MessageVortex

Transport Independent Messaging anonymous to 3rd 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
November 7, 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

Almon Brown Strowger was owner of a funeral parlour in St. Petersburg. He filed a patent in Mach 10th 1891 for an “Automatic Telephone Exchange” [23]. This patent built the base for modern automated telephone systems. According to several sources he was annoyed by the fact that his local telephone operator was married to another undertaker. She diverted potential customers of Mr. Strowger to her huband instead causing him to lose business. In 1922 this telephone dialing system which is nowadays called pulse dialing became the standard dialing technology for more than 70 years until it was replaced by tone dialing.

This dialing technology enabled automatic messaging for voice and text messages (e.g. telex) up until today and built the base technology for today’s routed networks.

These networks are building the base

Numerous events in present and past have shown that data is broadly collected in the internet. Wether this is a problem or not may be a disputable fact. Undisputed is however that if data is not handled with care peoples are accused with numerous “facts” that are more than questionable. To show that this may happen even under complete democratic control we might refer to events such as the “secret files scandal?? (or “Fichenskandal”) in Switzerland. In the years from 1900 to 1990 Swiss government collected 900’000 files in a secret archive (covering roughly 10% of the natural and juristic entities within Switzerland at that time).

A series of similar attempts to attack privacy on a global scale have been discovered by whistle blower Edward Snowden. The documents leaked in 2009 by him claim that there was a data collection starting in 2010. Since these documents are not publicly available it is hard to to prove claims based on these documents. However – due to the fact that the documents were screened by a significant number of journalists spanning multiple countries, the information seems credible.

According to these documents (verified by NRC) NSA infiltrated more than 50k computers with malware to collect classified information. They furthermore infiltrated Telecom-Operators (executed by british GCHQ) such as Belgacom to collect data and targeted high member of governments even in associated states (such as the germans president mobile phone). A later published shortened list of “selectors” showed 68 telephone and fax numbers targeting enconomy, finance and agricultural parts of the german government.

This list of events shows that big players are collecting and storing vast amounts of data for analysis or possibly future use. The list of events shows also that the use of this data has in the past been at least partially questionable. As a part of possible counter measures this work analyses the possibility of using state of the art technology to minimize the information footprint of a person on the internet.

We leave a vast information footprint in our daily communication. On a regular email we disclose everything in an “postcard” to any entity on its way. Even when encrypting a message perfectly with today’s technology (S/MIME[9] or PGP[8]) leaves at least the originating and the receiving entity disclosed or we rely on the promises of a third party provider which offers a proprietary solution. Even in those cases we leak informations such as “message subject”, “frequency of exchanged messages”, “size of messages”, or “client beeing used”. A good anonymity protocol has therefore far more attributes to cover than the message itself. It includes beside the message all metadata and the traffic flows. Furthermore a protocol anonymising messages should not rely on the trust of infrastructure other than the infrastructure under control of the sending or receiving entity.

Any central infrastructure is bound to be of special interest to anyone gathering data concerning the using entities of such a protocol. It furthermore may be manipulated in order to attack the messages or their flow. So central infrastructure has to be avoided.

In this work a new protocol is designed to allow message transfer through existing communication channels. These messages should be unobservable to any third party. This unobservability does not only cover the message itself but all metadata and flows associated with it. The protocol should be designed in such a way so that it is capable to use any type of transfer protocol. It should be even possible to switch protocols while have messages in transfer to allow media breaches (at least on a protocol level).

The new protocol should allow safe communication without the need of trusting the underlying transport media. Furthermore it is desirable that the usage of the protocol itself is possible without altering the immediate behaveour of the transport layer. That way it is possible to use the transport layers normal traffic to increase the noise in which information has to be searched.

This work splits into multiple parts. In the first part we are collecting available researches and technologies. We emphasize in all technologies on the strength and weaknesses relevant to this work. In the second part we reassemble the parts to a new protocol. In the third part we analyse the protocol for fitness of the purpose.

We try to find weaknesses and work out recommendations for the protocol usage. In the last part we discuss the results and try to summarize the findings. Try to elaborate to what extend the protocol fullfills the requirements of this work.

2. Main Research Question

The main topic of this thesis was defined as follows:

- Is it possible to have specialized messaging protocol used in the internet based on “state of the science” technologies offering a high level of unlikability (sender and receiver anonymity) towards an adverser with a high budget and privileged access to internet infrastructure?

Based on this main question there are several sub questions grouped around various topics:

1. What technologies and methods may be used to provide sender and receiver anonymity and unlinkability when sending messages against a potential adverser? (RQ1)
2. How can entities utilizing these technologies and methods be attacked and what measures are available to circumvent such attacks? (RQ2)
3. How can attacks targeting anonymity of a sending or receiving entity be mitigated by design? (RQ3)

2.1. RQ1: Sending messages maintaining unlikability against an adverser

This question covers the principal part of the work. As a result we get a list of suitable technologies. Based on this list we define a protocol combining these technologies and researches to a solution. This solution will be implemented and analysed for suitability based on the criteria defined.

2.2. RQ2: Attacking unlinkability and circumvention

Within this questions we look at common attacks and test resistance of the protocol based on the definition of the protocol.

2.3. RQ3: Mitigation by design

Within this question we define baselines in order to mitigate common attacks by design. We the flow these guidelines back into the protocol thus hardening it further.

Part I.

Methodes

In this part of the thesis we collect requirements, definitions, methods, and existing research relevant to the topic of this thesis. We start with collecting requirements for the protocol.

Having the requirements we collect numerous existing technologies on research and implementation level. Each of the technologies is quickly categorized and either further studied or rejected naming the reasons for rejection.

The list of technologies and research collected is big in this chapter. All relevant technologies either widely adopted or thoroughly researched should be included in this chapter. All Technologies and research are categorized.

Technologies are always referenced through their respective standard. If applicable multiple standards may be part of the analysis. A very quick introduction into the protocol is given and then analysed for suitability for a specific problem addressed in this work.

Research are referenced by a paper. If applicable multiple related researches and papers are collected together into a bigger picture an then analysed for suitability concerning specific problem. If related to a standard technology links to that technology are provided where relevant.

3. Requirements for an anonymising Protocol

In the following sections we try to elaborate the main characteristics of the anonymising protocol.

The main goal of the protocol is to enable Freedom of speech as defined in Article 19 of the International Covenant on Civil and Political Rights (ICCPR)[25].

> 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.

We imply that not all participants in the internet share this value. As of September 1st 2016 Countries such as China (signatory), Cuba (signatory), Quatar, Saudi Arabia, Singapore, United Arab Emirates, or Myanmar did not ratify the ICCPR. Other countries such as United States or Russia did either put local laws in place superseding the ICCPR or made reservations rendering parts of it ineffective. We may therefore safely assume that freedom of speech is not given in the internet as a lot of countries explicitly supersede them.

