MailVortex

A extension to traditional transport protocols to offer anonymity towards third parties

Inauguraldissertation
zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel
von
Martin Gwerder (06-073-787)
von Glarus GL
May 19, 2016

Genehmigt von der Philosophisch-Naturwissenschaftlichen FakultÃd't
Auf Antrag von
Prof. Dr. Christian F. Tschudin
Prof. Dr. Heiko Schuldt

Basel, FIXME datum der fakultÃd'tsversammlung FIXME Signature line with Name of Dean of Faculty below

Contents

List of Tables

List of Figures

1 Introduction

This document describes a solution, which should offer anonymity against third parties when sending emails based on SMTP and the respective client protocols (e.g. IMAPv4 or POP3). This seemed to bother very few peoples up until information in Echelon became public due to an investigation by a commitee of the european parlament in 2001[13]. Things settled again with local peaks up until a whistle blower named Edward Snowden disclosed $200\,000$ documents proofing activities of the NSA and other secret services. This led to the “2013 mass surveillance disclosures” and damaged the reputation of the american nation in many countries[51].

1.1 Overview over the current situation

SMTP as defined in RFC5321[28] is as of today (2013) state of the art transmission protocol for electronic mail. It is standardized in its current version since 2008 and is one of the few protocols, which is marked as "Standard". While the protocol delivers reliable mail transfer between two endpoint (mail servers) the anonymity of the message content towards any mail server is not given (For a detailed analysis see chapter "‘Analysis of current situation"’).

Anonymity against third party is not given due to the following facts.

- There is not always an encryption available between a mail user agent (MUA) and the outgoing mail server.
- There is no way to guarantee that a mail transfer between two SMTP hosts is encrypted.
- There is not always an encryption available between a SMTP host and the MUA of the recipient.
- Encryption based on top level protocols (such as S/MIME or PGP) do hide the message content. The sender, recipient, the subject and some technical information (eg. MIME-Headers) are always in plain available and not protected at all.
- Even if there is a reliable encryption between all endpoints and none of the intermediate servers are compromised sender and recipients might still be identified thru traffic analysis.

Keeping the message content confidential is more and more relevant in these days. The more the importance of mail transfer in today’s economy is growing the more is confidentiality and reliability a topic. Unfortunately Secret Services have already discovered the significance of today’s mail traffic and start to analyse those. With the presence of Secret Services in the internet, actively investigating data the importance of a reliable data channel for today’s messages has become increasingly important.

Quick wins such as the use of "Onion Router Networks" (such as TOR) do not offer any additional security since the message content would be revealed in full to an eventual exit node and any mail server on its way to the recipient.

1.2 Problem statement

This work is an approach to extend the existing mail routing based on SMTP by an intermediate layer, which should offer anonymity against third party.

This work delivers the following results:

- A throughout analysis of current technology and its weaknesses.

  Although the Simple Mail Transfer Protocol (SMTP) is a well-implemented and well proven technology its weaknesses are well known. The SMTP protocol was originally defined in RFC821[42] by Johnathan B. Postel. At this time internet was only available to universities, some mayor companies and governments. The objective of Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently[42, p. 1].

Confidentiality or having a tamper proof protocol was no design goal. Over the years many standards arose trying to close some of the gaps. Some of them are being used but most of them are not very common.

- An analysis of possible approaches to improve the current standards.

  Many standards and technologies do exist these days addressing parts of the issues mentioned above. A throughout research should be carried out to identify how can these technologies be combined to achieve the subsequent goals. Furthermore technology advanced. Namely in the field of cryptology few possibilities and ideas arose (such as new encryption classes [eg. elliptic curves] or the idea of crypto puzzles). Another field of research which emerged in the analysis of traffic flow is handled under the term "Big Data" where not single events but the sum of events is handled.

- A RFC document

  It will describe an approach offering a significant quality improvement of the existing solutions, which could be accepted by the internet community.

  The document has to follow the standards *RFC2223 Instructions to RFC Authors* [40], *RFC2119 Key words for use in RFCs to Indicate Requirement Levels* [3], *RFC3979 Intellectual Property Rights in IETF Technologys* [4] and *RFC5378 Rights Contributors Provide to the IETF Trust* [5].

- A prototype reflecting at least the minimum baseline of the RFC document to reflect prove its functionality.

  A prototype should be offered to show the feasibility. The Prototype should be a reference implementation and offer a quick way to use the new technology. It should be distributed under the LGPL license to simplify distribution of the technology.

1.3 Contributions

This thesis contributes to the topic in the following senses:

- It introduces a consistent model for message delivery which includes all endpoints and involved parties.
- It shows an approach based on existing protocols for anonymous communication which gives full control of the anonymity to the sender while controlling the costs.
- It offers a client application implementing the proposed Protocol as IMAPv4 cache daemon and as SMTP relay.

1.4 Notation

1.4.1 Cryptography

The theory in this document is heavily based on symetric encryption, asymetric encryption and hashing. In order to use a uniformed notation I use $E^{K_a}(M)$ (where a is an index to distinguish multiple keys) resulting in M^{K_a} as the encrypted message. Messages encrypted with multiple keys do list the used keys as a coma separated list in superscript $E^{K_b}\left(E^{K_a}(M)\right) = M^{K_a,K_b}$.

For a symetric encryption of a message M with a key K_a resulting in G^{K_a} where a is an index to distinguish different keys. Decryption uses therefore $D^{K_a}(M^{K_a})$.

As notation for asymetric encryption I use $E^{K_a^1}(M)$ where as K_a^{-1} is the private key and K_a^1 is the public key of a key pair K_a^p. The asymetric decryption is noted as $D^{K_a^{-1}}(M)$.

For hashing I do use $H(M)$ if unsalted and H^{S_a} if using a salted hash with salt S_a. The generated hash is shown as H_M if unsalted and $H_M^{S_a}$ if salted.

$$
\begin{aligned}
asymetric: & E^{K_a^{-1}}(M) && = M^{K_a^{-1}} \\
& D^{K_a^1}\left(E^{K_a^{-1}}(M)\right) && = D^{K_a^{-1}}\left(E^{K_a^1}(M)\right) = M \\
symetric: & E^{K_a}(M) && = M^{K_a} \\
& D^{K_a}\left(E^{K_a}(M)\right) && = M \\
hashing(unsalted): & H(M) && = H_M \\
hashing(salted): & H^{S_a}(M) && = H_M^{S_a}
\end{aligned}
$$

1.4.2 Code and commands

Code blocks are always displayed as light grey block with line numbers:

```
1 public class Hello {
2         public static void main(String args[]) {
3                 System.println("Hello. "+args[1]);
4         }
5 }
```

Commands entered at the command line are in a grey box with top and bottom line. Whenever root rights are required the command line is prefixed with a "#". Commands not requiring specific rights are prefixed with a "~". Lines without a trailing "~" or "#" are output lines of the previous command. If long lines have to be broken to fit into the paper a "↙" is inserted to indicate that the linebrake has been introduced for readability.

```
~ su -
# javac Hello.java
# exit
\$ java Hello
Hello.
\$ java Hello "This is a very long command-line that had to be broken to fit into the code box ↙
      displayed on this page."
Hello. This is a very long command-line that had to be broken to fit into the code box ↙
      displayed on this page.
```

2 Ground theory

2.1 Mail Transport

Todays mail transport is mostly done via SMTP protocol as specified in [28]. This protocol has proven to be stable and reliable. Most of the messages are passed from a MUA to a SMTP relay of a provider. From there the message is directly sent to the SMTP server of the recipient and from there to a server based storage of the recipient. The recipient may at any time connect to his server based storage and may optionally relocate the message to a client based (local) storage. The delivery from the server storage to the MUA of the recipient may happen by message polling or by message push (where as the later is usually implemented by a push-pull mechanism).

To understand the routing of a mail it is essential to understand the whole chain starting from a user(-agent) until arriving at the target user (and being read!). To simplify this I used a consistent model which includes all components (server and clients). The figure 2.1 shows all involved parties of a typical Mail routing. It is important to understand that Mail routing remains the same regardless of the used client. However – Availability of a mail at its destination changes drastically depending on the type of client used. Furthermore control of the mail flow and control is different depending on the client.

