:

=?charset?encoding?encoded_text?=

Where encoding can be either *Q* for Quoted-printable or *B* for Base64.

There is no real limit as to the number of fields that can be present in the header, but the length of the mail data (header plus body) is usually limited. The early standards set the limit to 65 536 bytes, but today it is up to the server to set the limit. New headers are often added as the email passes through servers on its way to the receiver. The end of the header is designated by an empty line.

Encoding	**MIME encoded word for “Předmět“**
Quoted-printable	=?windows-1250?Q?P=F8edm=ECt?=
Base64	=?windows-1250?B?UPhlZG3sdA==?=

Table 7: Examples of MIME encoded words

The header is flexible. The email standards define basic set of fields, but it is allowed to create new fields if necessary. All the new fields have to start with an X-. Email software that understands the header will use it and all other clients or servers will simply copy it to the output ignoring its meaning. This way it is possible to add other features to email communication. The most common use is to store information about virus and spam scanning into the X header fields. Examples are X-Virus-Scanner, X-VirusDetected, X-SpamScore and similar.

A list of most common header fields follows.

- **From**
  - Value of the field should define the sender of the message. It contains sender's email address and can also contain sender's name. Unfortunately, it is up to the client software to populate the field. The sender name that is transmitted through the email envelope and this *From* field are fully independent. In consequence, the *From* value does not have to any connection to the real sender. This is often misused in the spam emails.
  - Example: ***`From: Marek Handl <handlml@fel.cvut.cz>`***
- **To**
  - Defines receiver of the message. It follows the same format as the From field. It can optionally hold the receiver's full name as well. It gets populated by the sending user.
  - Example: ***`To: Marek Handl <handlml@fel.cvut.cz>`***
- **CC**
  - CC stands for carbon copy and defines other receivers of the message. It uses the same format as the *To* field.
- **BCC**
  - BCC stands for blind carbon copy. It again uses the same format as the To field. The usual implementation is that recipients that were specified in the BCC field are not mentioned in the email header when the message is delivered. However, the email standards are not clear whether or not it is a must that they will not be displayed.
- **Subject**
  - A short one line title of the message. It is up to the sending user to fill in this field. It is good practice not to leave it blank.
- **Date**
  - Date when the message was sent out. The sending client populates it with current date and time. There is a defined format to be used, that contains the time zone as well.
  - Example: ***`Date: Fri, 22 May 2009 19:38:30 +0200`***
- **Content-Type**
  - Value consists of two parts divided by a slash: type/subtype. The default value is text/plain, which denotes a message in the old plain text format. Modern usage of electronic mail required support for formatted texts and non-textual data. The MIME scheme introduces other values for the content-type field. Here are the most common ones.
    - text/html – message content is an html source code (i.e. text with formatting)
    - image/jpeg, image/gif …
    - audio/mpeg, audio/x-wav …
    - video/mpeg
    - application/pdf, application/zip, application/msword
    - application/octet-stream
    - multipart – this option is discussed below

    - This field also contains information about character set that was used to build the message.
    - Example: `Content-Type: text/html; charset=windows-1252`
- **Content-Transfer-Encoding**
    - It defines encoding that was used to transform 8-bit data into 7-bit representation. Default value is 7bit. Other most common values are: base64, quoted-printable, 8bit.
    - Example: `Content-Transfer-Encoding: quoted-printable`
- **Received**
    - Usually more than one *Received* field can be found in an email. Each email server that receives the message on its way from the sender to the receiver should prepend one *Received* field informing about where the message came from. By putting all the fields together it should be possible to trace the whole route of the message.
    - Example:
      ```
      Received: from mx1.phx.paypal.com ([66.211.168.231])
                by mx2.centrum.cz (Centrum Mailer) with ESMTP
                ;Thu, 21 May 2009 23:45:54 +0200
      ```
- **MIME-Version: 1.0**
    - This field should inform about the version of MIME that was used to construct the message. The reality is different, though. It has been discovered that changing the value from 1.0 causes problems with some servers and does not introduce any tangible improvement. Therefore, it was decided to keep it at 1.0 for ever. Its meaning is simply that the client or server used MIME extension.
- **Message-Id, In-Reply-To, References**
    - These fields are used to give messages identification numbers and keep a history of emails that are related. If used properly it is then possible to reconstruct the flow of communication between users.
- **Reply-To**
    - Can be used when the sender wants to identify themselves as one email address, but wants to receive eventual reply to another email address.
    - Example: `Reply-To: <handlml@fel.cvut.cz>`
- **Return-Path**
    - It contains the email address of the original sender and the route the message went through. It is created and put at the beginning of the email header by the final receiving email server by copying information from the SMTP MAIL command. It defines the address where failure messages (e.g. receiver mailbox full, receiver does not exist) are to be sent.
- **DKIM-Signature, DomainKey-Signature**
    - These fields are newly added to the standard. They are used for public key signing of the email. More on this can be found at the very end of the Spam chapter.