We always have to keep in mind that we have no control over the flow of data packets in the internet. Packets may pass though any point of the world. It is not even possible to detect what way has a packet taken. The common network diagnostic tool `tracert` respectively `traceroute` tries to figure that out. But neither can a route of a packet beeing sent forseen nor can it be measured while or after sending. This is due to the fact that all routers along the way only decide for the next hop of a packet.

As an example of the problems analysing a packet route we may look at `traceroute`. The mapage of traceroute of Linux tells us that `traceroute` uses UDP, TCP, or ICMP packets with a short TTL and analyses the IP of the peer sending an TIME_EXCEEDED (message of the ICMP protocol). This information is then collected and shown as a route. This route may be completely wrong. Some of the possible cases are described in the manpage.

The output of `traceroute` is therefore not a reliable indication of route. Since routes do not have to be static and may be changed or even be alternating the output represents in the best case a snapshot of the current routing situation. We cannot be sure that data packets we are sending are passing only through countries accepting the ICCPR to the full extend.

```
1  $ traceroute www.ietf.org
2  traceroute to www.ietf.org.cdn.cloudflare-
       dnssec.net (104.20.0.85), 64 hops max
3  1    147.86.8.253   0.418ms   0.593ms   0.421ms
4  2    10.19.0.253   1.177ms   0.829ms   0.782ms
5  3    10.19.0.253   0.620ms   0.427ms   0.402ms
6  4    193.73.125.35   1.121ms   0.828ms   0.905ms
7  5    193.73.125.81   2.991ms   2.450ms   2.414ms
8  6    193.73.125.81   2.264ms   1.961ms   1.959ms
9  7    192.43.192.196   6.472ms   199.543ms
       201.152ms
10 8    130.59.37.105   3.465ms   3.138ms   3.121ms
11 9    130.59.36.34   3.904ms   3.897ms   4.989ms
12 10    130.59.38.110   3.625ms   3.333ms   3.379ms
13 11    130.59.36.93   7.518ms   7.232ms   7.246ms
14 12    130.59.38.82   7.155ms   17.166ms   7.034ms
15 13    80.249.211.140   22.749ms   22.415ms
       22.467ms
16 14    104.20.0.85   22.398ms   22.222ms   22.146ms
```

Figure 3.1.: A traceroute to the host www.ietf.org

3.1. Anonymizing and unlinking

If we are unable to limit the route of our packets through named jurisdictions we must protect ourselves from unintentionally breaking the law of a foreign country. Therefore we need to be anonymous when sending or receiving messages. Unfortunately most transport protocols (in fact almost all of them such as SMTP, SMS, XMPP or IP) use a globaly unique identifier for senders and receivers which are readable by any party which is capable of reading the packets.

As a result anonymisation of a sender or a receiver is not simple. If messages are being sent through a relay at least the original sender might be concealed (Sender anonymity). By combining it with encryption we may even achieve a very simple form of sender and receiver pseudonymity. If cascading more relay like infrastructures and combining it with cryptography we might even achieve sender and receiver anonymity. This approach has however serveral downsides (see 5.10.1.4 and 5.10.1.2 for details) and is easily attackable.

3.2. Censorship Resistant

In our scenario in 3 we defined the adverser as someone with superior access to the internet and its infras-

tructure. Such an adverser might attack a message flow in serveral ways:

- Identify sender
- Identify recipient
- Read messages passed or extract meta information
- Disrupt communication fully or partially

We furthermore have to assume that all actions taken by a potential adverser are not subject to legal prosecution. This is due to the fact that an adverser trying to establish censorship may be part of a jurisdictions government. We may safely assume that there are legal exceptions in some jurisdiction for such entities.

In order to be able to withstand an adverser outlined above the protocol needs certain attributes. The message content needs to be unidentifiable by attributes or content. Whereas "Attributes" include meta information such as frequency, timing, message size, sender, protocol, ports, or recipient (list not conclusive).

3.3. Controllable trust

If we want to control trust we have to conclude that

1. trust in an infrastructure is given due to the fact that it is under full control of either the sender or the recipient.
2. trust in an infrastructure may not be necessary as the infrastructure is ideally unable to missuse data passed through it.

In this thesis we work with both cases. We will however never trust a third party apart from sender and recipient (no even their providers of infrastructures).

3.4. Reliable

Any message sending protocol needs to be reliable in its functionality. If means of message transport are unreliable users tend to use different means for communication.

3.5. Diagnosable

Reliability is somehow linked to transparent behaviour. This due to the fact that if something is generating a constant behaviour but we are unable to determine the reason for it (i.e. if we are expecting a different behaviour) we normally assume a malfunction. Therefore "Reliable" means not only stable by its behaviour. It means also diagnosable. User perception will not be "Reliable" if he is not able to determine causes for differences in observed and expected behaviour.

3.6. Available

Availability has two meanings in this context which do differ. A technology is available if. . .

1. a sender and a recipient have (or may have) the means of using it.
2. the infrastructure is providing the service (as oposed to: "is running in a degraded/faulty state unable to provide a service).

The first meaning tells us that a protocol must run independently from infrastructure the user has commonly ready.

The second meaning tells us that messages must always be capable of flowing from the sender to the recipient. As infrastructure may fail at any time the protocol must offer the possibilities to send messages through alternate routes. This sounds easy and many protocols implement such redundancies already. However – taking into account that sender and recipient is not known to a routing node this is a goal hard to achieve.

3.7. Rough draft of Protocol Layers

In order to fulfil the criteria given above we define a very rough idea of the protocol and its layers. The rough overview is given in figure 3.2. The protocol should work on the base of onion routing. Unlike Tor (see 5.10.2.8) it should not rely on central infrastructures. it furthermore should not be limited to onionize messages. It should be capable of splitting and reassemble messages at any layer

The protocol itself should send messages through a well known transport protocol ("Transport") which will be used for our messages.

The messages will be hidden within regular messages already using that transport layer. In an ideal implementation messages sent by our protocol should be indistinguishable from regular messages (by computers and humans). In order to achieve this particular task we introduce a "Hiding / Blending" layer. This layer is normally specific to the underlying transport layer and may vary.

The "Routing" layer is the layer that receives and sends messages. It parses only the core protocol and the data processed here is completely independent from the underlying transport layer.