The model has three main players storage (derfrefStorage), agent (derfrefAgent) and service (derfrefService). Storages are endpoint storages storing mails. Not explicitely shown are temporary storages such as spooler queues or state storages. Agents are simple programs taking care of a specific job. Agents may be exchangeable by other similar agents. A service is a bundle of agents which is responsible for a specific task or task sets.

In the following paragraphs (for definitions) the term “Mail” is used synonymously to the term “Message”. The reason why “Mail” has been chosen over “Messages” that a lot of terms do already exist in standard documents. In these documents the term mail is commonly used.

Mails are typically initiated by a Mail User Agent (MUA). A MUA accesses a local mail storage which may be the server storage or a local copy. The local copy may be a cache only copy, the only existing storage (when mails are fetched and deleted from the server after retrieval) or a collected representation of multiple server storages (cache or authoritative).

Besides the MUA the only other component accessing a local mail storage is the Mail Delivery Agent (MDA). An MDA is responsible for storing and fetching mails from the local mail storage. Mails destined for other accounts than the current one are forwarded to the MTA. Mails destined to a User are persistently stored in the local mailstorage. It is important to understand that a mailstorage not nessesarily reflects a simple mailbox. It may as well represent multiple mailboxes (eg. a rich client serving multiple IMAP accounts) or a combined view of multiple accounts (eg. a rich client collecting mail from multiple POP accounts. In the case of a rich client the local MDA is part of the software provided by the user agent. In the case of a mail server the local MDA is part of the local Mailstore (not necessarily of the mail transport service).

On the server side there are usually two components (services) at work. A “Mail Transport Service” (MTS) responsible for mail transfers and a “Mail Storage System” which offers the possibility to store received Mails in a local, persistent store.

A MTS consists generally out of three parts. For incoming connects there is a daemon called Mail Receiving Agent (Server MRA) is typically a SMTP listening daemon. A Mail Transfer Agent (MTA) which is responsible for routing, forwarding and rewriting mails. And a Mail Sending Agent (MSA) which is responsible for transmitting mails reliably to another Server MRA (usually sent via SMTP).

A MSS consists out of a local storage and delivery agents which do offer uniform interfaces to access the local store. They do also deal with replication issues and grant should take care of the atomicity of transactions committed to the storage. Typically there are two different kind of MDAs. Local MDAs offer possibilities to access the store via efficient (non network based) mechanisms (eg IPC or named sockets). This is usually done with a stripped down protocol (eg. LMTP). For remote agents there a publicly – network based – agent

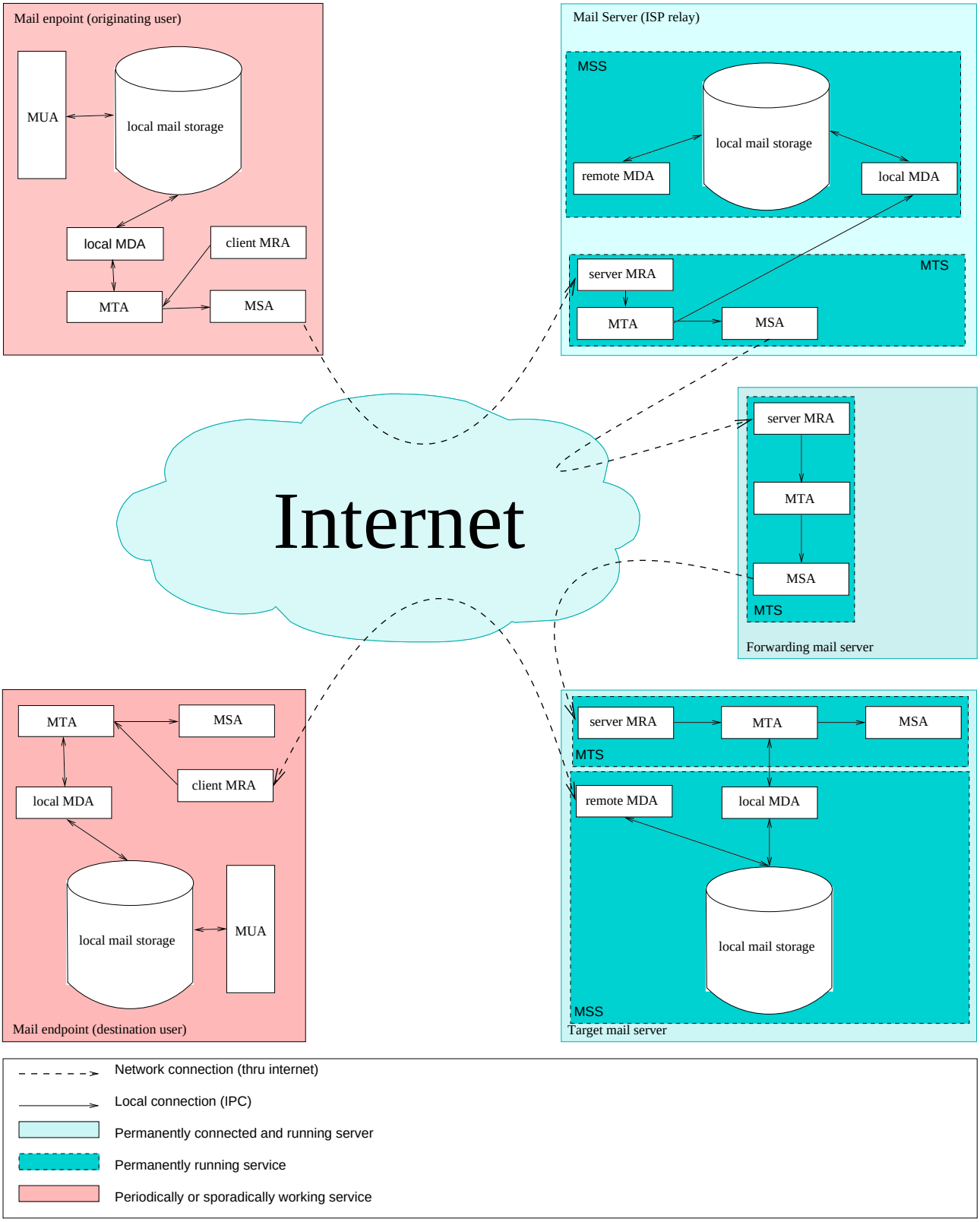

Figure 2.1: Mail Agents

available. Common Protocols for this Remote MDA include POP, IMAP, or MS-OXMAPIHTTP.

2.1.1 Mail Endpoints

Mail endpoints consist typically of the following components:

- A Mail User agent (MUA)
- A Local Mail storage (MUA)
- A Local Mail Delivery Agent (Local MDA)
- A Mail Transfer Agent (MTA)
- A Mail Sending Agent (MSA)
- A Mail Receiving Agent (MRA)

Only two of these components do have external interfaces. These are MSA and MRA. MSA usually uses SMTP as transport protocol. When doing so there are a couple of specialities.

- Portnumber is 587 (specified in [23]).
  Allthough port numbers 25 and 465 are valid and do have usually the same capabilities, they are for mail routing between servers only. Mail endpoints should no longer use them.
- Connections are authenticated.
  Unlike a normal server-to-server (relay or final delivery) SMTP connections on port 25 clients should always be authenticated of some sort. This may be based on data provided by the user (eg. username/-passsword or certificate) or data identifying the sending system (eg. IP address)[23]. Failure in doing authentication may result in this port beeing misused as an sender for UBE.

Mail User Agents (MUA) are the terminal endpoint of a mail delivery. Mail user agents may be implemented as fat clients on a desktop or mobile system or as an interface over a different generic protocol such as HTTP (Web Clients).

Server located clients are a special breed of fat clients. These clients share the properties of fat clients except for the fact that they do not connect to the server. The client application itself has to be run on the server where the mail storage persists. This makes delivery and communication with the server different. Instead of interfacing with a MSA and a client MDA they may directly access the local mail storage on the server. On these systems the local mail storage may be implemented as a database in a user specific directory structure.