Email attachments

The requirement to send messages with attachments introduces complexity into structure of the email. The MIME extension defines many ways how to embed other documents into an email. The basic idea is to compose the message from different parts. Each part has its *Content-Type* and usually a *Content-Transfer-Encoding* as well. This way it is possible to put contents of various types into one message.

Let’s demonstrate it on a simple example. The goal is to create a message that will contain some text and a PDF document as an attachment. First thing is to place a *Content-Type* field in the header and set its value to *multipart/mixed*, which denotes, that the message is built from more than one part and the parts use different types. Still in the *Content-Type* field a boundary – parts delimiter – must be defined. A generated pseudorandom string is often used. Each time the email software hits a line starting with two dashes followed by the boundary it knows a new message part begins. The boundary is also repeated at the final end of the message, it not only starts with two dashes, but two more dashes are also appended to the boundary in this case. At the beginning of each part there is a header again, which is followed by an empty line, which is followed by the contents of the part.

```
From: Marek Handl <handlm1@fel.cvut.cz>
To: Marek Receiver <receiver@fel.cvut.cz>
Subject: Message with a PDF attachment
Content-Type: multipart/mixed;
       boundary="----=_NextPart_000_01F8_01C9C84E.9D94F8C0"
                                                               (an empty line)
------=_NextPart_000_01F8_01C9C84E.9D94F8C0
Content-Type: text/plain; charset=windows-1250
Content-Transfer-Encoding: quoted-printable
                                                               (an empty line)
This is the text of the message. The attachment starts right below.
                                                               (an empty line)
------=_NextPart_000_01F8_01C9C84E.9D94F8C0
Content-Type: application/octet-stream;
       name=filename.pdf
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
       filename=filename.pdf
                                                               (an empty line)
JVBERi0xLjQKJeLjz9MKMiAwIG9iaiA8PC9UeXBlL1hPYmplY3QvQ29sb3JTcGFjZVsvSW5kZXhl
ZC9EZXZpY2VSR0IgMjU1KGTKq/ETzU6JNiJNYB0urLL/i6GljGtcbr5IIHidQt+PMmM3aiT5SpEx
0Te8nS5RzhN09p5SJtb9Llxy8jqsEnpTVZMChX5LXyHSV5eSgrXUlY/QJn3sitHFXCn7axYTcVi0
ZBY4Yqs4aW7hXCjW7lwoaJcWlaJsqdpBgCZcXLiDXW7lj1pcYoR0/4KPRPasnHFMHpzq1LP8V1xi
                                                               (an empty line)

------=_NextPart_000_01F8_01C9C84E.9D94F8C0–
                                                               (an empty line)
                                                              (end of message)
```

Figure 2: A skeleton of an email message with a PDF attachment showing usage of multipart content type

Another header field is needed to define whether the multipart part is to be handled as an inline document or an attachment. Its name is *Content-Disposition* and the way it is used is demonstrated in Figure 2.

There are other possible values for the multipart content type. A common one is *multipart/alternative*, which designates that the message contains two parts that have the same meaning, but are in different format. Typically, there is a message in HTML format and in plain text. Another option is *multipart/signed*, which is used for signing the contents of the message with a private key.

It is possible to build a tree hierarchy from the multipart parts, because they can be nested into each other.

Smtp

Overview

The Simple Mail Transfer Protocol is the major protocol used for sending electronic messages. It has been standardized in RFC 821 in 1982 for the first time. During its existence it has been improved and enhanced. The most up to date version is described in RFC 5321, which has been published in 2008.

SMTP is a push protocol, which transports the message from the MUA (email client) of the sender to the first MTA (email server) and then between consecutive MTA's until it reaches the final one, where the receiver's mailbox is located. See Figure 1 for graphical explanation. In the scenario when the sender and the receiver share the same email provider, the message has to travel to only one MTA. The most common situation is that the message visits two MTA's. The first one is the MTA of the sender's ISP and the second one is the MTA of the receiver's email provider. Once it is not possible to create a direct connection between these two MTA's, an intermediary MTA comes in play, which is then called an SMTP relay.

Each MTA that accepts a message becomes responsible for it. It means that each MTA receives the whole message and stores is locally or sends it further. It is its responsibility to react in case the message is not accepted at the next node. In case of such a failure, the MTA has to either send the message through a different route or wait and resend the message later or send a notice to the original sender that the message is undeliverable.

Routing the message is based on DNS (domain name system) records as other Internet services are. Necessary communication is displayed on Figure 3. The diagram is slightly misleading, because it displays *To* and *From* fields, which are also names used in email header and which eventually are the values visible to the end user. However, these values do not have any effect on where the message is sent. The SMTP communication is based on what is defined in the SMTP envelope.

