Freedom 2.0 Mail System

Roger McFarlane – roger@zeroknowledge.com
Adam Back – adamb@zeroknowledge.com
Graydon Hoare – graydon@zeroknowledge.com
Serge Chevarie-Pelletier – serge@zeroknowledge.com
Bill Heelan – bill@zeroknowledge.com
Christian Paquin – christian@zeroknowledge.com
Deniz Sarikaya – deniz@zeroknowledge.com
Zero-Knowledge Systems, Inc.

December 19, 2000

Abstract

This white paper, targeted at the technically knowledgeable user, looks at the entities, protocols, and systems that make up the Freedom Network Mail System.

More extensive documentation, containing implementation details with the intent of allowing and encouraging external cryptanalysis of the system will be available at some point in the future.

1 Introduction

1.1 This Paper

The 2.0 release of Freedom implements a completely new design of the mail system. It should not be thought of as an upgrade to the 1.0 Freedom mail system, but rather as a replacement of the previous design. This paper focuses on how the Freedom system handles email in all different cases, from both the server and client end, focusing more detail on the server-side aspects. From this point on, when we use the word "mail", we are actually referring to electronic mail, or "email". We do not cover security vulnerabilities extensively in this paper, and recommend the reader searching for such detailed coverage refer to Freedom 2.0 Security Issues and Analysis[1]. As there is a degree of overlap between various sections of this document, there is some redundancy in descriptions.

As of the 2.0 release of Freedom, there are two kinds of nym available: standard nyms and premium nyms. When a user purchases Freedom's "premium services", they receive an activation code that they exchange for tokens; these tokens are used to create premium nyms. Premium nyms have Freedom mail capabilities and full access to the Freedom Network for pseudonymous Internet privacy. Users who do not purchase Freedom "premium services" receive standard nyms, which do not have Freedom mail capabilities. For the scope of this paper, we assume the user has premium nyms.

1.2 Assumptions

We assume you have already read and understood the Freedom System 2.0 Architecture[2] document and understand terminology such as: nyms, public/private key pairs, and symmetric key pairs. We do not assume any knowledge of the Freedom 1.0 documents, as the 2.0 release is a redesign of the Freedom System, as opposed to an upgrade, especially with respect to Mail handling. Readers who are familiar with the Freedom 1.0 documents will better be able to appreciate the ways in which we have redesigned the mail system. Readers should be familiar with the basics of how POP mail works.

1.3 Freedom Mail Overview

As there were a number of usability, reliability, performance and security problems with the 1.0 system (the reply-block mail system), we have been designing a "next-generation" mail system. While we have some preliminary candidate designs, they will take some time to explore, analyze, and implement. The 2.0 release of Freedom comprises an implementation of a short-term fix: the POP mail system. The POP mail system primarily targets the usability and reliability aspects of the old mail system, and has the limited security objective of being no less secure, or at least of similar security to the old reply-block mail system.

To wit, Freedom users no longer need their own POP account in order to use Freedom mail. We provide a POP account for each nym, where mail is stored in encrypted format until the client retrieves it.

The objectives behind this POP mail system design were that it:

- Should be easy to implement, so that we could roll it out within a short timeframe.
- Should improve the usability experienced by customers.
- Should be highly scalable, allowing us to support a growing customer base.
- Should improve reliability of message delivery.

Pros

1. No reply-blocks.
2. No activation tokens.
3. Passive attacks seem to be harder.
4. Mail retrieval hidden inside other freedom traffic.
5. Lower bandwidth requirements.
6. We have control over the operational reliability of the system.
7. The system is conceptually simple and comprehensible.
8. Should result in faster delivery.
9. Should have greater operational reliability.
10. Immediate availability of mail services after nym creation.
11. Not vulnerable to legal attacks (i.e.: not subpoenable).
12. The user never gets encrypted mail that they have to manually decrypt.
13. The design is amenable to alternative user interfaces (web, access).

14. Mail is no longer stored in users' ISP accounts, removing the ISP as a potential attacker.
15. The design is amenable to quality of service payment options (e.g., user can buy more POP storage space for their nym).
16. We can leverage a great deal of existing, open, software to do most of the work.

Cons

1. Users must interact with the mail system separately for each nym in order to avoid giving Zero-Knowledge the power to correlate that their nyms may be owned by the same person.
2. Large infrastructure/storage requirements.
3. ZKS could learn who sent an email to whom and when.

2 Glossary

This glossary explains the terms and how we will use them throughout the rest of this document. We will go into further detail on each of our own entities and implementations in Section 4, Entities.

Nym

As of the 2.0 release of Freedom, there are two kinds of nym available. Users who install Freedom but do not purchase a serial number can use the Freedom client and its standard features free of charge. Standard features include several pseudonyms ("nyms") that do not have Freedom mail capabilities, nor do they use the Freedom Network for pseudonymous Internet communications. When a user purchases Freedom "premium services", they are given a serial number that they exchange for tokens, which in turn are used to create premium nyms. Premium nyms have Freedom mail capabilities and full access to the Freedom Network for pseudonymous Internet privacy.

For the purposes of this paper, we assume the user has premium nyms, as any other scenario is pointless in the context of the Mail system.

Cluster

A cluster is a group of physical machines that appears to a user to be a single machine. Examples in the Freedom Mail System are the IMEP, and IMEPB.

MTA – Mail Transfer Agent

Software that is responsible for the delivery of Internet mail from one site to another. sendmail, postfix, and qmail are all MTAs. If one draws an analogy to letter and parcel delivery in the "real world", an MTA is equivalent to your favorite postal or courier service; given a package, they take the responsibility for routing it from place to place such that the package eventually, you hope, reaches its addressee.

MUA – Mail User Agent

Software that performs mail operations on behalf of the user. For instance: Microsoft Outlook, Netscape Messenger, and Eudora are all MUAs. If one draws an analogy to letter and parcel delivery in the "real

world", an MUA is equivalent to a secretary or shipping clerk to which you delegate the tasks of making sure that your package is addressed properly and has the correct postage (i.e.: is well-formed Internet mail) and who will walk down to the corner of the street to put your package in the post box.

MAC – Message Authentication Code

An authentication technique involving the use of a secret key to generate a small fixed-size block of data that is appended to the message. This is also known as a cryptographic checksum. This technique assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key. The message plus the MAC is transmitted to the recipient B. B performs the same calculation on the received message, using the same secret key, to generate a new MAC. The message is authentic if the newly calculated MAC matches the received MAC[3].

The Mail System uses MACs in two different contexts. The Mail and PKI Systems share a secret key. This shared secret is used by the PKI system to generate an authentication certificate and a MAC key for each nym. The Mail system used the shared secret to calculate a MAC when attempting to validate an authentication certificate and to reconstruct a nym's MAC key when forwarding Internet to nym mail.

Protocols

POP3 – Post Office Protocol V3

A standard protocol used for the retrieval of mail from a remote server. When we say POP, we actually mean POP3, which is version 3 of the protocol. We have modified our POP server's authentication method slightly; see Section 4.4.5, POP Server, for more details.

SMTP – Simple Mail Transfer Protocol

The standard protocol used for the delivery of Internet mail. In its current deployment, the Freedom mail system speaks SMTP with the Internet.