2.1.1.1 Fat clients

The majority of mail clients are fat clients. These clients score over the more centralistic organized web clients in the way that they may offer mail availability even if an internet connection is not available (thru a client specific local mail storage). They furthermore provide the possibility to collect mails from multiple sources and store them in the local storage. Unlike Mail servers, clients are assumed to be not always online. In fact they may be offline most of the time. To guarantee the availability of a certain email address a responsible mail server for a specific address collects all mails (this is done by the MSS) and provides a consolidated view onto the database when a client connects thru a local or remote MDA.

As these clients vary strongly it is absolutely mandatory for the MDA that they are well specified. Lack in doing so would result in heavy interoperability problems. Most commonly the Protocols IMAP, POP and EWS are being used these days. For mail delivery the SMTP protocol is used.

Fat clients ar comonly used on mobile devices. According to [6] in Aug 2012 the most common fat email client was Apple Mail client on iOS devices (35.6%), followed by Outlook (20.14%), and Apple Mail (11%). *Email Client Market Share*[15] as a more recent source lists in February 2014 iOS devices with 37%, followed by Outlook (13%), and Google Android (9%).

2.1.1.2 Server located clients

server located clients build an absolute minority. This kind of clients have been used mainly in the days of centralized hosts. An example for a Server Located Client is the Unix command "‘mail"’. This client reads a mail storage from a file in the users home directory.

2.1.1.3 Web clients

Web clients are these days a common alternative to fat clients. Most big provider companies use their own proprietary web client. According to [15] the most common web clients are "‘Gmail"’, "‘Outlook.com"’, and

"'Yahoo! Mail"'. All these Interfaces do not offer a kind of public plugin interface. However, they do offer IMAP-interfaces. This important for a future generalistic aproach to the problem.

2.1.2 Interfaces of Mail Endpoints

There are two interfaces

FIXME

2.2 Anonymity

[**anon_terminology**]

Anonymity is according to Wikipedia[50] defined as follows:

> Anonymity is derived from the Greek word anonymia, meaning "'without a name"' or "'namelessness"'. In colloquial use, anonymity typically refer to the state of an individual's personal identity, or personally identifiable information, being publicly unknown.

A closely related but not identical term is "'Pseudonymity"' that is listed in Wikipedia as well as

> Sometimes it is desired that a person can establish a long-term relationship (such as a reputation) with some other entity, without necessarily disclosing personally identifying information to that entity. In this case, it may be useful for the person to establish a unique identifier, called a pseudonym, with the other entity. Examples of pseudonyms are pen names, nicknames, credit card numbers, student numbers, bank account numbers, and IP addresses. A pseudonym enables the other entity to link different messages from the same person and, thereby, the maintenance of a long-term relationship.
>
> Someone using a pseudonym would be strictly considered to be using "pseudonymity" not "anonymity", but sometimes the term "anonymity" is used to refer to both (in general, a situation where the legal identity of the person is disguised).

and under the "'Pseudonymity"' [56] entry:

> The pseudonym identifies a holder, that is, one or more human beings who possess but do not disclose their true names (that is, legal identities). Most pseudonym holders use pseudonyms because they wish to remain anonymous, but anonymity is difficult to achieve, and is often fraught with legal issues.[2] True anonymity requires unlinkability, such that an attacker's examination of the pseudonym holder's message provides no new information about the holder's true name.

2.2.1 Unobservable Communication

[26]

2.2.1.1 *k-anonymity*

In "k-Anonymous Message Transmission"[1] Ahn, Bortz, and Hopper outline that a individual in most of the cases does not have to be completely anonymous. Instead it might be sufficient to blend into a group of k identities. in [46] Shokri et al. do show that this is not always sufficient.

2.2.1.2 Plausible deniability

According to Wikipedia[55] "'Plausible Deniability refers to "'... a term coined by the CIA in the early 1960s to describe the withholding of information from senior officials in order to protect them from repercussions in the event that illegal or unpopular activities by the CIA became public knowledge. [...] The lack of evidence to the contrary ostensibly makes the denial plausible, that is, credible. The term typically implies forethought, such as intentionally setting up the conditions to plausibly avoid responsibility for one's (future) actions or knowledge."'

2.2.1.3 Deniable Encryption and Deniable Steganography

One form of plausible deniability are the two terms deniable encryption and deniable steganography.

In Wikipedia[54] "'Deniable Encryption"' is defined as ...

> In cryptography and steganography, plausibly deniable encryption is encryption that allows its users to convincingly deny that some specific encrypted data exists, that a given piece of data is encrypted, or that they are able to decrypt a given piece of encrypted data.

[9] FIXME incomplete section

2.2.1.4 Deniable Steganography

FIXME incomplete section

2.2.2 DC-Nets

[7] FIXME incomplete section

2.3 Identification and data signage

FIXME incomplete section

2.4 Steganography

According to Wikipedia steganography[52] is "... the art or practice of concealing a message, image, or file within another message, image, or file.".

Simple steganography applications manipulate the lowest signifficant Bits of lossless compressed images. As the user can not see these kind of variances in colors the image seems to remain "unchanged". Infact the message is obviously embedded in the Image and may be retrieved quite simply. To make the concealed messages less obvoious the message is usually encrypted before embedding. However - most of the embedded messages may be detected as encrypted messages have an equal probability of all characters within an encrypted message. As the lowest significant bits of an image do usually not have this variance (image locally speaking) most of the embedded information may be detected. The same applies to information encoded in the frequency spectrums of lossy packed images. The typical payload of a steganographic message is below 25% of the size of its carrier. The more information is packed into carrier the easier detectable it is.

A good example for this kind of steganalysis is "Steganalysis of JPEG Images: Breaking the F5 Algorithm". The authors have found means in F5[49] to detect the presence of a steganographic concealed message within JPEG images and can even determine its size using relatively simple histogram analysis methods.

Although there are quite some programs available for steganography it has never found a broad audience. Unlike cryptography most progarms implement their own algorithm which is usually not or badly documented and not scientifically analyzed. Interoperability between programs is not given due to a lack of accepted public standards.

FIXME incomplete section

2.5 Encryption and Hashing

With regards to the implementation some of the available cryptographic Algorithms have been chosen (see 6.1.4.4). These cryptographic algorithms are

FIXME incomplete section

2.5.1 Symetric encryption

FIXME incomplete section

2.5.1.1 Advanced Encryption Standard

FIXME incomplete section

2.5.1.2 Camellia

FIXME incomplete section

2.5.2 Asymetric encryption

FIXME incomplete section

2.5.2.1 RSA

FIXME incomplete section

2.5.2.2 Elliptic Curves

FIXME incomplete section

2.5.3 Hashing

FIXME incomplete section

2.5.3.1 Secure Hashing Algorithm

FIXME incomplete section

2.5.3.2 Tiger

FIXME incomplete section

2.6 Mix cascades

FIXME incomplete section

2.7 Remailers

Agents which do accept Mails from one party and forward it to another party while modifying its content well known under the name of “Remailers”. Wikipedia [53] lists four types of Remailers.

Pseudonymous Remailers (or Type-0-Remailers) are remailers that establish a pseudonymity. This means that the senders Email-Address is removed and replaced by a pseudonymous E-Mailadress under the remailers control. This sender address may be used as an ordinary email-Adress to reeach the original sender of the

mail. These types of Remailers allow to send mails while one or both recipients do not know their counterpart. The message (or at least parts of it) might be encrypted but do not have to be. For someone controling the Remailer it will always be possible to make a link between the pseudonymous mail address and a original mailadress. So pseudonymity is only granted towards people not controlling the remailer. Furthermore a person or organisation might be able to discover the Information tuple of Sender and pseudonymous email by analyzing messages and their timely context. So this remailer system is suspectible for traffic analysis.