1. Alice's email client is set up so that it knows that outgoing emails are to be sent to *smtp.a.org*, which usually represents email server of Alice's ISP.
2. When the server receives an email message from Alice it takes the receiver from the SMTP envelope and strips out the domain name, i.e. *b.org*. It then asks its dedicated DNS server for an MX (Mail eXchange) DNS record of *b.org*.
3. The DNS server responds with the MX record. If there is no MX record for *b.org*, the A record is used instead. The record represents URL address of Bob's email provider.
4. The message is sent to *mx.b.org* via SMTP, where it waits for Bob to pick it up.
5. Bob reads the message by fetching it from the server. This is no longer done through SMTP, other protocols are used, e.g. POP or IMAP.

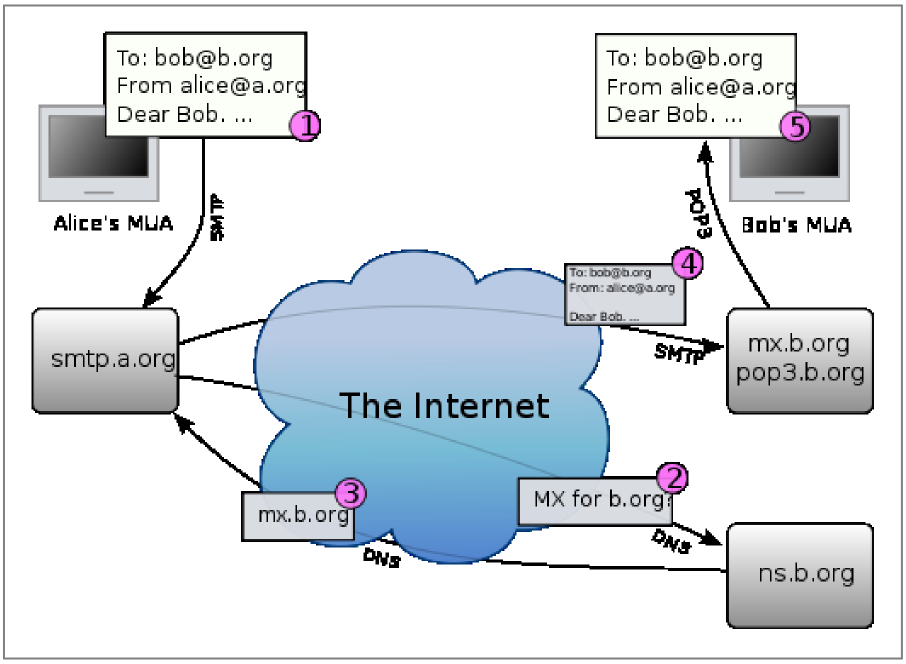


Figure 3: Electronic communication needed for email transport. Note that the To and From fields are values from the SMTP envelope. (Diagram taken from Wikipedia)

Commands

An SMTP command consists of a command code. Some commands have parameters in which case there is at least one space character between the code and the parameters. Each command ends with the character sequence CR LF (decimal ASCII codes are 13 and 10). The maximum length of a command line is 510 characters plus the terminating sequence. Text lines, i.e. lines transporting mail data (see DATA command below), can be longer – up to 998 characters plus terminating sequence.

A command code consists of exactly four alphabetic characters. The command codes are not case sensitive, but the values often are. The consensus is to write command codes in upper case.

The receiver end acknowledges commands from the sender by sending an SMTP reply. Replies are described by a 3 digit decimal number and an optional text message. The maximum length of a reply line is the same as of the command line, 510 characters plus terminating sequence. Replies in format 2xy represent a positive completion message, 3xy an intermediate reply, 4xy a temporary error condition and 5xy a permanent error.

250 Requested mail action okay, completed
354 Start mail input; end with <CRLF>.<CRLF>
500 Syntax error, command unrecognized
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented
550 Requested action not taken: mailbox unavailable

Table 8: List of most commonly used SMTP replies, i.e. reactions to SMTP commands.

A list of commonly used SMTP commands follows.