ESMTP – Extended Simple Mail Transfer Protocol

An extension of the standard protocol used for the delivery of Internet mail. In its current deployment, the Freedom mail system uses a slightly modified ESMTP for authentication purposes.

QMQP – Quick Mail Queuing Protocol

QMQP comes from Dan Bernstein's Qmail package[4]. It is a wire protocol designed for rapid enqueueing of email from multiple queue writers to a shared message queue. Almost all Freedom Mail System traffic behind the firewall is in QMQP.

NFS – Network File System

Network File System (NFS) is a file system that will mount remote file systems across homogenous and heterogeneous systems. NFS consists of a client and server system. An NFS server can export local directories for remote NFS clients to use. It was originally developed by Sun Microsystems Computer Corp. (SMCC) and is now part of their Open Network Computing (ONC) initiative. All the Freedom Mail System traffic behind the firewall that is not in QMQP is in NFS[5].

NNTP – Network News Transfer Protocol	A standard protocol used to control how news messages are distributed, queried, retrieved, and posted. The Relay speaks NNTP when it sends news postings to News Servers.

Servers

IMEP – Internet Mail Encryption Proxy	These servers accept mail that originates on the Internet at large and is destined for any given nym. IMEP servers implement the SMTP protocol and are publicly accessible from the Internet. IMEP servers forward, using QMQP, all mail they receive to a randomly selected IMEPB (see below) for further processing. These are mini-qmail installations with a changed qmail-qmqpc to randomly choose and IMEPB. They access their QMQP servers across a firewall.
IMEPB – Internet Mail Encryption Proxy Backend	These servers process and deliver Internet-to-nym mail. IMEPB servers implement QMQP to accept and queue mail from any given IMEP. IMEPB servers are not accessible to nyms or from the Internet; only the IMEP servers can connect to the IMEPB servers, and only for the purposes of enqueuing mail for delivery via QMQP. These are stock qmail installations with our scripts put in to handle various paths. They are completely behind a firewall and accessible only from the IMEP.
NMTA – Nym Mail Transfer Agent	These servers accept mail that originates from a nym and is destined for a nym, for a newsgroup or for an Internet user. NMTA servers implement the ESMTP protocol and are accessible to nyms via anonymous routes whose exit node is a Core-AIP. NMTA servers authenticate the connecting nym and forward, using QMQP, all mail they receive to a randomly selected NMTAB (see below) for further processing. These are mini-qmail installations (they have no queue) with an ESMTP patch and a custom password checking utility. They are completely behind a firewall and accessible only through Core AIPS on the Freedom Cloud.
NMTAB – Nym Mail Transfer Agent Backend	These servers process and deliver nym-to-nym and nym-to-Internet mail. NMTAB servers implement QMQP to accept and queue mail from any given NMTA. NMTAB servers are not accessible to nyms or from the Internet; only the NMTA servers can connect to the NMTAB servers, and only for the purposes of enqueuing mail for delivery via QMQP. These are stock Qmail installations with our scripts put in to handle various paths. They are completely behind the firewall, and accessible only from the NMTA.
POP Server	These servers implement the POP3 protocol and allow nyms to retrieve their email. The POP servers are accessible to nyms via anonymous routes whose exit node is a Core-AIP. POP servers authenticate the connecting nym, and if they do not already have an existing mail account, will create one. These are qmail-pop3d installations, with a custom password checking

utility. These servers are completely behind the firewall and accessible only through Core AIPs on the Freedom Cloud.

POP store This file system holds the nyms' mail until they download it. The nym's configuration for spam blocking, block list and tagline are also stored on the POP store. It is completely behind the firewall and accessible only through the IMEPB, NMTAB and Pop servers.

Relay These servers perform final delivery of Internet and Usenet bound messages on behalf of nyms. It also delivers failure messages to Internet users if a nym bound message cannot be delivered. Its primary purpose is to ensure that the other mail system entities do not have to open connections outside of the mail system. This is a stock Qmail installation with a special delivery rule for the News case. It bridges the firewall and accepts connections from the IMEPB and NMTAB.

3 Paths

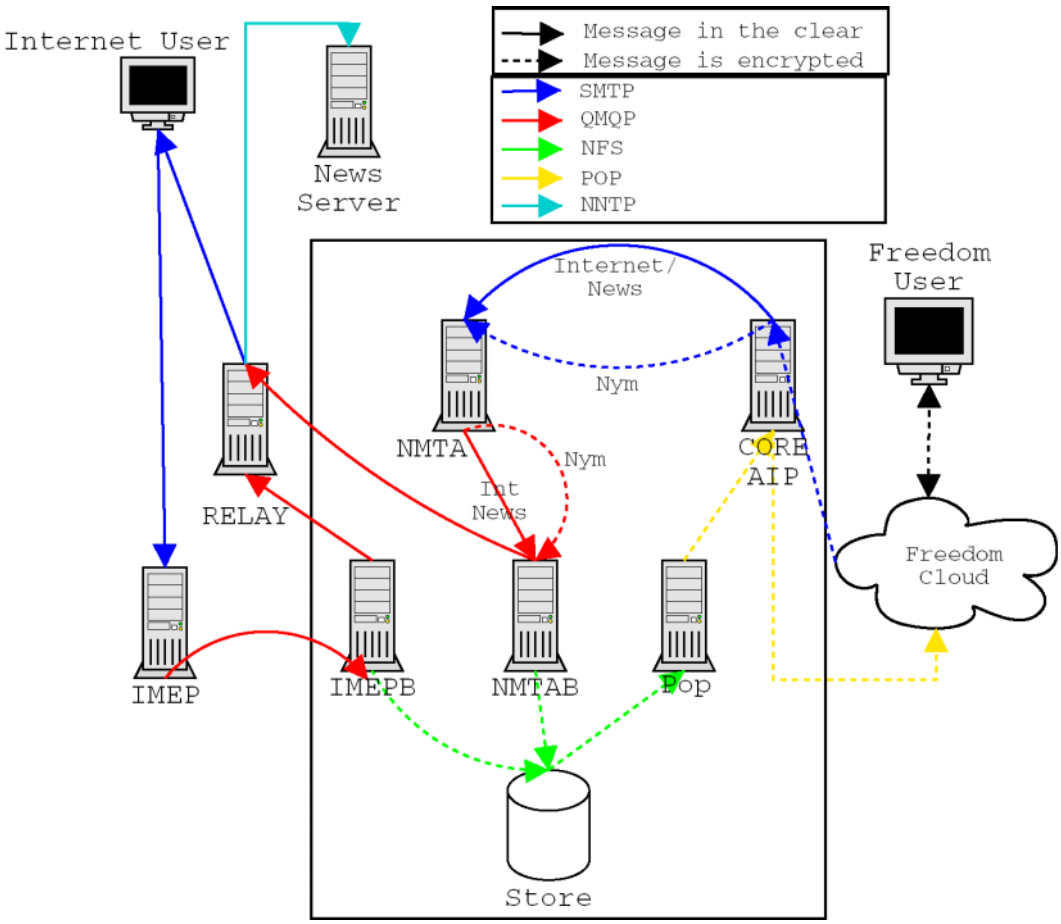

Figure 1: A High-Level Overview of How the Freedom Mail System Works.