Cypherpunk-Remailers (or Type-1-Remailers) do function a bit different. They take an encrypted message which was encrypted using the public key of the server, decrypt it and send it to a recipient. The original senders identity gets lost. A reply to a cypherpunk message is not possible. Messages sent to a cypherpunk server might contain messages to other cypherpunk remailers. This daisy-chaining of cypherpunk-nodes allows hiding the original sender-receiver-tuple from a single node. The first node knows only the the originating sender while the last node knows only the final recipient. All intermediate notes do only know the nodes they were linking. However if having traffic information of the entry and exit nodes the tuple might be discovered by traffic analysis.

Mixmaster remailer (or type-2-remailer) is a serie of mailers which split up a message into equally sized chunks and forward them using different paths (via SMTP) to an exit node where the message is reassembled and sent to the final recipient. However if having traffic information of the entry and exit nodes the tuple might be discovered by traffic analysis.

Mixminion remailer (or type-3-remailer) is an enhanced development of Mixmaster remailer. It is currently no longer under active development. It adresses serveral weaknesses of the mixmaster. Namely replies are possible. Forward anonymity is now given. Replay prevention and key rotation is part of the design and there are exit policies allowing ISPs to opt out from receiving remailer traffic. It is based on a proprietary communication network. It furthermore introduces dummy traffic to reduce traceability. This is the most complete approach email anonymity ever given. The aproach has however its weaknesses. To avoid partitioning attacks Miximinon distributes its network information with central redundant directory servers.

2.8 Ethics

2.8.1 Human rights

2.8.1.1 Freedom of speech

Article 19 of the ICCPR states that "everyone shall have the right to hold opinions without interference" and "everyone shall have the right to freedom of expression; this right shall include freedom to seek, receive and impart information and ideas of all kinds, regardless of frontiers, either orally, in writing or in print, in the form of art, or through any other media of his choice".

2.8.2 Ethics of the Internet

There is an RFC document regarding “Ethics and the Internet”[2, p. 1]. Document states as unethical behaveour:

- An activity that seeks to gain unauthorized access to the resources of the Internet.
- An activity that disrupts the intended use of the Internet.
- An activity that wastes resources (people, capacity, computer) through such actions.
- An activity that destroys the integrity of computer-based information.
- An activity that compromises the privacy of users.

Unfortunately these actions do exist in modern internet and the most powerful players discovered so far are governmental agencies. Using a mixer and cryptographic algorithms definitely wastes resources. But it must be considered the right of every single user of the internet to uphold these points. As a final conclusion the proposed system does not violate the ethics of the internet but it must be designed to be as economically as possible with the existing resources.

2.9 Possible legal issues

One of the first questions I have been asked when working for this topic was: Is this legal? The question is important but not easy at all. The mail system is a global spanning network coming across almost any country of the world. Some of these countries consider almost any kind of secret as illegal as long as the country itself is not able to capture it. Some countries consider it as perfectly legal and some will generally accept its presence as long as the country or establishment is not endangered due to its usage. Since there is usually control about mail traffic flow there is no mean to tell what laws have been violated by sending a mail. This is not specific to this work but a general problem which occurs often in connection to the internet.

To give some examples of illegality:

- Bahrain
  According to `http://www.cryptolaw.org` cryptography is not allowed in telecommunications networks using the radio frequency spectrum (see Section 50 Paragraph 2 of the 2002 Telecommunications Law [31])
- Hungary
  According to `http://www.cryptolaw.org` a provision in the Hungarian Digital Signature Act, which entered into force on 1 September 2001, holds that signature-creation data (such as a cryptographic key) shall not be used for other purposes than signing. The ministerial reasoning explains that the intention of this is to prohibit the use of private keys for cryptographic purposes, in the interest of national security. (Note that cryptographic keys not used for creating signatures can be used for encrypting.)
- Moroco
  According to *Loi numero 53-05 relative à l'échange électronique de données juridiques (intégrale)*[32] all cryptography used for encryption of content is illegal in Morocco unless you have a license issued by the government.
- Russia
  According to `http://www.cryptolaw.org` almost all cryptography is illegal in Russia unless you have a license issued by FSB.

3 Current situation

As of today the de facto standard for asynchronous mail transfer is SMTP as defined in RFC5321[28] and its predecessors. While the transfer protocol SMTP is quite compact, the protocol is enhanced with several standards for encryption, multimedia support and similar. A mail client offers today various support for a lot of sub-protocols. The following list is an excerpt of related sub-protocols which are either related to transport, reliability, identification or encryption.

3.1 Implemented protocols

3.1.1 SMTP

The SMTP protocol is currently specified in [28]. It specifies a method to deliver reliably asynchronous mail objects thru a specific transport medium (most of the time the internet). The document splits a mail object into a mail envelope and its content. The envelope contains the routing information which is the sender (one) and the recipient (one or more) in 7-Bit ASCII. The envelope may additionally contain optional protocol extension material.

The content should be in 7-Bit-ASCII (8-Bit ASCII may be requested). It is split into two parts. These parts are the headers (which do contain meta information about the message such as subject, reply address or a comprehensive list of all recipients) and the body which contains the message itself. All lines of the content must be terminated with a CRLF and must not be longer than 998 characters excluding CRLF.

The header consists of a collection of header fields. Each of them is built by a header name, a colon and the data. Exact outline of the header is specified in [45] and is separated with a blank line from the body.

It [28] furthermore introduces a simplistic model for mail communication. A more comprehensive model is introduced in the section Mail Transport. As the proposed model is not sufficient for a comprehensive end-to-end analysis.

3.1.1.1 Mail transport

Messages are transported from MTA to [29]

FIXME incomplete section

3.1.1.2 encryption

Encryption is the only anonymizing technology which is available. There are serveral kind of encryptions which have to be differentiated. Link encryption controls the E-Mail connection a guarantees that the whole communication between two servers is encrypted. It does however not guarantee that the message and routing information is protected all the way thru the network. Message encryption is a weaker encryption which is done at a higher level of the protocol stack. It guarantees that a message is end to end encrypted but discloses all routing and header information.

One kind of Mail link encryption is specified in [25]. This RFC specifies that when a STARTTLS-Command is issued a TLS handshake initiating a encrypted link should be carried out between two Servers. Only not public servers (not published in DNS using MX records) may enforce the use of TLS. All public servers must allow non-TLS transport. Authentication thru this port is possible but usually not done. The STARTTLS specification states clearly that securing a link provides no end-to-end security. An attack to this mechanism is very simple. The only thing required is injecting a 454 error code when the client issues a STARTTLS. According to the document the sending server may then refuse to deliver the document but in reality this never happens in public SMTP servers.

For encryption between a mail endpoint (repective its MSA) and the server MRA Clients may choose to use alternate ports which enforce a TLS handshake at the TCP handshake. This invalidates the possibility to disturb a connection while still in plain text modes with fake errorcodes but since it is a weak security anyway it makes really a difference. According to the [25] document the port 587 should be used. On some servers the same functionality is provided on port 465. This was originally intended for mail transmission between two MSAs. The usage of this port has however never been standardized, violates [24] and the port has been assigned to the URD Protocol by IANA.

The second type of encryption is message encryption. Message encryption does not cover the whole server communication starting from a specific point. It does only cover some parts or the full message body. The Two main protocols in use are S/MIME (As specified in [11]) and PGP/MIME (as specified in [14]; bases on [22]). Both do reveal vital information to all involved parties of the mail transport and a possible third party observer thus completely invalidating anonymity. Informations which are visible to anyone are:

- sender address (may be forged)
- sender client (may be forged)
- Recipient address
- message subject
- the full routing path including all rewrites, timing information and intermediate hops.
- the content type
- Mime-Version
- Date and time of sending

Any client or intermediate Server may furthermore add additional information of any kind (such as virus scanning information, anti spam taxation, reply address).