- **HELO mycomputer@myisp.cz**
    - It is used by the SMTP-sender to identify itself to the receiver. It should contain the host name of the sender, but most of email servers will accept any value here.
    - The HELO command must be the first command in communication session.
    - The receiver responds with an OK (250 SMTP reply) and the connection is established.
- **MAIL FROM:<handlm1@fel.cvut.cz>**
    - This is where the sender defines their email address. The from value can actually contain more than one address. In that case the last address is the sender's mailbox and the rest are hosts that the message relayed through so far. This routing information is used for error reporting messages (e.g. when message is undeliverable). The hosts are ordered so that the most recent relay is first. Maximum length of the value is 256 characters.
    - The receiver responds with an OK.
- **RCPT TO:<receiver@fel.cvut.cz>**
    - This command defines the final receiver of the message. The address can again be prepended with a list of hosts which define routing information, but this has been deprecated and routing fully relies on the DNS.
    - To send the same message to multiple receivers, multiple RCPT commands have to be used. One RCPT command defines one receiver.
    - The receiver responds with an OK for each receiver.
- **DATA**
    - By this command the sender says I'm ready to send data.
    - Once the receiver end acknowledges this command (by a 354 message), the sender starts sending data, i.e. email header and body. The data is terminated by a line containing only a period character, i.e. the character sequence: "CRLF.CRLF".
- **RSET**
    - Using this command the sender requests reset of the transaction. The receiver deletes all from, to and data information it received before and the transaction starts from scratch.
- **NOOP**
    - The no operation command can be used to keep the connection live and has no other effect.
    - The receiver responds with an OK.
- **QUIT**
    - This is used to end the communication.
    - The receiver responds with an OK and the transmission channel gets closed.

Following commands are also standard SMTP commands but are not used as often.

- **HELP some_string**
  - This can be used to request helpful information about commands. Since most of SMTP communication is done by automatic agents, it does not get used often.
- **TURN**
  - This command is used to switch roles of the SMPT-sender and the SMTP-receiver. This can be used to quickly start sending messages in the opposite direction.
- **VRFY username**
  - The sender can ask the receiver if it knows the user by name *username*.
  - The receiver responds with user's name and mailbox address.
- **EXPN mailinglist**
  - The sender can ask whether *mailinglist* is an existing mailing list.
  - The receiver responds with a list of names and mailboxes of all users on the list.
- **SEND FROM:<handlm1@fel.cvut.cz>**
  - This is the same as MAIL command, but the message is not delivered to the receiver's mailbox, but to their terminal. Terminals are not very common in modern world; therefore the use of the command is very limited.
- **SOML FROM:<handlm1@fel.cvut.cz>**
  - SOML stands for Send or Mail. It causes the receiver to try to deliver the message to the terminal and if unsuccessful to deliver it to the receiving user's mailbox.
- **SAML FROM:<handlm1@fel.cvut.cz>**
  - SAML stands for Send and Mail. It makes the receiver to deliver the message to the terminal and to the mailbox of the receiving user.

The two lists of commands above describe the whole set of SMTP commands.

Extended SMTP

The ESMTP (Extended Simple Mail Transfer Protocol) was defined in RFC 1869 in year 1995. It defines a structure for extensions that the email agents support.

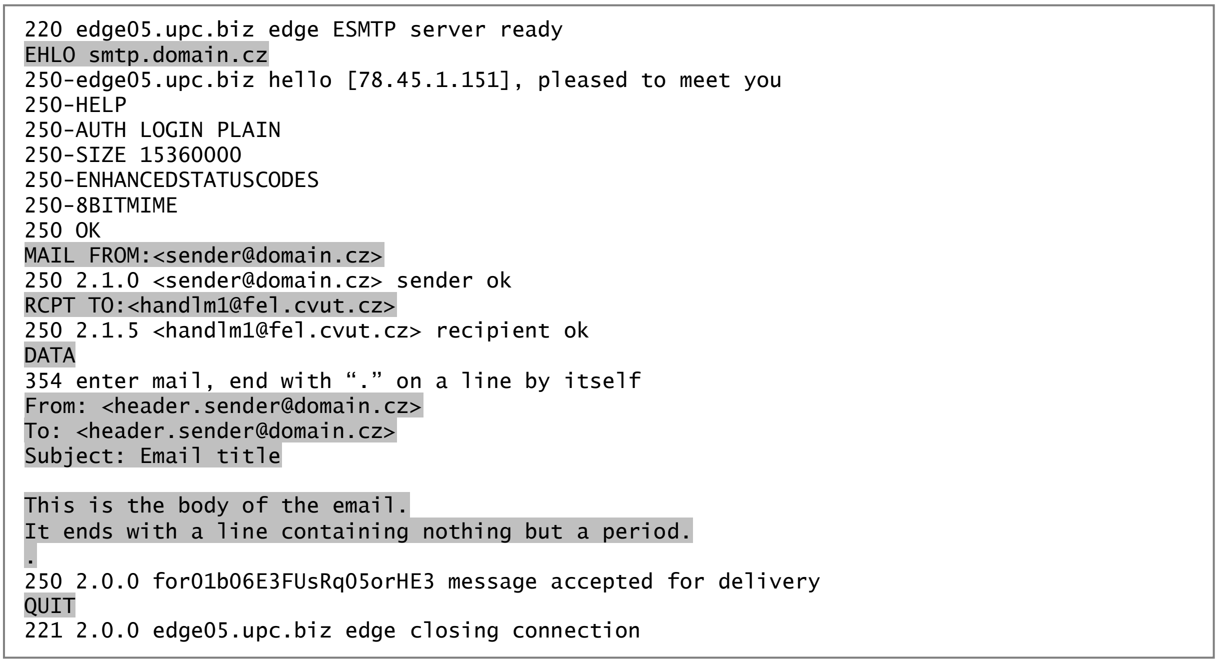
When the sending agent supports ESMTP, it starts the session with EHLO instead of HELO command. If the receiving agent knows ESMTP as well, it will respond with an OK and usually a list of extensions it supports. If it doesn't know ESMTP, the EHLO command will be ignored.