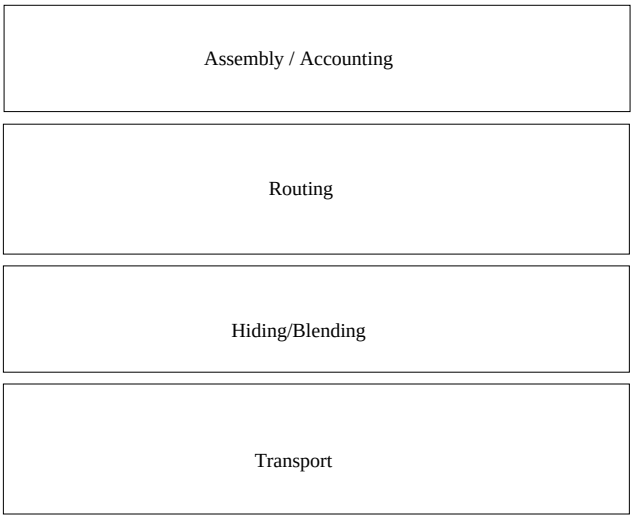

Figure 3.2.: A first rough overview for the protocol

The “Assembly / Accounting” layer disassembles the received messages into its parts. It then recomposes and stores the messages for further routing. In order to avoid spaming over this media some kind of accounting will be involved.

4. Existing Transport Layer Protocols

In this chapter we have a look at various available transport layer protocol. The main goal is to identify strong candidates as a transport layer. Main focus lies on the following criterias:

- Widely adopted (Ct1)
  The more widely adopted and used a protocol is the harder it is for an adverser to monitor (due to the sheer mass), filter or block the protocol (censorship resistance).
- Reliable (Ct2)
  Message transport between peers should be reliable. As messages may arrive anytime from everywhere we don not have means to synchronize the peer partners on a higher level without investing a considerable effort. In order to avoid this effort we do look for inherently reliable protocols.
- Symmetrical built (Ct3)
  The transport layer should be built on a peer to peer base. All servers implement a generic routing which requires no prior knowledge of possible targets. This criteria neglects centralistic infrastructures.

4.1. HTTP

The HTTP protocol allows message transfer from and to a server and is specified in RFC2616 [13]. It is not suitable as a communication protocol for messages due to the lack of notifications. There are some extensions which would allow such communications (such as WebDAV) but in general even those are not suitable as they require a continous connection to the server in order to get notifications. Having a “rollup” of notifications when connecting is not there by default but could be implemented on top of it.

HTTP servers listen on standard ports 80 or 443 for incomming connects. The port 443 is equivalent to the port 80 except for the fact that it has a wrapping encryption layer (usually TLS). The incomming connects (requests) must offer a header part and may contain a body part which would be suitable for transferring messages to the server. The reply onto this request is transferred over the same TCP connection containing the same two sections.

HTTP0.9-HTTP/1.1 are clear text protocols which are human readable (except for the data part which might contain binary data). The HTTP/2 (as specified in [4]) is using the same ports and default behaviour. Unlike HTTP/0.9-HTTP/1.1 it is not a clear text but a binary protocol. Headers and bodies of messages are sent as binaries.

The protocol does definitely satisfy the first two main criteria (Ct1: Widely Adopted and Ct2: Reliable).

The main disadvantage in terms as message transport protocol is that this protocol is not symetrically. This means that a server is always just “serving requests” and not sending actively information to peers. This Request-Reply violates criteria (Ct1: Symmetrically built) and makes the protocol not a primary choice for message transport.

4.2. MQTT

MQTT is an ISO standard (ISO/IEC PRF 20922:2016) and was formerly called MQ Telemetry Transport. The current standard as the time of writing tis document was 3.1.1 [3].

The protocol runs by default on the two ports 1883 and 8883 and may be encrypted with TLS. MQTT is a publish/subscribe based message passing protocol which is mainly targeted to m2m communication. This Protocol requires the receiving party to be subscribed in a central infrastructure in order to be able to receive messages. This makes it very hard to be used in a system without centralistic infrastructure and having no static routes between senders and recipients.

The protocol does definitely satisfy the second criteria (Ct2: Reliable). It is in the area of enduser (i.e. Internet) not widely adopted thus violating Criteria 1 (Ct1: Widely Adopted). In terms of decentralistic design the protocol fails as well (Ct3: Symmetrically built).

4.3. Advanced Message Queuing Protocol

The Advanced Message Queuing Protocol (AMQP) was originally initiated by numerous exponents based mainly in finance related industries. The AMQP-Protocol is either used for communication between two message brokers, or between a message broker and a client[2].

It is designed to be interoperable, stable, reliable and safe. It supports either SASL or TLS secured communication. The usage of such a tunnel is controlled by the immediate sender of a message. In its current version 1.0 it does however not support a dynamic routing between brokers[2].

Due to the lack of a generic routing capability this protocol is therefore not suitable for message transport in a global generic environment.

The protocol satisfys partially the first criteria (Ct1: Widely Adopted), and fully satisfies the second criteria (Ct2: Reliable). However the third criteria is violated due to the lack of routing capabilities between message brokers (Ct3: Symmetrically built).

4.4. Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) is a communication Protocol which is primarily destined to m2m communication. It is defined in RFC7252[5]. It is Defined as lightweight replacement for HTTP in IoT devices and is based on UDP.

The protocol does partially satisfy the first criteria (Ct1: Widely Adopted). The second criteria (Ct2: Reliable) is only partially fulfilled as it is based on UDP and does only add limited session control on its own.

The main disadvantage in terms as message transport protocol is that this protocol is not (like HTTP) symmetrically. This means that a server is always just “serving requests” and not sending actively information to peers. This Request-Reply violates criteria (Ct3: Symmetrically built) and makes the protocol not a primary choice for message transport.

4.5. Web Application Messaging Protocol

WAMP is a websockets based protocol destined to enable M2M communication. Like MQTT it is publish/-subscribe oriented. Unlike MQTT it allows remote procedure calls (RPC).

The WAMP protocol is not widely adopted (Ct1: Widely Adopted) but it is definitely reliable on a per node base (Ct2: Reliable). Due to its RPC based capability unlike MQTT a routing like capability could be implemented. Symmetrical protocol behaviour is therefore not available but could be built in relatively easy.

4.6. XMPP (jabber)

XMPP (originally named Jabber) is a synchronous message protocol used in the internet. It is specified in the documents RFC6120[21], RFC6121[21], RFC3922[20], and RFC3923[19]. The protocol is a very advanced chat protocol featuring numeros levels of security including end-to-end signing and object encryption[19]. There is also a stream initiation extension for transferring files between endpoints [24].

It has generic routing capabilities spanning between known and unknown servers.

The protocol itself seems to be a strong candidate as a transport layer as it is beeing used actively in the internet.

4.7. SMTP

The SMTP protocol is currently specified in [11]. 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 but this feature is not widely adopted). It is split into two parts. These parts are the header (which does 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 [17] and is separated with a blank line from the body.

It [11] furthermore introduces a simplistic model for mail communication. A more comprehensive model is introduced in the section **??**. As the proposed model is not sufficient for a comprehensive end-to-end analysis.