Here we describe the different paths mail follows depending on its originator and recipient. Please see Section 4.3, Freedom Cloud, for information on how each of these paths is encrypted.

3.1 Nym Creation

The Mail and PKI systems share a secret key. When the PKI system creates a nym on behalf of a client, it uses the nym's mail information to construct a certificate for the nym. The veracity of the certificate is established by including a MAC that is calculated using the shared secret. More details on this certificate may be found in Section 4.12, Mail Certificate. It also uses the shared secret in order to generate a MAC key for the nym. Because the mail system also knows the secret, it is also capable of generating the MAC key when needed. More details on this key may be found in Section 4.11, Shared Nym-

Mail System MAC Key. The certificate is returned to the client along with all other nym information. The PKI system has no further need for the certificate, so it discards it.

Given the certificate for the newly created nym, the client then attempts to authenticate itself to a Freedom POP server. The POP server validates the certificate by calculating its MAC using the shared secret and comparing it to the MAC contained in the certificate. If the nym proves to be authentic, then the server validates the presence of the nym's mail directory. If no mail directory is present, one is created and a "Welcome to Freedom Mail" message is sent to the nym.

3.2 Nym to Internet

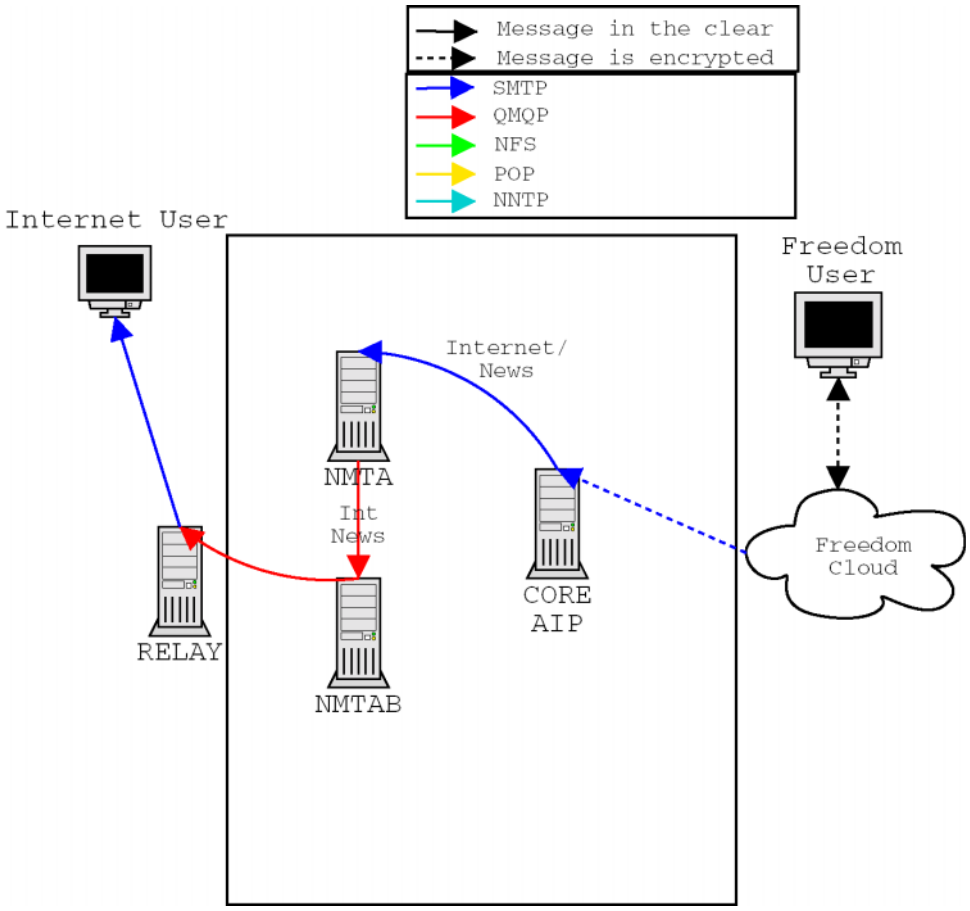

Figure 2: A High-Level Overview of the Nym to Internet Path.

To support the transmission of mail from nyms to the Internet (i.e.: non-Freedom mail addresses) the SMTP proxy component of the Freedom client software will accept plaintext connections from the end-user's MUA. After properly formatting the mail, the Freedom Client passes the Mail on to the NMTA using SMTP over an unauthenticated

route through the Freedom Cloud. The NMTA authenticates the nym and passes the mail on to the NMTAB, using QMQP. The NMTAB performs checks to ensure the message may be sent, and then passes it on to the Relay, using QMQP, from where it is sent off to the appropriate MTA, using SMTP.

A more detailed breakdown of the steps each entity performs may be found within Section 4, Entities.

3.3 Nym to News

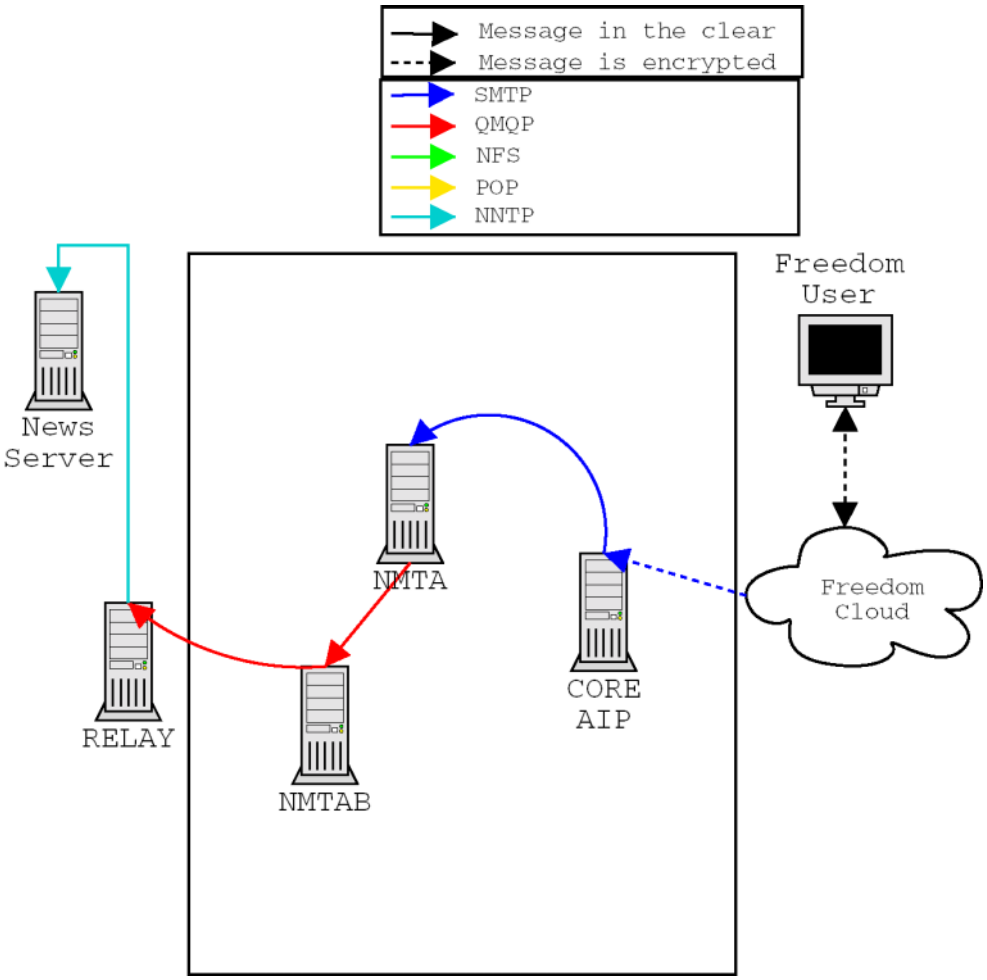

Figure 3: A High-Level Overview of the Nym to News Path.