One of the extensions, the 8BITMIME, introduces the possibility to send 8-bit data over SMTP connections. As discussed in the Email Encoding chapter, the email communication was originally intended to use only 7-bit US-ASCII characters. All other characters (or binary data) had to be encoded before transmissions. This introduces overhead and complicates email transmission. The 8BITMIME extension makes the whole process easier.

Another important and widely used extension is the SMTP-AUTH, which is used for authentication of the email client. Some MTA's may require the users to authenticate prior to sending a message. The client sends an AUTH command with the authentication mechanism name as a parameter. The receiver agent responds with a 334 reply. The sender then sends its credentials encoded using the mentioned mechanism. The major authentication mechanisms are LOGIN, PLAIN and CRAM-MD5.

Only the CRAM-MD5 mechanism uses encryption of the credentials and as such is more secure. It is important to realize that the SMTP-AUTH verifies only that the sender has an account on the server, or at least knows one account on the server. It does not control nor verify that the value in MAIL FROM is a valid and authenticated user.

There is also a simple extension called SIZE, which informs the sender about size limits the receiver uses. It defines the maximum size of the message in bytes that will be accepted.



```
220 edge05.upc.biz edge ESMTP server ready
EHLO smtp.domain.cz
250-edge05.upc.biz hello [78.45.1.151], pleased to meet you
250-HELP
250-AUTH LOGIN PLAIN
250-SIZE 15360000
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 OK
MAIL FROM:<sender@domain.cz>
250 2.1.0 <sender@domain.cz> sender ok
RCPT TO:<handlm1@fel.cvut.cz>
250 2.1.5 <handlm1@fel.cvut.cz> recipient ok
DATA
354 enter mail, end with "." on a line by itself
From: <header.sender@domain.cz>
To: <header.sender@domain.cz>
Subject: Email title

This is the body of the email.
It ends with a line containing nothing but a period.
.
250 2.0.0 for01b06E3FUsRq05orHE3 message accepted for delivery
QUIT
221 2.0.0 edge05.upc.biz edge closing connection
```

Figure 4: An example of sending an email on the SMTP level using the ESMTP syntax. Lines with grey background were sent by the sender. Lines with white background are replies from the receiver. It is a screenshot of a communication done from a command line program called telnet, which can be found in major operating systems.

Spam

Introduction

It is not possible to give an exact definition of spam. The New Oxford Dictionary of English describes spam as "*irrelevant or inappropriate messages sent on the Internet to a large number of newsgroups or users*". Often the term unsolicited email is used instead. The abbreviations UBE (Unsolicited Bulk E-mail) and UCE (Unsolicited Commercial E-mail) are sometimes used, mainly in the USA. The opposite of spam, i.e. a message that is welcome by the receiver, is called ham.

As of March 2009, 80-94% of all email messages are spam. It represents more than 100 bilion (10^11) messages a day. During past years the total volume of unsolicited messages grew exponentially. It has leveled off in recent years and based latest researches it grows 1.2% a day.

The history of spam started right at the beginning of email existence. The first message that is considered spam was sent by Gary Thuerk in 1978. Mr. Thuerk sent a commercial message advertising products of the DEC (Digital Equipment Corporation). The message was sent to all addresses that existed in the ARPANET at that time. It was around 600 addresses; due to technical incompleteness of the network the message got delivered into only 398 of them. One of the first modern spam emails was sent out in 1994. At that time Laurence Canter and Martha Siegel misused the email infrastructure by advertising their services to immigrants to the United States of America. It was called the Green Card Spam. Soon after spam become an unstoppable phenomenon.

Most of the spam is sent in order to generate financial revenue. The messages often contain links that lead to web sites that offer some products. Other messages direct the user to fraudulent web sites. These web pages pretend to be pages of some other trustworthy company (e.g. a financial bank) and if the user submits their login credentials on such a page, the spammer gets to know user's password. This method is called *phishing*. There are also spam emails that do not seem to have any sense. These usually are either test messages or address discovery messages, which are explained below.