FIXME unfinished section

3.1.2 MIME

[17] [18] [19] [20] [21] FIXME incomplete section

3.1.2.1 S/MIME

[44] FIXME incomplete section

3.1.2.2 PGP/MIME

[43] FIXME incomplete section

3.1.3 DNS

[12] FIXME incomplete section

3.1.3.1 DNSSEC

[30] FIXME incomplete section

3.1.3.2 Sender Policy Framework

[58] [27] FIXME incomplete section

3.1.3.3 Sender ID

[57] FIXME incomplete section

3.1.4 Transport Protocols

FIXME incomplete section

3.1.4.1 IPv4

[41] [37] [47] [38] [35] [33] [34] [39, p. 3] FIXME incomplete section

3.1.4.2 IPv6

[10] FIXME incomplete section

3.1.4.3 TCP

FIXME incomplete section

3.1.5 Remote MDA protocols

FIXME incomplete section

3.1.5.1 POP3

[36] FIXME incomplete section

3.1.5.2 IMAP

[8] FIXME incomplete section

4 Analysis of current situation

FIXME waiting for this text to appear

4.1 Current state of common Technology

FIXME incomplete section

4.1.1 Mailrouting

FIXME incomplete section

4.1.1.1 SMTP

FIXME incomplete section

4.1.1.2 LMTP

FIXME incomplete section

4.1.1.3 IMAP

FIXME incomplete section

4.1.1.4 POP

FIXME incomplete section

4.1.1.5 MS-OXMAPIHTTP

FIXME incomplete section

4.2 Current state of available Technology

FIXME incomplete section

4.3 Missing Gap

FIXME incomplete section

4.4 Skeleton of Mails and mail transfer

As shown in figure 2.1 on page 6 a typical mail starts with its creation in a mail client. The message is sent by SMTPSMTP to the providers ISP which accepts the mail for relaying if the user is identifiable. As identification an IP or a username/password tupel is typically accepted. The mail is stored in a local queue and the final recipient server is determined by the domain name of the recipient mail address.

The next hop of the mail is determined by looking up the MX record of the recipient domain. If multiple MX records are found then the server with the lowest "'preference number"' is taken and an SMTP daemon tries to send the mail to that host usin SMTP. Unlike the first SMTP server this server will accept the mail without prior authentication as he relays mail for the recipient domain.

The receiving mailserver may apply secondary analysis to the mail (such as anti virus or anti spam analysis) and then pass it to the mailbox of the final recipient.

A recipient may connect at any time to his mailbox thru the providers Remote MDA (Typically using IMAP, pop, or MS-OXMAPIHTTP) and pickup the mail.

The scenario described above is a very simple case of mail delivery and ignores a lot of possible scenarios. All of these Scenarios are usually intermediate steps covered by the SMTP protocol.

5 Designing an approach

In this chapter I lay out the design goals, the main design and the decisions which led to them.

5.1 Defining system boundaries

5.1.1 Thread model

As an adverser we assume the following attributes:

- Available founding is huge.
- Can have own mailer infrastructure.
- Is able to read, write or modify network data freely at any point of the net.

His intensions are:

- Discover message flows
- Discover message contents
- Identify users of the system

5.1.2 User model

The assumed user of the system is:

- Does care about privacy.
- Does or does not have support from a mail server admin.
- Has no special computer knowhow.
- Has the ability to install a program or plugin on his personal computer.
- Has no cryptographic knowhow.
- Is using a device with enough calculation power to solve cryptographic tasks.

His intensions are:

- Send personal or confidential Information securely to another user

His expectations are:

- System should be easy to configure and maintain (in an ideal world: Zero touch).
- System should be fast.
- System should be reliable.
- System should work on any client he is using.
- System should not be a legal problem to him or any of his peers.

5.1.3 Mail server admin model

The assumed mail server admin of the system is:

- Does care about privacy.

- Has considerable computer knowhow.
- Has the ability to install a program or plugin.
- Has possibly no cryptographic knowhow.
- Does know his own mail infrastructure.
- Is using a device with enough calculation power to solve cryptographic tasks.

His intensions are:

- Support his users in sending personal or confidential information securely to another user

His expectations are:

- System should be easy to configure and maintain (in an ideal world: Zero touch).
- System should be fast.
- System should be reliable.
- System should work on any client he is using.
- System should not be a legal problem for him or his company.
- System should still allow him to do regulatory tasks such as virus scanning or backup.

5.2 Basic Requirements of an Approach

Different types of peers must be available:

- Passive
  This peer is sending and receiving only. It works as a endpoint for communication and does only allow relay transfer for messages containing payload for this peer.
- Stealth
  This peer is behaving absolutely passive to unknown peers. No automatic replys are beeing sent unless the sender has been identified. Sender identification may be based on any criteria such as SMTP AUTH, known identity to vortex or knowledge about the public key of the recipient.
- public
  This peer is publicly known to be a participating peer. It may be used for local or relay delivery.

5.2.1 Transport Layer Blending

In order to blend into SMTP-Transport layer the following Criterias should be met:

Criteria	Parameter	Reason
SMTP address	May have non mandatory extensions	SMTP adresses may have extension. However, these extensions must not be mandatory since a target address must be able to be in stealth mode
Transfer channel negotiation	Should always use SMTPS or STARTTLS	Hide immediate peer partners. This increases amount of work to be done for analyzing traffic.
encoding mime message	Should always use Base64	Any other encoding would differentiate MailVotex from regular traffic
Transport media	Attachment	May be any kind of attachment. Recommended are mime types which have no verifiable structures such as .raw or .pcm files to avoid detection thru content analysis.

continued on next page

continued from previous page

Criteria	Parameter	Reason

Table 5.1: Transport layer decisions

FIXME incomplete section

6 Specifying a target solution

6.1 General

We have chosen a general message approach which may use any established message passing protocol as a carrier. In general we use an existing protocol (or protocol set) as carrier to transport the message (transport layer) on top of this layer we have a layer handling all presentation concerns for the transport layer (shaping layer). This layer handles the building of the message (including but not limited to steganographic messages and dead parrot avoidance). The Topmost layer handles all messages, calculates next hops and queues messages (routing layer). It takes furthermore care of the error handling.

The routing layer has a couple of partnering blocks. The main Block is called “router”.

The next important block is the accounting helper. This building block takes care of all states required to manage the system . It stores identities and quotas and authorizes messages to be routed.

The integrity helper collects messages authenticates and checks the integrity. It gets messages from the inbound queue

The queue helper maintains queues of the system.

6.1.1 Application Design

We first have to define the main hook for our solution. While most of the sent messages are always passed from server to server using SMTP the protocols for accessing a mailbox are varying. The most common interfaces within the private mail traffic are POP and IMAP. In the business environment it is mainly IMAP or MS-OXMAPIHTTP.

I will use IMAP as my hook to the mail account to maximize portability. The Solution is based on a IMAP proxy server able to join encrypted delivered data with ordinary data. This hook provides means to turn any mailbox into a vortex

Final goal should be let the device run on a RASPI-Device (or similar) with permanent internet access.

For message sending I will implement an SMTP relay server. This relay server only accepts authenticated messages. The messages shall be scanned for header requesting a specific type of security. The message then is packaged according to these specs and forwarded to the outgoing SMTP server.

All operation carried out on mails are implemented in a thread safe library called libVortex. The API should be stable and suitable for thirdparty apps to be used. Around this a series of connectors are implemented.