To support the transmission of news postings from nyms to Usenet, the NNTP proxy component of the Freedom client software will accept plaintext connections from the end-user's MUA. After properly formatting the posting, the Freedom Client passes it on to the NMTA using SMTP over an unauthenticated route through the Freedom Cloud. The

NMTA authenticates the nym and then passes the posting on to the NMTAB using QMQP. The NMTAB, after performing various checks to make sure the nym may send the posting, passes it to the Relay, using QMQP, from where it is sent to the Freedom Network News Host using NNTP.

A more detailed breakdown of the steps each entity performs may be found within Section 4, Entities.

3.4 Nym to Nym

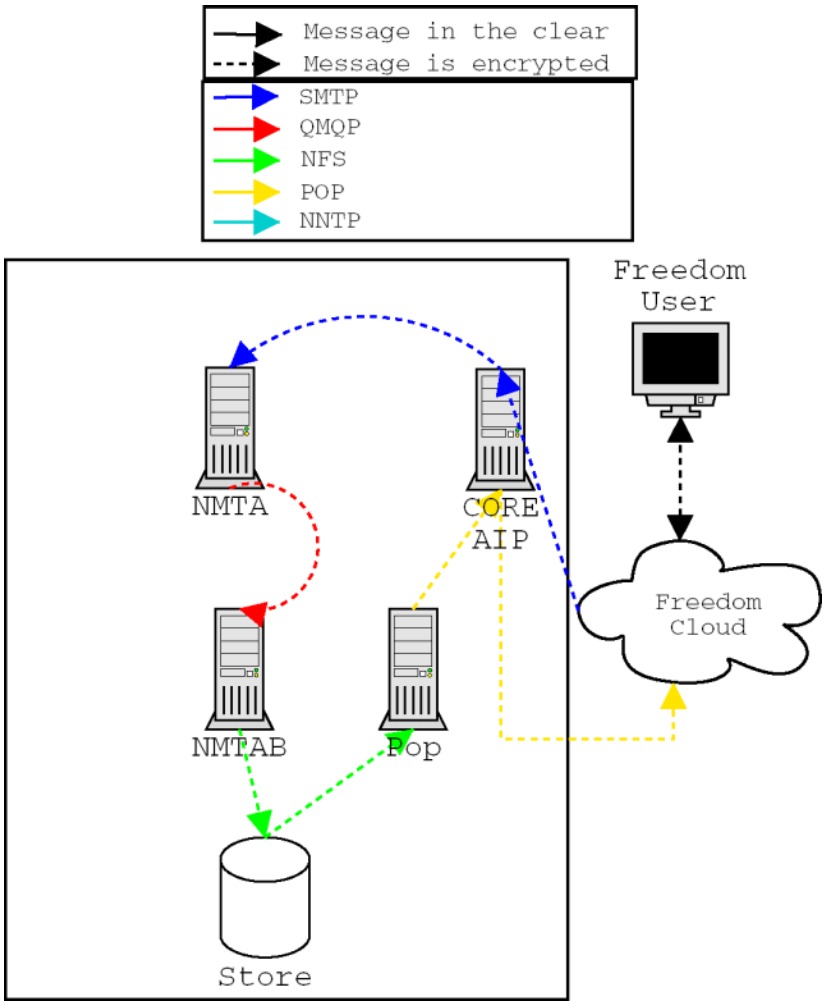

Figure 4: A High-Level Overview of the Nym to Nym Path.

To support the transmission of mail from nyms to the Internet (i.e.: non-freedom email addresses) the SMTP proxy component of the Freedom client software will accept plaintext connections from the end-user's MUA. After properly formatting and encrypting

the mail, the Freedom Client passes the mail on to the NMTA using SMTP over an unauthenticated route through the Freedom Cloud. The NMTA authenticates the nym and passes the mail on to the NMTAB, using QMQP. The NMTAB, after performing various checks to ensure the nym may send the mail, writes it to the receiving nym's mailbox over NFS.

A more detailed breakdown of the steps each entity performs may be found within Section 4, Entities.

3.5 Internet to Nym

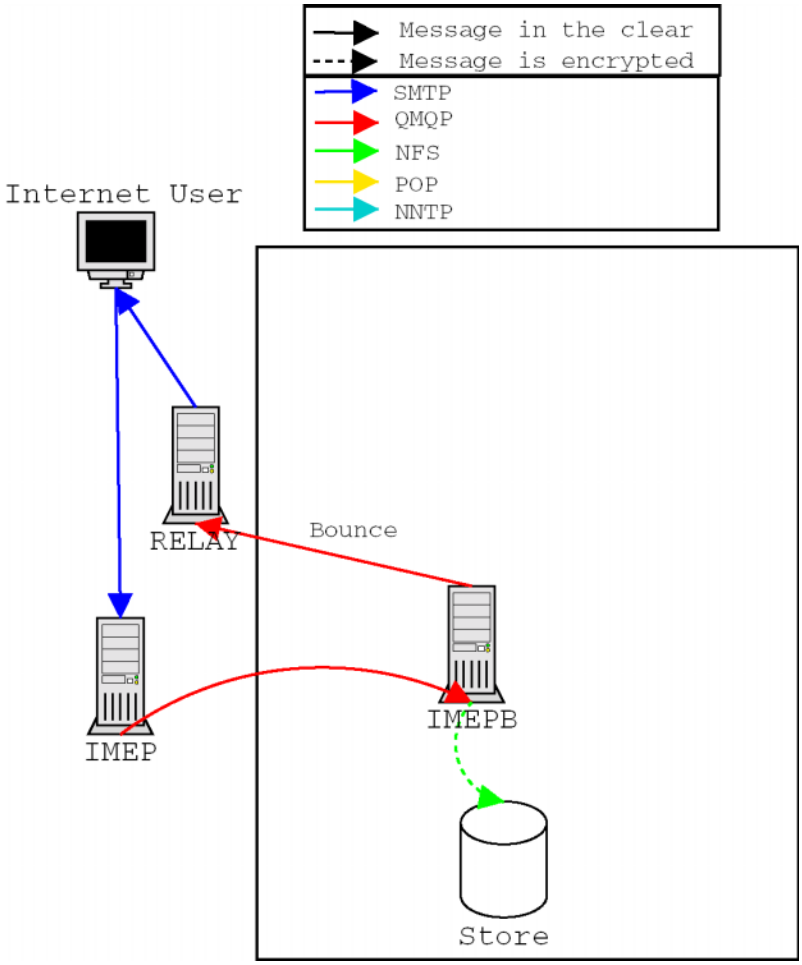

Figure 5: A High-Level Overview of the Internet to Nym Path.