Traditionally the message itself is mime encoded. The MIME messages are mainly specified in [9], and [10]. MIME allows to send messages in multiple representations (alternates), and attach additional information (such as possibly inlined images or attached documents).

SMTP is one of the most common messaging protocols in the internet (Ct1: Widely Adopted) and it would be devastating for business of a country if for censoring reasons this protocol would be cut off. The protocol is furthermore very reliable as it has built in support for redundancy and a throughout message design making it relatively easy to diagnose problems (Ct2: Reliable). All SMTP servers are normally capable of routing and and receiving messages. Messages going over serveral servers are common (Ct3: Symmet-

rically built) so the third criteria may be considered as fulfilled as well

SMTP is considered a strong candidate as transport layer.

4.8. SMS

SMS capability was introduced in the SS7 protocol. This protocol allows the message transfer of messages not bigger than 144 character. Due to this restriction in size it is unlikely to be suitable for this type of communication as the keys beeing required are already sized similarly leaving no space for Messages or routing information.

Secondly the protocol is not widely adopted within the internet domain. There are gateways providing bridging functionalities to the SMS service. However – the protocol itself is insignificant in the internet itself.

4.9. MMS

The Multimedia Messaging Service (MMS) is maintained by 3GPP (3rd Generation Partnership Project). This protocol is manly a mobile protocl based on telephone networks.

4.10. Roundup for Transport Protocols

In table 4.1 we sum up all previously analysed protocols. We use "✓" for a fulfilled criteria, "~" for a partially fulfilled criteria, and "×" for a not fulfilled criteria. This overview shows in compact form which protocols have been identified as strong candidates for use as a transport layer in terms of an anonymising protocol.

This table shows that strong identified candidates are SMTP (being already a message sending protocol on asynchronous base) and XMPP (a real time chat protocol able to attach files. This protocol features furthermore end-to-end encryption and signing). Both have the advantages that they are really wide adopted in the internet and do support additionally advanced content (such as alternatives or attachments).

[1] omitted due to message size being too small

Criteria / Protocol	Ct1: Widely adopted	Ct2: Reliable	Ct3: Symmetrically built
HTTP	✓	✓	×
MQTT	~	✓	×
AMQP	~	✓	×
CoAP	~	~	×
WAMP	×	✓	~
XMPP	✓	✓	✓
SMTP	✓	✓	✓
SMS[1]			
MMS	✓	✓	×

Table 4.1.: comparison of protocols in terms of the suitability criteria as transport layer

5. Existing Research and Implementations on the Topic

5.1. Anonymity

As Anonymity we take the definition as specified in [15].

> Anonymity of a subject means that the subject is not identifiable within a set of subjects, the anonymity set.[1]

and

> Anonymity of a subject from an attacker's perspective means that the attacker cannot sufficiently identify the subject within a set of subjects, the anonymity set.[1]

Whereas the anonymity set is defined as the set of all possible subjects.

Especially the anonymity of a subject from an attacker's is very important to this paper.

5.1.1. k-Anonymity

k-anonymity is a term introduced in [1]. This work claims that no one might be held responsible for an action if the action itself can only be identified as an action which has been taken by one unidentifiable entity out of k entities.

The Document distinguishes between *Sender k-anonymity* where the sending entity can only be narrowed down to a set of k entities and *Receiver k-anonymity*

5.1.2. ℓ-Diversity

In [12] an extended model of k-anonymity. According to the authors it is possible to break a k-anonymity set if there is additional Information available which may be merged into a data set so that a special entity can be filtered from the k-anonymity set. In other words if an anonymity set is to tightly specified a single additional background information might be sufficient to identify a specific entity in an anonymity set.

While it might be arguable that a k-anonymity in which a member is not implicitely k-anonymous still is sufficient for k-anonymity in its sense the point made in this work is definitely right and should be taken into account.

Their approach is to introduce an amount of invisible diversity into k-anonymous sets so that simple background knowledge is no longer sufficient to isolate a single member.

5.1.3. t-Closeness

While ℓ-diversity protects the identity of an entity it does not prevent information gain. A subject which is in a class has the same attributes. This is where t-closeness[14] comes into play. t-closeness is defined as follows:

> An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold. A table is said to have t-closeness if all equivalence classes have t-closeness.

5.2. Zero Trust

Zero trust is not a truly researched model in systems engineering. It is however widely adopted.

We refer in this work to the zero trust model when denying the trust in any infrastructure not directly controlled by the sending or receiving entity. This distrust extends especially but not exclusively to the network transporting the message, the nodes storing and forwarding messages, the backup taken from any system except the client machines of the sending and receiving parties, and software, hardware, and operators of all systems not explicitly trusted.

As explicitely trusted in our model we do regard the user sending a message (and his immediate hardware used for sending the message), and the users receiving the messages. Trust in between the receiving parties (if more than one) of a message is not necessarily given.

[1] footnotes omitted in quote

5.3. Pseudonymity

As Pseudonymity we take the definition as specified in [15].

> A pseudonym is an identifier of a subject other than one of the subject's real names. The subject which the pseudonym refers to is the holder of the pseudonym. A subject is pseudonymous if a pseudonym is used as identifier instead of one of its real names.[2]

5.4. Undetectability

As undetectability we take the definition as specified in [15].

> Undetectability of an item of interest (IOI) from an attacker's perspective means that the attacker cannot sufficiently distinguish whether it exists or not.[2]

5.5. Unobservability

As unobservability we take the definition as specified in [15].

> Unobservability of an item of interest (IOI) means
>
> - undetectability of the IOI against all subjects uninvolved in it and
> - anonymity of the subject(s) involved in the IOI even against the other subject(s) involved in that IOI.

This part is very important. As we are heading for a censorship resistant solution unobservability is a key. As mentioned in this paper unobservability raises the bar of required attributes again (⇒ reads "implies"):

$$\begin{aligned} censorship\,resistance &\Rightarrow unobservability \\ unobserability &\Rightarrow undetectability \\ unobserability &\Rightarrow anonymity \end{aligned}$$

So this means that we have to use an undetectable data channel in order to achieve censorship resistance.

5.5.1. Ephemeral Identity

In this work we use accounting on various levels. While we are dealing with anonymity accounting has still to be linked to some kind of identity for this reason we are introducing a term called "ephemeral identity".

> A Ephemeral identity is a temporary identity which is defined by the following attributes:
>
> - It is an artificial identity
> - It is only used for a short timespan
> - It is not linkable to another identity

The key in this definition is the last point is crucial and at the same time hard to achieve.