It is not possible to eliminate spam entirely, because it sometimes is hard to distinguish between regular and unsolicited messages. What one can consider a welcome information message, the other can consider an obtrusive email. However, in vast majority of cases there are no doubts that it is a spam. Furthermore, the amount of this kind of spam can be limited significantly. The spammers often misuse imperfections in the email infrastructure. Even if the email servers adhered to standards, the spammers would still be able to poison the world of email. There are numerous ways how to fight back, though. Techniques used by the spammers as well as techniques used to fight them will be covered in the next subchapters.

Spammer Techniques

Email addresses

The first step a spammer has to do is to get a list of email addresses. There are various ways how to achieve that. The simplest approach is to generate pseudorandom addresses. Although the successful rate of this technique is not high, it is often used, because it is easy to generate millions of addresses in a short amount of time.

Another way to gain addresses is by searching the web. By traversing existing web pages and mailing list discussions it is possible to find a number of existing email addresses. This technique requires having a bot (a piece of software) that can traverse the web automatically, though.

Even more sophisticated method is breaking into a closed database of users. It may be a database of business customers, employees or similar. This not only requires expertise, but also is illegal in many countries and as such is not very common. Some companies even offer databases of email addresses for sale. The origins of addresses are often not clear, some were gained in a legal way by making the user register for a service, but often they are not gathered in an ethic way. Nevertheless, even such companies exist.

It is not uncommon that the spammer also runs another web service; typically a web page requiring registration through email. On these web pages some largely demanded article is offered. These usually include warez (illegally acquired software) or porn.

There are also malicious applications that install itself on user's computer and monitor email communication. It can break into the personal address book or just spy on user's activities. Once it gathers some email addresses it sends them secretly back to the spammer over the Internet.

Blank spam is a term connected with this topic, which is used to describe messages with an empty subject and an empty body. These are used for discovery of existing email addresses. The spammer has generated large number of addresses and now wants to know which are valid. He/she sends a blank or some other email to each address on the list. Those addresses that exist will accept the message. But messages sent to non-existing addresses will bounce back and inform the spammer that the address is invalid.

Botnets

Until a couple of years ago spammers usually used a single computer to send out the messages to a large number of recipients. This has been soon discovered and these computers were quickly put on a blacklist, which made them effectively unable to send emails.

Spammers found a way around it. Instead of using a single computer, they use different computers around the world. These, in most cases, not are their computers, but computers they managed to break into. The common scheme is as follows. The user downloads untrustworthy software, typically some warez or an infected email attachment, and runs it. The application does what the user expected, but besides that it also installs malicious software. This piece of software allows the spammer to control the computer. Such a computer is then called a Zombie computer. Zombies can act autonomously or can be fully dependent on the main controller, the spammer. A botnet is a term for a set of Zombie computers. The spammer can then control large number of computers that span over different parts of the Globe. These computers cannot be blocked easily, partly because there is many of them, but mostly because they represent normal users whose computers have been unfortunately hacked.

Filter workarounds

As will be discussed below, some spam detection methods are based on analysis of email body content. The spammers are aware of the fact; and therefore are trying to make it harder for the filters to discover unsolicited emails.

A typical technique is using intentionally misspelled or modified words. They are easy to understand for a human reader, but can mislead the automatic filter easily. An example are different forms of the word *Viagra*, e.g. V1agra, Via'gra, Vlagra, \/iagra, Vi@gr@ and similar.

Another approach is to use images instead of text in the emails. The image usually contains a text, which is the real content of the message. User can read it easily, but an automatic analysis of the image is a very complex procedure and as such is usually not performed.

It is also possible to put hidden text in the message. This is usually done by using HTML emails with HTML comments or invisible text. This text will not be shown to the user, but the automatic filter will find it and if it contains something meaningful, it will mislead the filter.

Bayesian poisoning is a technique used to fool the Bayesian filter, which have been quite successful in the recent years. The idea is to put fragments of a real and meaningful text into the body of the email. The filter will then evaluate such message as trustworthy.

How to fight it

Blocking the sender

The optimal method of fighting spam would stop the messages even before they get to be sent out. There are many ways how to achieve this and they represent a very effective means of fight.

The easiest thing that can be done is to restrict use of the SMPT port 25. Most of the ISP's offer an SMTP server to be used by their clients. If the server only accepts port 25 connections from the inside of the network, the server cannot be misused by an external spammer. Naturally a spammer that manages to connect from inside will not be affected by this measure.

An email server that accepts a connection from everyone is called an Open relay. In the old days all email servers worked as an open relay, but as the time went by and the spam emerged, administrators started to close the servers. Nowadays only a very few open relays exist and their providers are often criticized for it.