To support the transmission of mail from Internet (i.e.: non-Freedom) users to nyms, the IMEP accepts and understands SMTP transactions with the Internet at large. The IMEP is configured to act as the mail host for all freedom.net email addresses. The IMEP

passes messages on to the IMEPB, using QMQP. The IMEPB, after performing various checks to make sure the mail may be sent, writes it to the receiving nym's mailbox over NFS. In the case of bounces or special aliased addresses (e.g.: postmaster@freedom.net might be aliased to postmaster@zeroknowledge.com), the IMEPB passes the mail on to the Relay, using QMQP, from where it is sent off to the appropriate MTA, using SMTP.

A more detailed breakdown of the steps each entity performs may be found within Section 4, Entities.

3.6 Mail Retrieval

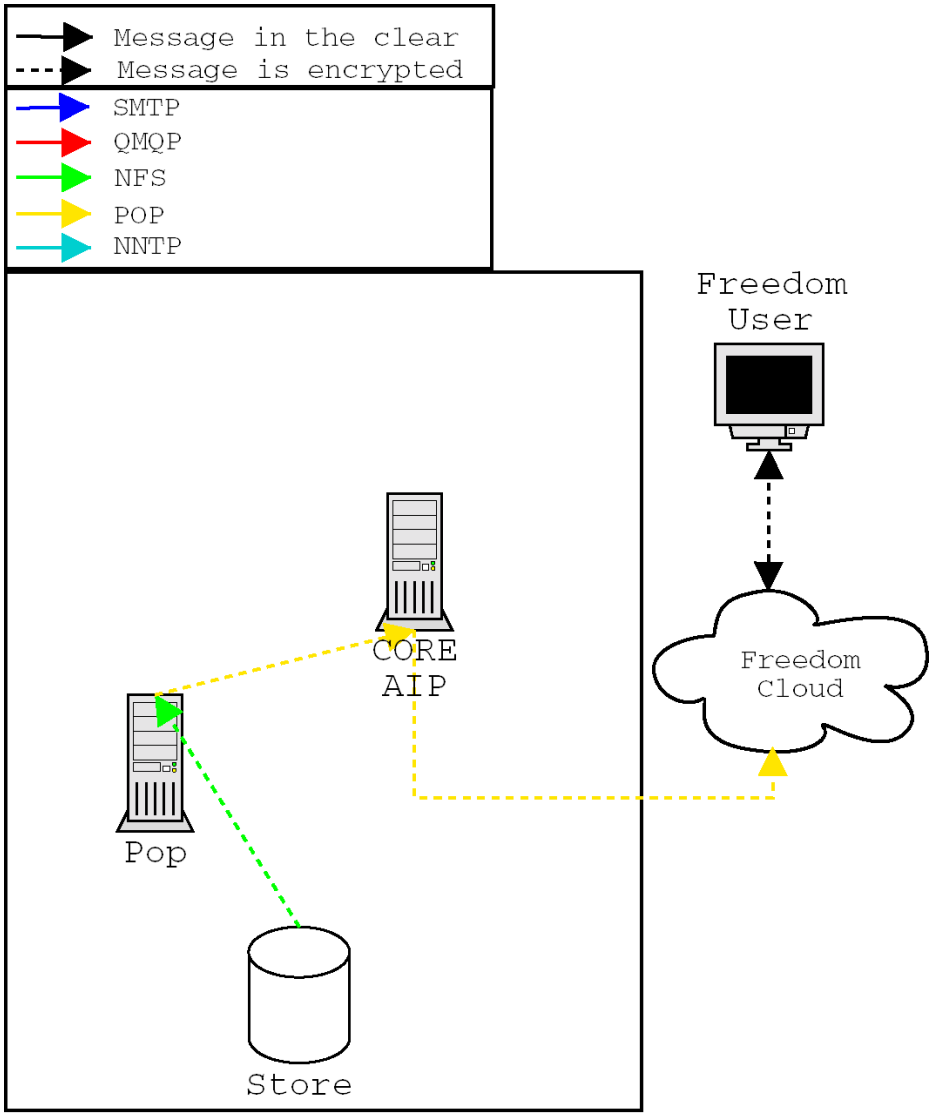

Figure 6: A High-Level Overview of the Mail Retrieval Path.

Mail retrieval is initiated when the POP proxy component of the Freedom Client software intercepts the user's MUA attempting to connect to a POP server. The Freedom Client authenticates itself to the POP server, using POP over an unauthenticated route through the Freedom Cloud, and downloads a list of messages to present to the MUA. It also connects to the user's POP server at his ISP and echoes the MUA's requests to obtain a list of messages. The Freedom Client then merges these lists into a virtual mailbox that it presents to the user's MUA. It then redirects the MUA's processing through the appropriate path so it can obtain the mail messages it requests.

In effect, the Freedom Client can talk to both the Freedom POP Server and the user's ISP POP server, and acts as a go-between for these two POP Servers and the user's MUA.

A more detailed breakdown of the steps each entity performs may be found within Section 4, Entities.

3.7 News Reading

The Freedom Mail System does not provide a pseudonymous news reading service. Instead, we recommend the user to read news through a web interface using an authenticated route over the Freedom Cloud if they wish to preserve their privacy while using Usenet. Please refer to "Freedom 2.0 Security Issues and Analysis" for more information.

4 Entities

4.1 Nyms

Nyms all need to have valid mail accounts that can send and receive mail. An "account" comprises:

- A nym record in the nym database of the PKI system.
- Public encryption and signature keys in the key database of the PKI system (by implication, the nym must have the corresponding private keys).
- Disk space in the mail system in order to store mail they receive.
- The configuration options denoting the operating parameters of the services to which the nym is entitled.

The mail system representation of an account consists of a directory holding per-nym configuration files as well as the nym's stored mail. Mail is stored using Maildir format mailboxes. Maildir is a standard directory format that supports multiple simultaneous delivery agents and is reliable over NFS.

The location of the account directory is calculated as:

```
h = 32_bit_hash( nym_name )
i = hex( h[23..20] )
j = hex( h[19..16] )
k = hex( h[15..12] )
l = hex( h[11..8] )
m = hex( h[7..4] )
```

hex is the hexadecimal representation using lowercase letters.

```
mail_dir = ${ACCOUNT_ROOT}/i/j/k/l/m/nym_name
```

This nym to directory mapping scheme is employed for the following reasons:

- It provides a manageable level of fanout at each level such that the physical distribution of the mail system storage can be spread across multiple file servers using different NFS mount points.
- The hash distributes nym account (and hence nym storage) across the account space. This helps to balance the load on the file servers.
- In addition, the hash distribution serves to generally reduce the number of nym accounts at each leaf of the directory tree. The search performance of many file systems degrades as the number of entries in the directory being searched gets larger.