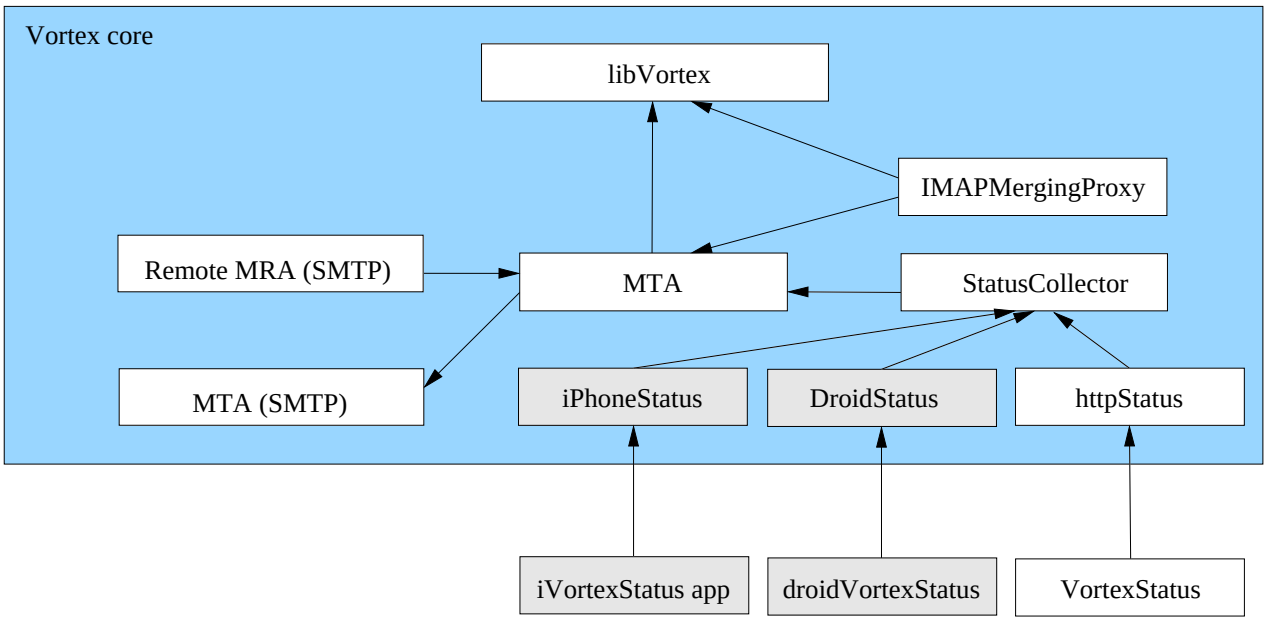

Figure 6.1: Vortex Modules

Main language of the core will be Java for portability reasons. All GUI relevant parts are not part of the core to improve portability.

Platform specific status apps are not created as part of this work. They are however built in a way so that they may

6.1.2 Third Party Code

6.1.2.1 Java

The code will be written in Java 7 compliant code. And allThe JDK beeing used is OpenJDK (`http://openjdk.java.net/`). As a crosscompiler I do use gcj (`http://gcc.gnu.org/java/`). All configuration is either done in files or using a http interface.

6.1.2.2 SQL Server

As the main local database backend H2 is used.

6.1.2.3 Mail Server

Base for all local caching and intermediate mail services a Hedwig Mail Server will be used. However This code will undergo heavy modifications.

6.1.3 Folder storage

The configuration data is stored in a IMAP folder. To hide the IMAP store to an observer the client is provided with a folder name and password (instead of a fixed folder name). The key for all config data and all messages is stored in a message in this folder. The folder is hidden by the proxy if known.

6.1.4 Protocol Design

6.1.4.1 General Design

The protocol is based on serveral Design criterias. It is important to understand that the protocol does not trust any infrastructure exept the own. It is therefore based on a zero trust infrastructure. It splits up into the layers transport, obfuscation and routing. Whereas routing may be split up in accounting, queuing.

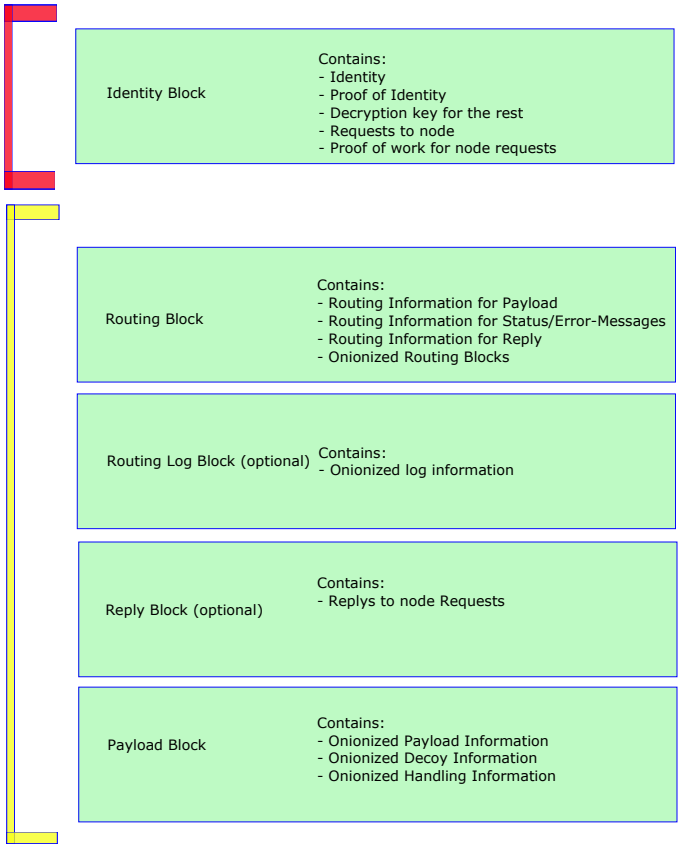

Figure 6.2: Vortex Modules

6.1.4.2 General Process of Communication

FIXME incomplete section

6.1.4.3 General Process of Bootstrapping

FIXME incomplete section

6.1.4.4 General Information about Encryption and Hashing

In This Protocol a lot of encryption and hashing algorithms have been chosen. This Choice should be explained.

First of all we need a subset of encryption algorithms all implementations may rely on. Defining such a subset garatees interoperability between all nodes regardless of their origins.

Secondly we need to have a spectrum of algorithm in such a manor that it may be (a) enlarged if necesary and (b) there is an alternative if an algorithm is broken (so that algorithms may be withdrawn if required without affecting the function in general).

And third due to the onion like design described in this document asymetric encryption should be avoided in favour of symetric encryption to minimize losses due to the key length and the generally higher CPU load oposed by asymetric keys.

If the algorithm is generally bound to specific keysizes (due to S-Boxes or similar constructs) the key size is incorporated into the definition. If not the key size is handled as parameter.

The keysizes have been chosen in such a manor that the keytypes form tupples of approximately equal strength. The support of Camelia192 and Aes192 has been defined as optional. But as they are wildly common in implementations they have allready been standardized as they build a possibility to step up security in future.

Having this criterias for choice I chose to use th following keys and keysizes:

- Symetric
  - AES (Keysizes 128, 192, 256)
  - Camellia (Keysizes 128, 192, 256)
- Asymetric
  - RSA (Keysizes free)
  - DSA
  - Named Elliptic Courves
    - secp384r1
    - sect409k1
    - secp521r1
- Hashing
  - sha384
  - sha512
  - tiger192

6.1.5 Block Details

6.1.5.1 Preamble

The following sections contains exact building information about the blocks used by the Protocol. It contains handling instructions and validity ranges. FIXME incomplete section

6.1.5.2 Routing block

FIXME incomplete section

6.1.5.3 Address request block

FIXME incomplete section

6.1.6 Messages

FIXME incomplete section

6.1.6.1 Basecom

FIXME incomplete section

6.2 libVortex

7 Verification of solution

7.1 User acceptance of the target system

From a perspective of a user Collected requirements to a mail system:

Criteria	Parameter	Weight
The System should be able to transport mails fast under normal conditions	Mails should travel with at least 1MB/min	5
The System should transport mails reliable	Mails should always arrive or their status should be retrievable	9
The System should offer anonymity against spying from third parties	Neither original sender nor final destination or any part of the message content should be determinable by any part of the system except for the original sender and the final recipient.	9
The system must be easy to handle		8
The system must be easy to install	Installation should be almost a “single-click”-Thing. Details should be copied or accessed from the existing configurations.	8
Messages should be prepared fast		8
The degree of anonymity should be controllable		8

Table 7.1: User acceptance requirements

FIXME incomplete section

7.2 Admin acceptance of the target system

Collected requirements to a mail system from an admin perspective:

Requirement	
The cost of mail routing should be controllable The system should not be missusable for UCE	under

Table 7.2: Admin aceptance requirements

FIXME incomplete section

7.3 Possible attacks to the system

FIXME incomplete section

7.3.1 Generic DoS attacks

It is always possible to overload a system. However due to the combination with cryptopuzzles it is very hard for an attacker to use costly system ressources (such as cpu for decrypting or encrypting messages) without having far higher resource costs on his side. FIXME

7.3.1.1 Overloading single nodes

The cost for detecting an illic message are very small (just two or three cypher blocks) while the costs for generating load are very high. FIXME

7.3.2 Attacks on the users anonymity

FIXME incomplete section

7.3.3 Reputaional attacks

FIXME incomplete section

7.3.3.1 Misuse for sending spam

FIXME incomplete section

7.3.3.2 Misuse for covering illegal actions

FIXME incomplete section

Glossary

adverser FIXME

Agent FIXME

EWS FIXME

IMAP IMAP (currently IMAPv4) is a typical protocol to be used between a Client MRA and a Remote MDA. It has been specified in its current version in [8]. The protocol is capable of fully maintaining a server based message store. This includes the capability of adding, modifying and deleting messages and folders of a mailstore. It does not include however sening mails to other destinations outside the server based store.