A slightly more advanced technique is requiring an authentication. This is again up to the ISP to implement the functionality. The ESMTP with extension SMTP-AUTH (more on this in chapter Extended SMTP) is supported in all major emailing software. If the ISP required users to authenticate before accepting any message from them, the spammer would not only need to be inside the ISP's network, but also would have to know username and password of a valid ISP client.

The ISP's could also monitor the email traffic on their network. Since spammers mostly send emails in large chunks, the ISP could easily identify them. This would, however, require investments of the ISP, which is why it is usually not performed.

Large portion of applications that are used to send spam are defective software. They often do not follow email standards and can be thus discovered. Unfortunately even email software used for legitimate communication does not fully adhere to standards. Therefore, it is not possible to block everyone who behaves in contrast with the standard, but there definitely is space for improvements in this area. Spam software is oriented on high effectiveness, therefore they often do not wait for responses from the receiving server, they do not finish the connection properly by using the QUIT command etc.

The fact of not following standards is also used in technique called Greylisting. Whenever an MTA receives a connection from an MUA that it does not know, it responds with a temporary unavailable reply (SMTP code 4xx) and stores the MUA's IP address for record. Valid email software will try to resend the message after some time (that is what the standard prescribes), but spam software will simply drop it, because it is not worth waiting. If the same IP address connects in an acceptable timeframe again with the same message, the email will be accepted. Usually not only the IP address is stored but the so called greylist triplet is used instead (sender and recipient from the SMTP envelope and the IP address). This technique is very effective, even against botnets (see above), but it has a downside too. It increases delivery time. It can be matter of minutes but can go up to many hours or even days.

Blacklisting was very effective in the 1990's. During that time spammers used single source mass mailers, i.e. they send all messages from a single computer. Nowadays they use botnets and blacklists lose their effectiveness. DNSBL (Domain Name System Black List) still exist and are kept up to date. They use the DNS infrastructure. Each IP address that has been identified as a spammer is stored in a DNS record on a dedicated DNS server. Whenever an MTA wants to verify that the connecting MUA is not a spammer it does a DNS request asking for information about MUA's IP address. If the DNS server responds with a valid record, the IP address is on the black list and represents a spammer.

Another approach based on IP filtering is based on the country of origin of the sender. It is possible to determine the country where the sender's computer is located from its IP address. Based on statistics email providers know where do emails to their server usually come from. They can they decide to deny senders from uncommon countries, or from countries that send nothing but spam. Although it might seem too restrictive, this technique is used and is effective.

SMTP callback verification is another technique that seemed to be promising. Whenever an MTA receives a connection, it takes the address from the MAIL SMTP command and tries to create a session to send an email to the address. If there is a spammer on the other end, the session will not be established, because the address is usually forged. However, this approach turned out to be unsuccessful. Servers performing callback verification exhibit spammer-like behavior; therefore this method is rarely used.

Technique called Sender Policy Framework (SPF) uses DNS records to verify sender's IP address. At the beginning of the connection session, the receiving MTA takes the IP address of the sender and does a reverse DNS lookup on it. Dedicated DNS server replies with a domain that this IP address belongs into. The MTA does another DNS lookup, this time a forward lookup on the domain. The DNS server replies with a list of IP addresses that are authorized to send emails from this domain. If sender's IP address is not on the list, the connection will be refused. This scheme requires that each

ISP maintains their list of authorized IP addresses. Unfortunately ISP's and email providers are often not willing to implement SPF. Nevertheless, the SPF is one of the most effective measures to fight spam at the moment. Unfortunately there is a downside to the SPF - it breaks when forwarding is used. However, there are workarounds like SRS (Sender Rewriting Scheme) that address this issue. Among major companies that implement SPF belong Google and AOL. There also exists a similar framework called SenderID, which also validates sender email address and is very effective as well.

Spammer traps

Black lists need to be kept up to date to be effective. There are ways how to achieve that. They can be updated manually or based on statistics of email servers. Another approach is to set traps and wait for spammers to eat the bait.

One way is to create a so called Honeypot. It is a server that pretends to be an open relay. Open relays are very welcome by spammers, so if they see any, they are going to use it. However, the relay will not work. All it does is store the IP address of the sender and some statistics. This way the spammers compromise their IP addresses.

Another method uses so called Spamtraps. These are existing email addresses that are, however, never published or are published in an obscure way. If the address is never published there is no way that any valid email gets sent to it. The only way it becomes a recipient is a random generation of email addresses, which is a traditional spammer technique. The other option is to publish the address, but to do it in a way that no human will ever find it (under normal conditions). An example is putting the address into a hidden web page or disguising it on a web page (hidden text or HTML comment). Either way, if an email arrives to an address that has not been published, it is almost for sure that the sender is a spammer. Again, the spammer compromises their IP address.