The per-nym config files we store are:

.type	The numeric identifier for the nym type.
.quota	The storage quota for the nym.
.cross-post-limit	Maximum simultaneous newsgroups in a post.
.tagline	Indicates whether or not the Freedom Mail System should append tagline for nym to Internet traffic.
.send-limit	Maximum number of recipients per 24 hours.
.volume-limit	Maximum number of bytes sent per 24 hours (not enforced).
blocklist.cdb	hash of addresses blocked for delivery.
Stat/*	Counter for enforcing send-limit.

Since it is impractical for each nym to have its own user id, we are utilizing Paul Gregg's method of hosting an arbitrary number of mail accounts from a single mail system user id[6]. All deliveries to nym accounts are performed under a single user id that has read/write access to the mail store. The entities utilizing this user id are the NMTAB, IMEPB, and the Freedom POP Server.

Since qmail-local receives knowledge of user directories by delegating to our replacement version of qmail-getpw, frname2account, our password file does not map these directories or tell the POP server what the password should be. More information on user authentication can be found in Section 4.13, Mail Certificate Authentication Protocol.

4.2 Freedom Client

This section describes how the Freedom Client distributed by Zero-Knowledge handles mail. Be aware, however, that since the Linux Client has been open-sourced, there is a chance that a Freedom Client provided by somebody other than Zero-Knowledge may not work entirely in this manner.

4.2.1 Outgoing Mail

These are the common steps performed by the Freedom Client on all outbound mail regardless of whether the recipient is a nym, a non-Freedom Internet user, or a newsgroup. Upon receipt of an outgoing message, the Freedom Client's SMTP proxy:

1. Sanitizes the message, ensuring that it does not contain compromising information.
2. Modifies the message headers such that the sender is the currently selected nym.

Internet Recipient

Assuming the previous common steps have been successfully completed, for Internet recipients, the Freedom Client's SMTP proxy:

1. Constructs an authentication signature header for each Internet recipient using the nym's signing key. These can be later used by the recipient to validate that the sending nym really did send them the message. (Section 4.14, Mail Message Format.)
2. Constructs an unauthenticated private route through the Freedom cloud to an NMTA.
3. Authenticates the nym to the NMTA using the nym's authentication certificate. (Section 4.13, Mail Certificate Authentication Protocol.)
4. Transmits the recipient list and the message.
5. Waits for notification from the NMTA that the message has been accepted for delivery before informing the user's MUA that the message has been accepted for delivery.

News Recipient

Assuming the previous common steps have been successfully completed, for News postings, the Freedom Client's NNTP proxy:

1. Constructs an unauthenticated private route through the Freedom cloud to an NMTA.
2. Authenticates the nym to the NMTA using the nym's authentication certificate. (Section 4.12, Mail Certificate Authentication Protocol.)
3. Transmits the message using mail2news@freedom.net as the recipient.
4. Waits for notification from the NMTA that the message has been accepted for delivery before informing the user's News Agent that the message has been accepted for delivery.

Nym Recipient

Assuming the previous common steps have been successfully completed, for Internet Recipients, the Freedom Client's SMTP proxy:

1. Encrypts the message with the public encryption key of the recipient nym.
2. Constructs a message authentication header for the encrypted message; this header is signed with the sending nym's private signature key and encrypted with the recipient nym's public encryption key. (Section 4.14, Mail Message Format.)
3. Composes a failure notification message to be delivered to the sending nym if the mail system is unable to deliver the message to the intended recipient. This message is

encrypted using the sender's public-key (i.e.: the sender creates their own failure notification). (Section 4.6, Bounce Handler.)
4. Appends the failure notification to the real message.
5. Constructs a private route through the Freedom cloud to an NMTA.
6. Authenticates the nym to the NMTA using the nym's authentication certificate. (Section 4.13, Mail Certificate Authentication Protocol.)
7. Transmits the message and the recipient.
8. Waits for notification from the NMTA that the message has been accepted for delivery before informing the user's MUA that the message has been accepted for delivery.

4.2.2 Incoming Mail

To retrieve mail, for a given nym, the POP proxy component of the Freedom Client software:

1. Connects to a mail system POP server over an unauthenticated route through the Freedom Cloud.
2. Authenticates the nym to the POP server using the nym's authentication certificate (see MAC in the glossary).
3. Downloads a list of the messages in the POP box.
4. Connects to the user's POP server at the user's ISP.
5. Echoes the POP authentication commands issued by the user's MUA to the POP server at the user's ISP.
6. Presents the user's MUA with a consolidated list of the messages contained in the user and nym mailboxes.
7. Redirects each retrieval command issued by the user's MUA to the appropriate POP server for message download.
8. If the retrieval command refers to a message that resides on the user's POP server at their ISP then the message is forwarded through the proxy to the user's MUA with no other processing.
9. If the retrieval command refers to a message that resides on the POP server in the Freedom mail system then the message is first decrypted and verified by the proxy before being forwarded to the user's MUA. If the message failed to decrypt or verify, an error message is displayed.
10. Messages are deleted from the Freedom POP servers at the request of the Freedom Client POP proxy.

4.3 Freedom Cloud

All interactions with Core Freedom servers (PKI, KQS, and Mail) must occur through a Core AIP, which can only be accessed through either an authenticated or an unauthenticated route through the Freedom Cloud. Sending and retrieving mail involves Mail System interactions over unauthenticated routes through the Freedom Cloud. Public key lookups for nym to nym mail involve interactions with the KQS over unauthenticated routes through the Freedom Cloud. Nym creation involves Nym Server interactions over authenticated routes through the Freedom Cloud.