LMTP FIXME

Local Mail Store A Local Mail Store offers a persistent store on a local non volatile memory in which messages are beeing stored. A store may be flat or structured (eg. supports folders). A Local Mail Store may be an authoritative store for mails or a "‘Cache Only"’ copy. It is typically not a queue.

mail server admin FIXME

MDA An MDA provides an uniform access to a Local Mail Store.

Remote MDA A Remote MDA is typically supporting a specific access protocol to access the data stored within a Local Mail Store .

Local MDA A Local MDA is typically giving local applications access to a server store. This may be done thru an API, a named socket or similar mechanisms.

MRA A Mail receiving Agent. This agent receives mails from a agent. Depending on the used protocol two subtypes of MRAs are available.

Client MRA A client MRA picks up mails in the server mail storage from a remote MDA. Client MRAs usually connect thru a standard protocol which was designed for client access. Examples for such protocols are POP or IMAP

Server MRA Unlike a Client MRA a server MRA listens passively for incomming connections and forwardes received Messages to a MTA for delivery and routing. A typical protocol supported by an Server MRA is SMTP

MS-OXMAPIHTTP FIXME

MSA A Mail Sending Agent. This agent sends mails to a Server MRA.

MTA A Mail Transfer Agent. This transfer agent routes mails between other components. Typically an MTA receives mails from an MRA and forwardes them to a MDA or MSA. The main task of a MTA is to provide reliable queues and solid track of all mails as long as they are not forwarded to another MTA or local storage.

MTS A Mail Transfer Service. This is a set of agents which provide the functionallity tor send and receive Messages and forward them to a local or remote store.

MSS A Mail Storage Service. This is a set of agents providing a reliable store for local mail accounts. It also provides Interfacing which enables clients to access the users mail.

MUA A Mail User Agent. This user agent reads mails from a local storage and allows a user to read existing mails, create and modify mails.

Privacy From the Oxford English Dictionary: “

1. The state or condition of beeing withdrawn from the society of others, or from the public intrest; seclusion. The state or condition of beeing alone, undisturbed, or free from public attention, as a matter of choice or right; freedom from interference or intrusion.
2. Private or retired place; private apartments; places of retreat.
3. Absence or avoidance of publicity or display; a condition approaching to secrecy or concealment. Keeping of a secret.

4. A private matter, a secret; private or personal matters or relations; The private parts.
5. Intimacy, confidential relations.
6. The state of being privy to some act.

”[48, FIXME]
In this work privacy is related to definition two. Mails should be able to be handled as a virtual private place where no one knows who is talking to whom and about what or how frequent (except for directly involved people).

POP POP (currently in version 3) is a typical protocol to be used between a Client MRA and a Remote MDA. Unlike IMAP it is not able to maintain a mail store. Its sole purpose is to fetch and delete mails in a server based store. Modifying Mails or even handling a complex folder structure is not doable with POP

Service FIXME

SMTP SMTP is the most commonly used protocol for sending mails across the internet. In its current version it has been specified in [28].

Storage A store to keep data. It is assumed to be temporary or persistent in its nature.

user FIXME

UBE FIXME

Bibliography

[1] Luis von Ahn, Andrew Bortz, and Nicholas J. Hopper. “k-Anonymous Message Transmission”. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*. Ed. by Vijay Atluri and Peng Liu. ACM Press, Oct. 2003, pp. 122–130. DOI: 10.1145/948109.948128. URL: http://www.abortz.com/papers/k-anon.pdf (cit. on p. 8).

[2] Internet Activities Board. *RFC1087 Ethics and the Internet*. Ed. by IETF. IETF, 1989. URL: http://tools.ietf.org/pdf/rfc1087.pdf (cit. on p. 11).

[3] S. Bradner. *RFC2119 Key words for use in RFCs to Indicate Requirement Levels*. IETF, 1997. URL: http://tools.ietf.org/pdf/rfc2119.pdf (cit. on p. 2).

[4] S. Bradner. *RFC3979 Intellectual Property Rights in IETF Technologys*. IETF, 2005. URL: http://tools.ietf.org/pdf/rfc3979.pdf (cit. on p. 2).

[5] S. Bradner and J. Contreras. *RFC5378 Rights Contributors Provide to the IETF Trust*. IETF, 2008. URL: http://tools.ietf.org/pdf/rfc5378.pdf (cit. on p. 2).

[6] *Campaign Monitor*. 2012. URL: http://www.campaignmonitor.com/resources/will-it-work/email-clients/ (cit. on p. 7).

[7] David Chaum. “The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability”. In: *Journal of Cryptology* 1 (1988), pp. 65–75. URL: http://www.cs.ucsb.edu/~ravenben/classes/595n-s07/papers/dcnet-jcrypt88.pdf (cit. on p. 9).

[8] M. Crispin. *RFC3501 INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*. IETF, 2003. URL: http://tools.ietf.org/pdf/rfc3501.pdf (cit. on pp. 15, 31).

[9] Alexis Czekikis, David J. St. Hilare, Karl Koscher amd Sven D. Gribble, Taddayoshi Kohno, and Bruce Schneier. “Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications”. In: (2008). URL: https://www.schneier.com/paper-truecrypt-dfs.pdf (cit. on p. 9).

[10] S. Deering and R. Hinden. *RFC2460 Internet Protocol, Version 6 (IPv6) Specification*. IETF, 1983. URL: http://tools.ietf.org/pdf/rfc2460.pdf (cit. on p. 15).

[11] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. *RFC2311 S/MIME Version 2 Message Specification*. IETF, 1998. URL: http://tools.ietf.org/pdf/rfc2311.pdf (cit. on p. 14).

[12] D. Eastlake, E. Brunner-Williams, and B. Manning. *BCP42 Domain Name System (DNS) IANA Considerations*. IETF, 2000. URL: http://tools.ietf.org/pdf/rfc2929.pdf (cit. on p. 14).

[13] Temporary Committee on the ECHELON Interception System. *REPORT on the existence of a global system for the interception of private and commercial communications (ECHELON interception system)*. 2001. URL: http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+REPORT+A5-2001-0264+0+DOC+PDF+V0//EN&language=EN (cit. on p. 1).

[14] M. Elkins. *RFC2015 MIME Security with Pretty Good Privacy (PGP)*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2015.pdf (cit. on p. 14).

[15] *Email Client Market Share*. 2014. URL: http://emailclientmarketshare.com/ (cit. on p. 7).

[16] Jessica Fidrich, Miroslav Goljan, and Dorin Hogea. “Steganalysis of JPEG Images: Breaking the F5 Algorithm”. In: (2002). URL: http://www.ws.binghamton.edu/fridrich/research/f5.pdf (cit. on p. 9).

[17] N. Freed and N. Borenstein. *RFC2045 Multipurpose Internet Mail Extensions; (MIME) Part One: Format of Internet Message Bodies*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2045.pdf (cit. on p. 14).

[18] N. Freed and N. Borenstein. *RFC2046 Multipurpose Internet Mail Extensions; (MIME) Part Two: Media Types*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2046.pdf (cit. on p. 14).

[19] N. Freed and N. Borenstein. *RFC2047 Multipurpose Internet Mail Extensions; (MIME) Part Three: Message Header Extensions for Non-ASCII Text*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2046.pdf (cit. on p. 14).