5.6. Single Use Reply Blocks and Multi Use Reply Blocks

The use of single use reply blocks were first introduced by Chaum in [6]. The concept is that we have in our case a routing block which might be used up to n times ($0 < n < FIXME$)

5.7. Censorship

5.7.1. Censorship Resistant

5.7.2. Parrot Circumvention

5.7.3. Censorship Circumvention

5.7.3.1. Covert Channel

5.7.3.2. Spread Spectrum

5.8. Cryptography

5.8.1. Symmetric Encryption

5.8.2. Asymmetric Encryption

5.8.2.1. RSA

[18]

[2] footnotes omitted in quote

5.8.2.2. Elliptic Curve Cryptogaphy

5.8.3. Homomorphic encryption

5.9. Routing

5.9.1. Mixing

5.9.2. Onion Routing

5.9.3. Crowds

5.9.4. Mimic routes

5.9.5. Cryptopuzzles

5.10. System Implementations

The following sections describe

5.10.1. Concepts

5.10.1.1. DC Networks

5.10.1.2. MIX Networks

5.10.1.3. Onion Routing

5.10.1.4. Remailer

5.10.2. Implementations

The following sections emphasize on implementations of anonymising (and related) protocols regardless of their usage in the domain of messaging. It is a list of system classes or their specific implementations together with a short analysis of strength and weaknesses. Wherever possible we try to refer to original sources. This is however not always possible since some of these systems are no longer in use.

5.10.2.1. Pseudonymous Remailer

5.10.2.2. Babel

5.10.2.3. Cypherpunk-Remailer

5.10.2.4. Mixmaster-Remailer

5.10.2.5. Mixminion-Remailer

5.10.2.6. Crowds

5.10.2.7. Tarzan

5.10.2.8. TOR

5.10.2.9. Herbivore

5.10.2.10. Dissent

5.10.2.11. P5

5.10.2.12. Gnutella

5.10.2.13. Gnutella2

5.10.2.14. Freenet

5.10.2.15. Darknet

5.10.2.16. Sneakernet

5.10.2.17. Hordes

5.10.2.18. Salsa

5.10.2.19. Hydra-Onion

5.11. Known Attacks

In the following sections we emphasize on possible attacks to an anonymity preserving protocols. In the following sections we describe classes of attacks. These attacks may be used to attack the anonymity of any entity involved in the message channel. In a later stage we test the protocol for immunity against these classes of attacks.

5.11.1. Broken Encryption Algorithms

5.11.2. Attacks Targeting Anonymity

5.11.2.1. Hotspot Attacks

5.11.2.2. Message Tagging and Tracing

5.11.2.3. Side Channel Attacks

5.11.2.4. Timing Attacks

5.11.2.5. Sizing Attacks

5.11.2.6. Bugging Attacks

5.11.3. Denial of Service Attacks

5.11.3.1. Censorship

Where as traditional censorship is widely regarded as selective information filtering and alteration a very repressive censorship can even include denial of information flows in general. Any anonymity system not offering the possibility to hide in legitimate information flows is therefore not censorship resistant.

5.11.3.2. Credibility Attack

Another type of DoS attack is the credibility attack. While not necessarily a technical attack it is very effective. A system not having a sufficiently big user base is offering thus a bad level of anonymity due to the fact that the anonymity set is too small or the traffic concealing message flow is insufficient.

In a credibility attack a systems reputation is degraded in such a way that the system is no longer used. This may be achieved in serveral ways. This is usually done by reducing the reputation of a system.

This may be achieved in several ways. Examples:

- Disrupt functionality of a system.
- Publicly dispute the effectiveness of a system.
- Reduce the effectiveness of a system.
- Dispute the credibility of the system founders.
- Dispute the credibility of the infrastructure.

6. Applied Methodes

Based in the findings in this chapter we used the following methodology:

1. Identify problem hotspots for a new protocol.
2. Design a protocol which addresses the previously identified hotspots.
3. Build a protocol prototype.
4. Analyse the protocol for weaknesses using attack schemes.
    a) Tagging/Bugging attacks
    b) Tracing attacks

6.1. Problem Hotspots

6.2. Protocol Design

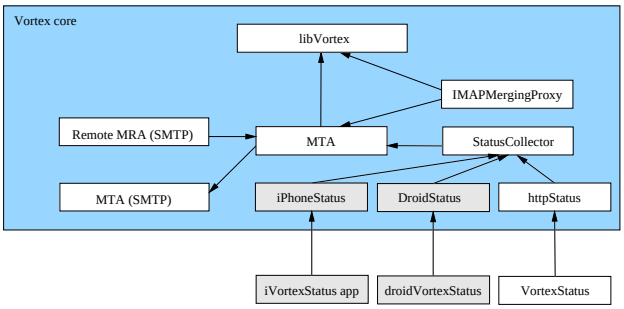

Figure 6.1.: Overview of the Vortex modules

6.3. Protocol implementation

6.4. Protocol Analysis

Part II.

Results

To verify the hypothesis made in this paper, and to analyse properties of the protocol in a real world scenario a library was implemented in Java which was capable of handling all message packets and the routing stack as a whole. The following paragraphs describe the protocol developed in general as a generic approach. Appendix IV gives the full ASN.1 representation of the protocol.

It is important to notice that ASN.1 has no mean to express encrypted structures. Due to this fact we defined all encrypted fields as `OCTET STRING`. The protocol offers according to the ASN.1 the possibility to store onionized information in an unencrypted form. This is meant for debuing purposes. At no point this should be used in a production environment.

The protocol described in the next chapter is independent from routing. At the moment capabilities include SMTP and XMPP. The protocol may be extended by adding new transport layer capabilities and their addressing schemes.

7. MessageVortex - Transport Independent Messaging anonymous to 3rd Parties

7.1. Protocol Description

7.2. Accounting

7.3. Message Flows

7.4. Considerations for Building Messages

7.4.1. Timing of messages

7.4.2. Building Diagnostic Paths

7.4.2.1. Implicit Diagnostic

7.4.2.2. Automatic Explicit Diagnostic

7.4.2.3. On-Demand Explicit Diagnostic

7.5. Real World Considerations

This approach is heavily dependent of the transport protocol and builds on top a new obfuscating/routing layer. For this system to become a real peer-to-peer approach some additional quirks are required. A message-Vortex-Account needs always an active routing handler. This routing handler may be introduced by new server capabilities or by having a device handling the routing from the client side. For this reason we built a RaspberryPi appliance capable of connecting to one (or more) accounts fetching incomming mails, analysing them and reroute them if necessary. Although the system is designed to be run on a RaspberryPi the software might be installed to any Java capable client. The RaspberryPi is just an affordable lightweight device which offers all required capabilities.

8. Security Analysis

9. Additional Considerations

9.1. Storage of Messages and queues

The storage of messages sent though MessageVortex should be handled with great care. It seems on the first sight a good idea to merge all messages in a globally available storage such as the mail account of the receiving entity. However – In doing so we would discover the message content to the providing party of a mail account. Since we handled the message with great care and tremendous costs up until this point it would be careless doing so.