4.4 Freedom Mail Servers

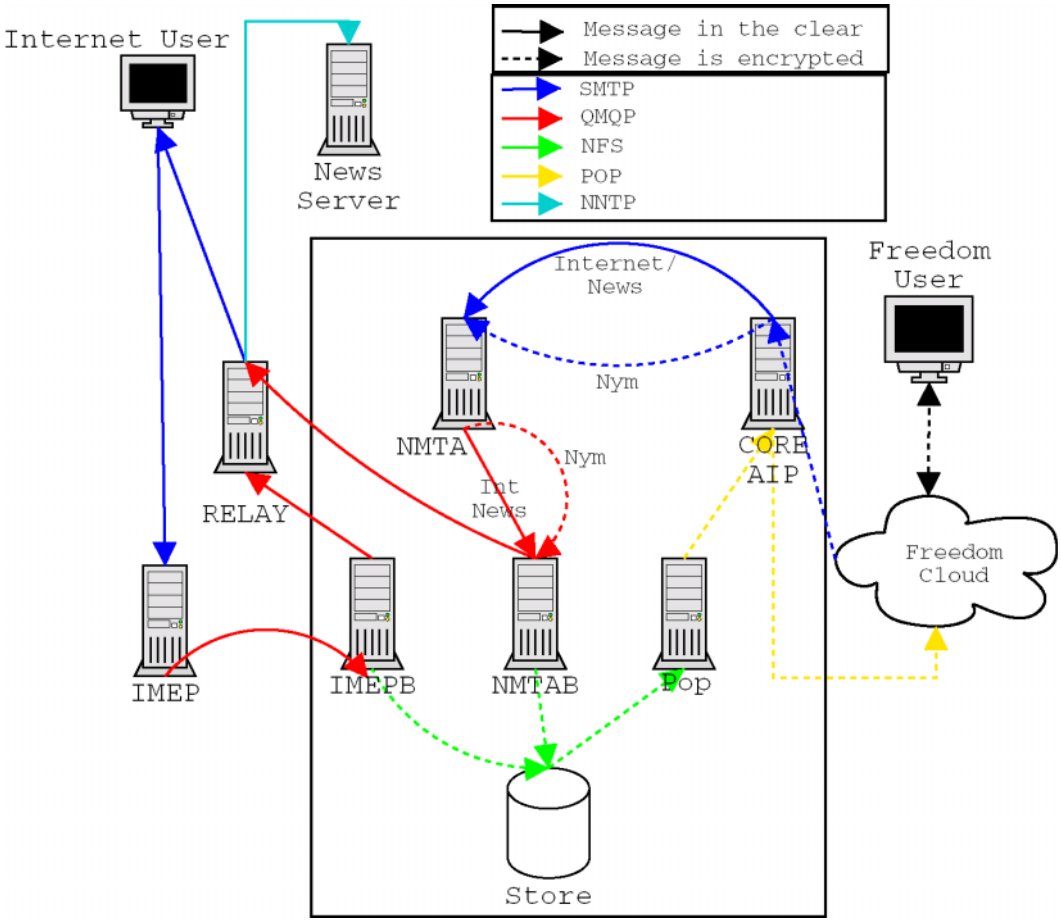

Figure 7: High-Level Overview Revisited.

4.4.1 IMEP

The IMEP, upon the opening of a new ESMTP connection from an MTA:

1. Validates that the recipient address is from the list of Internet domains that the mail system represents (i.e.: freedom.net).
2. Randomly selects an IMEPB.
3. Forwards the message to the IMEPB for further processing and delivery.
4. Waits for notification from the IMEPB that the message has been accepted for delivery before informing the MTA that the message has been accepted for delivery.
5. If it cannot contact the selected IMEPB, the IMEP selects another and retries, until it exhausts the set of IMEPB servers.

4.4.2 IMEPB

The IMEPB handles all the administrative and cryptographic manipulations needed for Internet to Nym mail. For the purposes of this section, the Internet sender is referred to as the sender, the recipient nym is referred to as the recipient, and a specific mail message is referred to as the message.

The IMEPB is a stock qmail installation with two sets of delivery rules: .qmail-default handles all nym deliveries using the following rules, and qmail-remote with a wrapper script for qmail-qmqpc for handling bounce messages which may occur within the IMEPB. The IMEPB is completely behind the firewall, and sends its bounce messages to the Relay.

Upon receipt of mail destined for a nym, the IMEPB:

1. Checks to see if the nym is aliased to an Internet address; if so, the mail is redirected to that address (e.g.: postmaster@freedom.net might be aliased to postmaster@zeroknowledge.com).
2. Validates that the recipient nym exists; if not, the message is bounced to the sender via the Relay.
3. If the recipient has enabled spam blocking, the message is examined to see if it came from a known spammer (as defined by the sending MTA being found in one or more of the known spammer lists consulted by the mail system). If the message is determined to be spam, it is discarded. (Section 4.7, Spam Control Program.)
4. Validates that the recipient nym has not blocked the sender; if so, the message is discarded. (Section 4.8, Block List.)
5. Verifies that the recipient nym has enough space in their mailbox to accept the mail. (Section 4.5, Mail Store Quota Enforcer.)
6. Derives the nym MAC key from the Mail/PKI shared secret.
7. Constructs a message authentication header using the nym MAC key.
8. Encrypts the message authentication header using the public encryption key of the nym recipient.
9. Encrypts the message body using the public encryption key of the nym recipient.
10. Adds the encrypted header and body to the recipient nym's mailbox.

4.4.3 NMTA

The NMTA, upon the opening of a new ESMTP connection from a nym:

1. Authenticates the nym's authentication certificate by validating its MAC using the Mail/PKI shared secret.
2. Validates that the sender address on the message envelope is the same nym name as that contained in the authentication certificate.
3. Validates that the sender address is from the list of Internet domains that the mail system represents (i.e.: freedom.net).
4. Randomly selects an NMTAB.
5. Forwards the message to the NMTAB for further processing and delivery.

6. Waits for notification from the NMTAB that the message has been accepted for delivery before informing the client SMTP proxy that the message has been accepted for delivery.
7. If it cannot contact the selected NMTAB, the NMTA selects another and retries, until it exhausts the set of NMTAB servers.

4.4.4 NMTAB

The NMTAB handles all the administrative manipulations needed for Nym to Nym, Nym to Internet mail and Nym to News. These three paths are handled separately (though they do perform some of the same functions). The NMTAB determines which path to follow by examining the message envelope.

Basically, the NMTAB is a stock qmail installation with the following delivery rules: .qmail-default handles all nym-to-nym deliveries; .qmail-config-* handles special configuration deliveries, such as turning on/off spam blocking, block list and taglines; .qmail-mail2news handles news posting; and a replacement wrapper script for qmail-remote that implements our nym to Internet deliver rules by using qmail-qmqpc to send the message to the Relay using QMQP.

The NMTAB, upon receipt of a new QMQP connection from an NMTA:

1. Writes the message to the queue.
2. Notifies the NMTA that the message has been accepted for delivery.
3. Validates that the nym has not exceeded their daily limit of outgoing mail.
4. Validates that the nym has not been blocked from sending mail to the given recipient(s) (Section 4.8, Block List), or in the case of a News posting, validates that the nym has not exceeded their cross-post limit.

Subsequently, in the case of Nym to Internet mail, the NMTAB queue handler processes the message by:

1. Validating that the mail contains one authentication signature header for each recipient (each recipient only receives their particular authentication signature header, see Section 4.14, Mail Message Format for further details).
2. Reforming each message in MIME format on a per recipient basis, stripping signatures for all other recipients. Please see Section 4.9, Authentication Headers, for more information.
3. Checking if the user allows taglines. If so, attach tagline to message. Please see Section 4.10, Tagline Inserter, for more information.
4. Submitting the message to the Relay, via QMQP, for final delivery.
5. Upon notification that the Relay has accepted the message for delivery, removing the message from the queue.

Subsequently, in the case of nym to news posting, the NMTAB queue handler processes the message by:

1. Checking if the user allows taglines. If so, attach tagline to message. Please see Section 4.10, Tagline Inserter, for more information.

2. Submitting the message to the Relay, via QMQP, for final delivery to mail2news@news.freedom.net.
3. Upon notification that the Relay has accepted the message for delivery, removing the message from the queue.

Subsequently, in the case of nym to nym mail, the NMTAB queue handler processes the message by:

1. Validating that storing the message in the recipient nym's mailbox would not cause the nym to exceed their storage quota.
2. If the message exceeds the recipient's quota, the pre-constructed failure notification is added to the sending nym's mailbox; otherwise, the main message is added to the recipient nym's mailbox. (Section 4.5, Mail Store Quota Enforcer.)
3. The message is removed from the queue.