[20] N. Freed, J. Klensin, and J. Postel. *RFC2048 Multipurpose Internet Mail Extensions; (MIME) Part Four: Registration Procedures*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2048.pdf (cit. on p. 14).

[21] N. Freed, J. Klensin, and J. Postel. *RFC2049 Multipurpose Internet Mail Extensions; (MIME) Part Five: Conformance Criteria and Examples*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2049.pdf (cit. on p. 14).

[22] J. Galvin, S. Murphy, S. Crocker, and N. Freed. *RFC1847 Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*. IETF, 1995. URL: http://tools.ietf.org/pdf/rfc1847.pdf (cit. on p. 14).

[23] R. Gellens and J. Klensin. *RFC4409 Message Submission for Mail*. IETF, 2006. URL: http://tools.ietf.org/pdf/rfc4409.pdf (cit. on p. 7).

[24] R. Gellens and J. Klensin. *STD72 Message Submission for Mail*. IETF, 2011. URL: http://tools.ietf.org/pdf/rfc6409.pdf (cit. on p. 14).

[25] P. Hoffman. *RFC3207 SMTP Service Extension for Secure SMTP over Transport Layer Security*. IETF, 2002. URL: http://tools.ietf.org/pdf/rfc3207.pdf (cit. on pp. 13, 14).

[26] Amir Houmansadr, Giang T. K. Nguyen, Matthew Caesar, and Nikita Borisov. "Cirripede: Circumvention Infrastructure using Router Redirection with Plausible Deniability". In: *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*. Oct. 2011. URL: http://hatswitch.org/~nikita/papers/cirripede-ccs11.pdf (cit. on p. 8).

[27] S. Kitterman. *RFC6652 Sender Policy Framework (SPF) Authentication Failure Reporting Using the Abuse Reporting Format*. IETF, 2012. URL: http://tools.ietf.org/pdf/rfc6652.pdf (cit. on p. 14).

[28] J. Klensin. *RFC5321 Simple Mail Transfer Protocol*. IETF, 2008. URL: http://tools.ietf.org/pdf/rfc5321.pdf (cit. on pp. 1, 5, 13, 32).

[29] J. Klensin, N. Freed, and K. Moore. *RFC1870 SMTP Service Extension for Message Size Declaration*. IETF, 1995. URL: http://tools.ietf.org/pdf/rfc1870.pdf (cit. on p. 13).

[30] B. Laurie, G. Sisson, R. Arends, and D. Blacka. *RFC5155 DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*. IETF, 2008. URL: http://tools.ietf.org/pdf/rfc5155.pdf (cit. on p. 14).

[31] *LEGISLATIVE DECREE NO. 48 OF 2002 PROMULGATING THE TELECOMMUNICATIONS LAW*. 2002. URL: http://www.ictregulationtoolkit.org/Documents/Document/Document/1453 (cit. on p. 12).

[32] *Loi numero 53-05 relative à l'échange électronique de données juridiques (intégrale)*. 2007. URL: http://droitmaroc.wordpress.com/2008/01/29/loi-n%C2%B0-53-05-relative-a-lechange-electronique-de-donnees-juridiques-integrale/ (cit. on p. 12).

[33] J. Mogul. *RFC922 BROADCASTING INTERNET DATAGRAMS IN THE PRESENCE OF SUBNETS*. IETF, 1984. URL: http://tools.ietf.org/pdf/rfc922.pdf (cit. on p. 15).

[34] J. Mogul and J. Postel. *RFC950 Internet Standard Subnetting Procedure*. IETF, 1985. URL: http://tools.ietf.org/pdf/rfc950.pdf (cit. on p. 15).

[35] Jeffrey Mogul. *RFC919 BROADCASTING INTERNET DATAGRAMS*. IETF, 1984. URL: http://tools.ietf.org/pdf/rfc919.pdf (cit. on p. 15).

[36] J. Myers and M. Rose. *RFC1939 Post Office Protocol - Version 3*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc1939.pdf (cit. on p. 15).

[37] J. Postel. *RFC791 INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*. IETF, 1981. URL: http://tools.ietf.org/pdf/rfc791.pdf (cit. on p. 15).

[38] J. Postel. *RFC792 INTERNET CONTROL MESSAGE PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*. IETF, 1981. URL: http://tools.ietf.org/pdf/rfc792.pdf (cit. on p. 15).

[39] J. Postel. *RFC793 TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*. IETF, 1981. URL: http://tools.ietf.org/pdf/rfc793.pdf (cit. on p. 15).

[40] J. Postel and J. Reynolds. *RFC2223 Instructions to RFC Authors*. IETF, 1997. URL: http://tools.ietf.org/pdf/rfc2223.pdf (cit. on p. 2).

[41] Jon Postel. *RFC760 DOD STANDARD INTERNET PROTOCOL*. IETF, 1980. URL: http://tools.ietf.org/pdf/rfc760.pdf (cit. on p. 15).

[42] Jonathan B. Postel. *RFC821 Simple Mail Transfer Protocol*. IETF, 1982. URL: http://tools.ietf.org/pdf/rfc821.pdf (cit. on p. 1).

[43] B. Ramsdell. *RFC2440 Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification*. IETF, 2004. URL: http://tools.ietf.org/pdf/rfc2440.pdf (cit. on p. 14).

[44] B. Ramsdell. *RFC3851 Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification*. IETF, 2004. URL: http://tools.ietf.org/pdf/rfc3851.pdf (cit. on p. 14).

[45] P. Resnick. *RFC5322 Internet Message Format*. IETF, 2008. URL: http://tools.ietf.org/pdf/rfc5322.pdf (cit. on p. 13).

[46] Reza Shokri, Carmela Troncoso, Claudia Diaz, Julien Freudiger, and Jean-Pierre Hubaux. "Unraveling an Old Cloak: k-anonymity for Location Privacy". In: *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2010)*. Chicago, IL, USA: ACM, Oct. 2010. URL: http://infoscience.epfl.ch/record/150348/files/ShokriTDFH-WPES10_1.pdf?version=2 (cit. on p. 8).

[47] T. Socolofsky and C. Kale. *RFC1180 A TCP/IP Tutorial*. IETF, 1991. URL: http://tools.ietf.org/pdf/rfc1180.pdf (cit. on p. 15).

[48] A. Stevenson. *Oxford Dictionary of English*. Oxford reference online premium. OUP Oxford, 2010. ISBN: 9780199571123. URL: http://www.oed.com (cit. on p. 32).

[49] Andreas Westfeld. “F5 - A Steganographic Algorithm”. In: *none* none (2002). URL: http://www.ws.binghamton.edu/fridrich/research/f5.pdf (cit. on p. 9).

[50] Wikipedia. *Anonymous remailer*. 2013. URL: http://en.wikipedia.org/w/index.php?title=Anonymous_remailer&oldid=584455506 (cit. on p. 8).

[51] Wikipedia. *Edward Snowden*. 2013. URL: http://en.wikipedia.org/w/index.php?title=Edward_Snowden&oldid=586147644 (cit. on p. 1).

[52] Wikipedia. *Steganographie*. 2013. URL: https://en.wikipedia.org/wiki/Steganographie (cit. on p. 9).

[53] Wikipedia. *anonymity*. 2014. URL: https://en.wikipedia.org/wiki/Anonymity (cit. on p. 10).

[54] Wikipedia. *DeniablEncryption*. 2014. URL: hhttps://en.wikipedia.org/wiki/Deniable_encryption (cit. on p. 9).

[55] Wikipedia. *Plausible deniability*. 2014. URL: https://en.wikipedia.org/wiki/Plausible_deniability (cit. on p. 8).

[56] Wikipedia. *Pseudonymity*. 2014. URL: https://en.wikipedia.org/wiki/Pseudonymity (cit. on p. 8).

[57] N. Williams. *RFC4401 Sender ID: Authenticating E-Mail*. IETF, 2006. URL: http://tools.ietf.org/pdf/rfc4401.pdf (cit. on p. 15).

[58] N. Williams. *RFC4408 Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*. IETF, 2006. URL: http://tools.ietf.org/pdf/rfc4408.pdf (cit. on p. 14).