Storing them in a localized and receiving entity controlled storage is definitely a good idea but leaves security considerations like a backup possibly to an end user. This might be better but in effect a questionable decision. There is however a third option. By leaving the message unhandled on the last entity of the MessageVortex chain we may safely backup the data without disclosing the message content. Merging the content then dynamically through a specialized proxy would allow the user tu have a unified view on his without compromising the security.

9.2. Economy of transfer

Part III.

Discussion

10. Anonymity

10.1. Effects of anonymous communication on behaveour

[16]

Part IV.

Appendix

ASN.1 representation of the protocol

Listing 1: ASN.1 representation of the protocol

```
-- FIXME ADAPT NeoScrypt

-- encryption: as specified in the key. If not specified default mode is ECB and default padding is PKCS1Padding

-- States: Tuple()=Value() [vallidity; allowed operations] {Store}
-- - Tuple(identity)=Value(messageQuota,transferQuota,sequence of Routingblocks for Error Message Routing) [validity; Requested at creation; may be extended upon request] {identityS
-- - Tuple(Identity,Serial)=maxReplays ['valid' from Identity Block; from First Identity Block; may only be reduced] {IdentityReplayStore}

Message-Blocks DEFINITIONS EXPLICIT TAGS ::=
BEGIN

  -- define constants
  maxSerial             INTEGER ::= 4294967295      -- maximum serial number
  maxChunkSize          INTEGER ::= 4294967295      -- maximum size of a message chunk
  maxNumberOfReplays    INTEGER ::= 65535           -- maximum number of replys
  maxNumberOfRequests   INTEGER ::= 8               -- maximum number of administrative requests

  Message      ::= SEQUENCE {
    header          IdentityBlock,
    blocks          CHOICE {
        encrypted [1101] OCTET STRING,      -- contains encrypted UnencryptedBlocks structure; Decryption key is in identity block [decryptionKey]
        -- it is not allowed to use plain in production environments; This is for testing and analysis only
        plain     [1102] UnencryptedBlocks -- should not be used except for internal diagnostic purposes
    }
  }

  -- represents the identity of the rights owner
  IdentityBlock ::= SEQUENCE {
    headerKey             [1000] OCTET STRING OPTIONAL,  -- contains SymmetricKey encrypted with recipient nodes public key
    identityBlock         CHOICE {
        encrypted  [1001] OCTET STRING,
        -- it is not allowed to use plain in production environments; This is for testing and analysis only
        plain      [1002] IdentityPayloadBlock,
    },
    identitySignature OCTET STRING   -- contains signature of Identity [as stored in identityBlock;signed identityBlock without Tag]
  }

  IdentityPayloadBlock ::= SEQUENCE {
    -- Public key of the identity representing this transmission
    identityKey         AsymmetricKey,

    -- serial identifying this block
    serial              INTEGER (0..maxSerial),

    -- number of times this block may be replayed (Tuple is identityKey,serial while
    maxReplays          INTEGER (0..maxNumberOfReplays),

    -- subsequent Blocks are not processed before valid time.
    -- Host may reject too long retention. Recomended validity support >=1Mt.
    valid               UsagePeriod,

    -- represents the chained secret which has to be found in subsequent blocks
    -- prevents reassembly attack
    forwardSecret [2000] ChainSecret OPTIONAL,

    -- contains SymmetricKey encrypted with private key of identityKey
    -- encryption is done as proof of identity (identity hijack protection)
    decryptionKey       OCTET STRING,   -- contains (encrypted) DER encoded ASN1BitString with key representation

    --  contains the MAC-Algorithm used for signing
    hash                MacAlgorithm,

    -- contains administrative requests such as quota requests
    requests            SEQUENCE (SIZE (0..maxNumberOfRequests)) OF HeaderRequest ,

    -- padding and identitifier required to solve the cryptopuzzle
    identifier      [2001] INTEGER (0..maxSerial) OPTIONAL,
    padding         [2002] OCTET STRING OPTIONAL  -- This is for solving crypto puzzles
  }

  UnencryptedBlocks ::= SEQUENCE {
    -- contains routing information (next hop) for the payloads
    -- FIXME how handle multiple payloads
    routing       [3000] RoutingBlock OPTIONAL,

    -- contains encrypted log data of the data traveling
    routingLog    [3010] RoutingLogBlock OPTIONAL,

    -- contains replys to header requests (eg. quota and identity handling)
    reply         [3020] ReplyBlock OPTIONAL,

    -- contains the actual payload
    payload       [3100] SEQUENCE (SIZE (0..128)) OF PayloadChunk
  }


  -- represents the building and sending process for the next hop
  RoutingBlock ::= SEQUENCE {

    -- contains the next recipient in sequence
    recipient NodeSpec,

    -- contains the period when the payload should be processed
    -- Router might refuse to long queue retention
    -- Recommended support for retention >=1h
    queueTime UsagePeriod,

    nextHop             SEQUENCE (SIZE (0..128)) OF NextHopBlock,   -- encrypted next RoutingBlocks for the payload

    -- contains the secret of the identity block (if any)
    forwardSecret       [111]   ChainSecret OPTIONAL,
```

```
103     -- contains a routing block which may be used when sending error messages back to the quota owner
104     -- this routing block may be cached for future use
105     replyBlock [131]    RoutingBlock OPTIONAL,
106
107     -- This section is required if payload is routed with a prebuilt RB (
108     -- Messages MAY always request recompression (otherwise the message is identifyable from a non reply routing block)
109     decryptionKey       [200] SEQUENCE (SIZE (1..2)) OF SymmetricKey OPTIONAL,
110     encryptionKey       [201] SymmetricKey OPTIONAL,
111
112     -- contains information for building replys
113     cascade             [300] SEQUENCE(SIZE (0..255)) OF CascadeBuildInformation
114   }
115
116   NextHopBlock ::= SEQUENCE {
117     nextIdentityBlock    [13100] OCTET STRING,
118     nextRoutingBlock     [13200] OCTET STRING,
119     nextReplyBlock       [13300] OCTET STRING OPTIONAL,
120     nextErrorReplyBlock [13400] OCTET STRING OPTIONAL
121   }
122
123   PayloadOperation ::= SEQUENCE{
124     operation            PayloadType,
125     thisid               [12010] SEQUENCE (0..128) OF INTEGER (0..65535)
126   }
127
128   PayloadType ::= CHOICE {
129         randomPayload    [100] RandomPayloadOperation,
130         splitPayload     [150] SplitPayloadOperation,
131         mergePayload     [200] MergePayloadOperation,
132         xorPayload       [250] XorPayloadOperation,
133         encryptPayload   [300] EncryptPayloadOperation,
134                 ...
135   }
136
137   CascadeBuildInformation ::= SEQUENCE {
138     encryptionKey        SymmetricKey,
139     secret               ChainSecret,
140     payloadOp            PayloadOperation,
141     ...
142   }
143
144   PercentSizeType ::= SEQUENCE {
145     fromPercent          REAL (0..100),
146     toPercent            REAL (0..100)
147   }
148
149   AbsoluteSizeType ::= SEQUENCE {
150     fromAbsolute         INTEGER (0..maxChunkSize),
151     toAbsolut            INTEGER (0..maxChunkSize)
152   }
153
154   SizeType ::= SEQUENCE{
155         reference               INTEGER (0..65535),