Email body analysis

Large number of spam detection techniques is based on analysis of email body contents. These methods are not subject of this paper, and therefore will be covered only briefly. The downside of this approach is its computational cost. To do a thorough analysis of email body requires lots of computational time and therefore can often not be performed online. The server simply accepts the message and does the analysis later.

URI DNSBL (Uniform Resource Identifier Domain Name System Black List) uses the fact that spam usually contains a link to some web page. This is a natural consequence of the character of spam. Since these web pages need to exist to prove useful, the spam must contain a valid address. By keeping a list of these links it is then easy to identify spam if it contains one of the listed addresses. The database of addresses is usually built around DNS infrastructure, the same way as DNSBL is (see above).

Spam emails often resemble one another. It is possible to create a checksum of the message and store it in a database. Each incoming message can then be compared to messages stored in the database. However, this approach requires a sophisticated checksum algorithm that will make sure that similar spam email is identified while normal email will not be affected.

Bayesian filtering has been very popular in the recent years. It is based on probability of words in common texts. Each word has particular probability that it occurs in every day communication. These filters are highly customizable and show high success rate.

There is a system called DomainKeys Identified Mail (DKIM), which starts to be used widely. It does not do an analysis of the message, but uses public key cryptography to distinguish between spam and ham. Since it has to compute a signature of mail data (header and body), it is computationally expensive, but this process can be optimized. Each domain that implements DKIM has its private and public keys. Public keys are stored on a DNS server. Private keys are known only to the MTA that is responsible for sending emails from the domain. The process is as follows. Sending MTA computes a hash of the message and encrypts it with the private key. The signature is put into additional email header field called DKIM-Signature. The MTA then sends the message out. Receiving MTA looks at the address from SMTP MAIL command and takes only the domain part of it. Consequently, it does a DNS lookup on the domain. The DNS server replies with a public key for the domain. The receiving MTA decrypts the signature with the public key and compares it with a hash it computed from the message it received. If the two hashes do not match, the email is considered tampered or unsolicited. This verification scheme faces one problem. Some email software modifies the mail data by adding footers to the message. These usually include virus scanner information or commercial ads. If the message is modified, its signature becomes immediately invalid. There is no systematic solution to this problem, the only way out is to make sure the message is not changed under way. DKIM has been implemented among others by Google and Yahoo.

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
       d=gmail.com; s=gamma;
       h=domainkey-signature:received:received:message-id:from:to
        :content-type:content-transfer-encoding:mime-version:subject:date
        :x-mailer;
       bh=aJCKPnU9HYK7wat9Wp6yjz6wwjB9zeann1YFS9qBU0U=;
       b=w7vsH45O4owTzXGlNosLIaRLXaQi/sHcKmd0r70Z2VXr5soc+9Gvl0UURwjkE3rU09
        Uwn2HN5PjwGD1VTnI2gmLD4omLtj9SUOI4zh+RXtPVEthgChWHfu+FWQUluR6jGGlVRY
        5j5xb1jkMPVjyp+WpwN1hs3p9vebHhqrFGof4=
```

Figure 5: An example of a DKIM field that gets inserted into the email header. It defines version of DKIM and encryption algorithm used, states that the email is from gmail domain and lists all headers that were used when computing the signature, the signature is at the end of the field.

Summary

History and technical background of electronic mail has been covered in the first chapter of the paper. An emphasis has been put on standardized format of email, especially on the header fields, and on encoding schemes used to transfer non-english characters. The second chapter focuses on the SMTP protocol, which is used to send emails. All SMTP commands have been covered and an example of SMTP communication has been presented. The final chapter deals with spam problematic. Spam is defined and typical techniques used by spammers are presented. The paper ends with a description and discussion of measures that are used to fight spam.

The final output of the paper can be summarized as follows. Email standards were devised long before the Internet became as ubiquitous as it is today. Nobody could expect which way the electronic mail would go. Therefore; the standards were created in a way that causes problems in nowadays world. Spammers misuse all mistakes in design of the protocols and in the Internet infrastructure. Luckily enough, there are ways how to fight back. Numerous schemes were worked out that can identify and block spam. Unfortunately, these schemes get not implemented as widely as one would expect. Internet Service Providers and email providers have the power to limit the amount of spam significantly, but they are rarely willing to do so. The most promising techniques are SMTP-AUTH, greylisting, SPF and DKIM.

Sources

http://www.ietf.org/rfc.html : Request for Comments repository

http://www.motobit.com : Online Base64, Quoted-Printable and Charset encoders/decoders

http://en.wikipedia.org : Wikipedia, the free encyclopedia

http://wiki.asrg.sp.am/wiki/Taxonomy_of_anti-spam_techniques

http://www.openspf.org : Sender Policy Framework project homepage

http://www.dkim.org : DomainKeys Identified Mail project homepage