156         size CHOICE {
157                 percent     [15001] PercentSizeType,
158                 absolute    [15101] AbsoluteSizeType
159         }
160   }
161
162   RandomPayloadOperation ::= SEQUENCE {
163     size                 SizeType
164   }
165
166   SplitPayloadOperation ::= SEQUENCE {
167     originalId                          INTEGER (0..65535),
168     firstSize                SizeType,
169         secondId                                INTEGER (0..65535)
170   }
171
172   MergePayloadOperation ::= SEQUENCE {
173     -- FIXME
174   }
175
176   XorPayloadOperation ::= SEQUENCE {
177     -- FIXME
178   }
179
180   EncryptedRoutingLogBlock ::= OCTET STRING -- contains symmetrically encrypted RoutingLogBlock
181   RoutingLogBlock ::= SEQUENCE {
182     routingLog            SEQUENCE (SIZE (0..16)) OF RoutingLog,
183     nestedRoutingInformationBlock EncryptedRoutingLogBlock
184   }
185
186   RoutingLog ::=SEQUENCE    {
187     nodeIdentifier      IA5String,
188     time                GeneralizedTime,
189     code                ErrorCode,
190     information         IA5String
191   }
192
193   IdentityReplayStore ::= SEQUENCE {
194         replays          SEQUENCE (SIZE (0..4294967295)) OF IdentityReplayBlock
195   }
196
197   IdentityReplayBlock ::= SEQUENCE {
198     identity             AsymmetricKey,
199     valid                UsagePeriod,
200         replaysRemaining         INTEGER (0..4294967295)
201   }
202
203   IdentityStore ::= SEQUENCE {
204     identities SEQUENCE (SIZE (0..4294967295)) OF IdentityStoreBlock
205   }
206
207   IdentityStoreBlock ::= SEQUENCE {
208     valid                UsagePeriod,
209     messageQuota         INTEGER (0..4294967295),
210     transferQuota        INTEGER (0..4294967295),
211     identity             [1001] AsymmetricKey OPTIONAL,     -- if omitted this is a node identity
212     nodeAddress          [1002] NodeSpec      OPTIONAL,     -- if ommited own identity key
213     nodeKey              [1003] SEQUENCE OF AsymmetricKey OPTIONAL,     -- Contains the identity of the owning node; May be ommited if local node
214     routingBlocks        [1004] SEQUENCE OF RoutingBlock OTIONAL
215     ...
216   }
217
218   -- contains a node spec of a routing point
219   -- At the moment either smtp:<email> or xmpp:<jabber>
220   NodeSpec ::= IA5String
221
222   ChainSecret ::= INTEGER (0..4294967295)
223
```

```
224  -- FIXME define requests
225  HeaderRequest ::= CHOICE {
226    identity       [0] HeaderRequestIdentity,
227    capabilities   [1] HeaderRequestCapability,
228    messageQuota   [2] HeaderRequestIncreaseMessageQuota,
229    transferQuota  [3] RequestIncreaseTransferQuota,
230    quotaQuery     [4] HeaderRequestQueryQuota,
231    ...
232  }
233
234  ReplyBlock ::= CHOICE {
235    identity       [0] ReplyIdentity,
236    capabilities   [1] ReplyCapability,
237    ...
238  }
239
240  HeaderRequestIdentity ::= SEQUENCE {
241    identity AsymmetricKey,
242    period UsagePeriod,
243    ...
244  }
245
246  ReplyIdentity ::= SEQUENCE {
247    challenge      BIT STRING, -- bit sequence at beginning of hash from encrypted identity block
248    hash           MacAlgorithmIdentifier,
249    valid          UsagePeriod,
250    identifier     INTEGER (0..4294967295),
251    ...
252  }
253
254  HeaderRequestQueryQuota ::= SEQUENCE {
255    identity AsymmetricKey,
256    ...
257  }
258
259  HeaderRequestIncreaseMessageQuota ::= SEQUENCE {
260    identity AsymmetricKey,
261    messages INTEGER (0..4294967295),
262    ...
263  }
264
265  RequestIncreaseTransferQuota ::= SEQUENCE {
266    identity AsymmetricKey,
267    size     INTEGER (0..4294967295),
268    ...
269  }
270
271  HeaderRequestCapability ::= SEQUENCE {
272    period UsagePeriod,
273    ...
274  }
275
276  ReplyCapability ::= SEQUENCE {
277    cypher            SEQUENCE (SIZE (2..256)) OF CypherSpec,
278    maxTransferQuota  INTEGER (0..4294967295),
279    maxMessageQuota   INTEGER (0..4294967295),
280    supportedProtocol SEQUENCE OF Protocol,
281    ...
282  }
283
284  CypherSpec ::= SEQUENCE {
285      asymmetric AsymmetricAlgorithmIdentifier,
286      symmetric  SymmetricAlgorithmIdentifier,
287      mac        MacAlgorithmIdentifier
288  }
289
290  Protocol ::= ENUMERATED {
291    smtp  (100),
292    xmmp  (110),
293    ...
294  }
295
296  ErrorCode ::= ENUMERATED {
297
298    -- System messages
299    ok                    (2001),
300    transferQuotaStatus   (2101),
301    messageQuotaStatus    (2102),
302
303    -- protocol usage failures
304    transferQuotaExceeded (3001),
305    messageQuotaExceeded  (3002),
306    identityUnknown       (3101),
307    messageChunkMissing   (3201),
308    messageLifeExpired    (3202),
309
310    -- Mayor host specific errors
311    hostError             (5001),
312    ...
313  }
314
315  PayloadChunk ::= SEQUENCE {
316    offset                 INTEGER (0..MAX),
317    routingBlockIdentifier [0]  SEQUENCE (SIZE (0..255)) OF INTEGER, -- FIXME limit integer range
318    payload                [100] OCTET STRING
319  }
320
321  -- Compatible to PrivateKeyUsagePeriod taken from RFC3280
322  UsagePeriod ::= SEQUENCE {
323    notBefore       [0]    GeneralizedTime OPTIONAL,
324    notAfter        [1]    GeneralizedTime OPTIONAL
325  }
326
327  -- adapted from RFC3280
328   SymmetricAlgorithmIdentifier  ::=  SEQUENCE  {
329     algorithm                 SymmetricAlgorithm,
330     parameter                 AlgorithmParameters OPTIONAL
331   }
332
333   AsymmetricAlgorithmIdentifier  ::=  SEQUENCE  {
334     algorithm                 AsymmetricAlgorithm,
335     parameter                 AlgorithmParameters OPTIONAL
336   }
337
338   MacAlgorithmIdentifier  ::=  SEQUENCE  {
339     algorithm                 MacAlgorithm,
340     parameter                 AlgorithmParameters
341   }
342
343
344   SymmetricAlgorithm ::= ENUMERATED {
```

```
345     aes128       (1000),
346     aes192       (1001),  -- optional support
347     aes256       (1002),
348     camellia128 (1100),
349     camellia192 (1101),  -- optional support
350     camellia256 (1102)
351   }
352
353   AsymmetricAlgorithm ::= ENUMERATED {
354     rsa        (2000),
355     dsa        (2100),
356     secp384r1 (2500),
357     sect409k1 (2501),
358     secp521r1 (2502)
359   }
360
361   MacAlgorithm ::= ENUMERATED {
362     sha384    (3000),
363     sha512    (3001),
364     -- FIXME check AEAD
365     tiger192 (3100)
366   }
367
368   ECCurveType ::= ENUMERATED{
369     secp192r1,
370     sect163k1,
371     sect163r2,
372     secp224r1,
373     sect233k1,
374     sect233r1,
375     secp256r1,
376     sect283k1,
377     sect283r1,
378     secp384r1,
379     sect409k1,
380     sect409r1,
381     secp521r1,
382     sect571k1,
383     sect571r1,
384     ...
385   }
386
387   AlgorithmParameters ::= SEQUENCE {
388     keySize    [10000] INTEGER (0..65535) OPTIONAL,
389     curveType [10001] ECCurveType OPTIONAL,
390     ...
391   }
392
393   -- Symmetric key
394   SymmetricKey ::= SEQUENCE {
395     keyType SymmetricAlgorithmIdentifier,
396     key       OCTET STRING
397   }
398
399   -- Asymmetric Key
400   AsymmetricKey ::= SEQUENCE {
401     keyType         AsymmetricAlgorithmIdentifier,
402     publicKey       [1] OCTET STRING,
403     privateKey      [2] OCTET STRING OPTIONAL
404   }
405
406   pkcs-1     OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }
407
408 END
```

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 [7]. 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.

Item of Interest (IoI) FIXME

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.

”[22, 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 [11].

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. 17).

[2] *AMQP v1.0*. AMQP.org, 2011. URL: http://www.amqp.org/confluence/display/AMQP/AMQP+Specification (cit. on p. 13).

[3] Banks Andrew and Gupta Rahul. *MQTT*. en. OASIS, 2014. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf (cit. on p. 13).

[4] Mike Belshe, Roberto Peon, and Martin Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540. May 2015. DOI: 10.17487/rfc7540. URL: https://rfc-editor.org/rfc/rfc7540.txt (cit. on p. 13).

[5] Carsten Bormann, Klaus Hartke, and Zach Shelby. *RFC7252: The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014. DOI: 10.17487/rfc7252. URL: https://rfc-editor.org/rfc/rfc7252.txt (cit. on p. 14).

[6] David Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms". In: *Communications of the ACM* 24.2 (Feb. 1981), pp. 65–75. URL: http://users.cis.fiu.edu/~carbunar/teaching/cis5372.2013.online/reading/chaum_untraceable.pdf (cit. on p. 18).

[7] M. Crispin. *RFC3501 INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*. IETF, 2003. URL: http://tools.ietf.org/pdf/rfc3501.pdf (cit. on p. 43).

[8] M. Elkins. *RFC2015 MIME Security with Pretty Good Privacy (PGP)*. IETF, 1996. URL: http://tools.ietf.org/pdf/rfc2015.pdf (cit. on p. 1).

[9] 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 pp. 1, 14).

[10] 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).

[11] J. Klensin. *RFC5321 Simple Mail Transfer Protocol*. IETF, 2008. URL: http://tools.ietf.org/pdf/rfc5321.pdf (cit. on pp. 14, 43).

[12] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "l-diversity: Privacy beyond k-anonymity". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 3. URL: http://www.cs.cornell.edu/~vmuthu/research/ldiversity.pdf (cit. on p. 17).

[13] Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. June 1999. DOI: 10.17487/rfc2616. URL: https://rfc-editor.org/rfc/rfc2616.txt (cit. on p. 13).

[14] Suresh Venkatasubramanian Ninghui Li Tiancheng Li. "t-Closeness: Privacy Beyond k-Anonymity and". In: (). URL: http://www.cs.purdue.edu/homes/li83/papers/icde_closeness.pdf (cit. on p. 17).

[15] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. v0.34. Aug. 2010. URL: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf (cit. on pp. 17, 18).

[16] Tom Postmes, Russell Spears, Khaled Sakhel, and Daphne De Groot. "Social influence in computer-mediated communication: The effects of anonymity on group behavior". In: *Personality and Social Psychology Bulletin* 27.10 (2001), pp. 1243–1254. DOI: 10.1177/01461672012710001. URL: http://psp.sagepub.com/content/27/10/1243.short (cit. on p. 35).

[17] P. Resnick. *RFC5322 Internet Message Format*. IETF, 2008. URL: http://tools.ietf.org/pdf/rfc5322.pdf (cit. on p. 14).

[18] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-key Cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342 (cit. on p. 18).

[19] J. P. Sain-Andre. *RFC3923: End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)*. IETF, 2004. URL: http://tools.ietf.org/pdf/rfc3923.pdf (cit. on p. 14).

[20] P. Saint-Andre. *RFC3922: Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)*. IETF, 2004. URL: http://tools.ietf.org/pdf/rfc3922.pdf (cit. on p. 14).

[21] P. Saint-Andre. *RFC6120: Extensible Messaging and Presence Protocol (XMPP): Core*. IETF, 2011. URL: http://tools.ietf.org/pdf/rfc6120.pdf (cit. on p. 14).

[22] A. Stevenson. *Oxford Dictionary of English*. Oxford reference online premium. OUP Oxford, 2010. ISBN: 9780199571123. URL: http://www.oed.com (cit. on p. 43).

[23] Almon Brown Strowger. “Automatic Telephone-Exchange”. en. Pat. 447918. Mar. 1891 (cit. on p. 1).

[24] Muldowney Thomas, Miller Mathew, Eatmon Ryan, and Saint-Andre Peter. *XEP-0096: SI File Transfer*. XMPP Standards Foundation, 2004. URL: http://xmpp.org/extensions/xep-0096.html (cit. on p. 14).

[25] UNHR. *International Covenant on Civil and Political Rights*. 1966. URL: http://www.ohchr.org/en/professionalinterest/pages/ccpr.aspx (cit. on p. 9).

Index