`qmail-esmtpd` We have extended some existing patches to qmail-smtpd which enable esmtp AUTH LOGIN style authentication checking, which it delegates to the authentication system's checkpasswd (see Section 4.13, Mail Certificate Authentication Protocol, for more information). In addition, the esmtp patches compare the authenticated nym name to the envelope sender of each message, and reject non-matches. In addition, mixed message types (nym and Internet, or multiple nyms) are rejected as they should have been separated by the client.

4.4.5 POP Server

Our POP server is a stock qmail-pop3d installation with a replacement function for authentication. For more information on this process, please see Section 4.12, Mail Certificate Authentication Protocol.

4.4.6 Relay

The Relay is a stock qmail installation that negotiates with Mail Delivery Hosts across the Internet to deliver mail in its queue. In the special case of news, which contains domains of the form @news.freedom.net, it reforms the messages into one compatible with News servers, reforming the destinations to the actual newsgroups, and then passes them to Freedom Network News Server.

The Relay, upon receipt of a new QMQP connection from an NMTAB:

1. Writes the message to the queue.
2. Notifies the NMTAB that the message has been accepted for delivery.

Subsequently, for normal mail, the Relay queue handler processes the message by performing an SMTP delivery to the appropriate MTA. The Relay queue handler processes message sent to mail2news@news.freedom.net by performing an NNTP posting to the freedom network news host.

4.5 Mail Store Quota Enforcer

The Mail Store Quota Enforcer ensures that each nym's mailbox doesn't exceed the amount of disk space that has been allocated to the nym.

Since all investigated out-of-the-box solutions for quotas have presented incompatibilities with our mail system design, the mail quota enforcer was implemented from scratch. In its current incarnation the quota enforcer is a C program that implements a simple and inefficient quota-checking scheme.

4.6 Bounce Handler

In the case that mail cannot be delivered and the sender should be notified, we have to manipulate the message to deliver a properly pseudonymized and confidential bounce notice.

To facilitate this, the client constructs the proper bounce notification that can be extracted from the message by the mail system. With a MIME based message format, we can use the reformime utility to extract the parts we want to store/bounce when delivering messages.

In this manner, we are able to ensure that the following goals are met:

1. Mail is never dropped. Blocked mail is covered in Section 4.8 and is considered a "successful delivery".
2. Nyms never receive any cleartext in their inbox.
3. Internet users find out when mail to a nym failed, receiving a message that actually makes some sense.
4. Nyms find out when their mail failed, and likewise receive a sensible, non-alarming message.
5. The postmaster finds out some lesser amount of information when a bounce-loop occurs; only enough information to diagnose the problem, not any cleartext that could be a privacy leak.

The bounce handler must be examined in each of the different contexts in which it may be invoked.

1. *Internet to nym*: a bounce is generated using the standard qmail bounce mechanism if any of the quota checks fail or if the nym does not exist. The cleartext message is returned.
2. *Nym to Internet*: the bounce message enters the mail system via the IMEP and is handled as a normal Internet to nym message, and is called only if the mail had reached the Relay before bouncing.
3. *Nym to nym*: the sender sends a 2-part MIME message (as formatted by the client), whose second part is a short, explicit bounce message pre-encrypted for themselves. If a bounce occurs, the second part of the MIME message is extracted, formed into its own MIME message, and sent back to the sender. Otherwise, the non-bounce portion is extracted and reformed into just the message and sent on to the recipient.

4. *Internet to Internet*: our mail system does not relay for any non-nyms, rendering this bounce functionality moot.

4.7 Spam Control Program

The spam control program is optionally called in the Internet to Nym case to block known spammers, using Realtime Blackhole List (RBL) style authentication.

A modified version of rblcheck[7] (changes fed back upstream) is configured as a qmail-command[8] delivery target. Mail is sent to it on stdin, it parses the qmail `Received:` line and looks the connecting IP up in a variety of anti-spam DNS servers. If any of the tests succeed (the mail is from a spam source) the program exits 99, aborting further delivery attempts. If all tests fail, the program exits 0, and qmail-local[9] will carry on delivering to the user's Maildir.

4.8 Block List

The message blocking functionality searches a list of items to see if a given sender or recipient has revoked, or had revoked, their permission to perform a given operation. This is used for:

- Revoking nym access to the mail system.
- A nym wishing to block delivery from unwanted senders.
- An Internet address wishing to block delivery from unwanted nyms.

This allows us to, on behalf of both Freedom and non-Freedom users, make sure that an email recipient can request (and have enforced) their desire not to receive unwanted mail from a particular sender or domain delivered via the Freedom mail system. We want to silently drop messages through conversations that have been expressly blocked.

There are four situations where we would want to "block" a user or combination of users:

1. **Block on Log-In to POP server**
If we wish to be able to revoke the mail system usage privileges of a nym the authentication system checks whether or not the nym attempting to authenticate itself is located in a block list.

2. **Block on Log-In to NMTA**
If we wish to be able to revoke the mail system usage privileges of a nym the authentication system checks whether or not the nym attempting to authenticate itself is located in a block list.

3. **Block on Internet Delivery**
When attempting to deliver a message from a nym to an Internet address, we validate the following conditions:
- the sender is allowed to have mail delivered on their behalf;
- the recipient has not requested that all mail from the Freedom network to their address not be delivered;

- the recipient domain has not requested that all mail from the Freedom network to their address not be delivered;
- the recipient has not requested that all mail from the sending nym to their address not be delivered;
- and the recipient domain has not requested that all mail from the sending nym to their address not be delivered.

4. **Block on Nym Delivery**

When attempting to deliver a message to a nym, from either an Internet address or another nym, we validate the following conditions:

- the sender is allowed to have mail delivered on their behalf;
- the recipient has not requested that all mail from the sender not be delivered;
- and the recipient has not requested that all mail from the sender's domain not be delivered.

The block list functionality is implemented by storing the block list within the mail system. This is combined with a filter rule to the delivery mechanism which checks that the sender isn't in that list. To update the block list, a nym sends a new list to the reserved nym blocklist@freedom.net (all done by the client) that validates that the sender is legitimate, runs the plaintext link through the hashing function, and replaces the block list stored for the sender with the one contained in the message. The list of blocked senders contains a hash of the names rather than the actual sender names (for privacy purposes). Blocklists are accessible to the mail system, but are not a matter of public record.

We implemented this functionality by maintaining a database of revoked target recipient pairs. This is a database of keys without corresponding values. The presence of a key in the database indicates that the sender-recipient pair it represents is a disallowed delivery path. Thus, a block list is a database of keys having the general form:

```
hash([sender][->recipient])
```

where sender is the email address or domain of the sending party and recipient is the email address or domain of the receiving party. Both sender and recipient are optional.

When the system wants to validate that a particular operation is allowed to take place, it searches for the appropriate key in the database. For instance, if the mail system wants to validate that it is allowed to deliver mail to an Internet recipient on behalf of a particular nym, it performs the following steps:

1. normalizes the sender;
2. normalizes the sender's domain;
3. normalizes the recipient's domain;
4. normalizes the recipient;
5. checks if "`hash(sender)`" is in the database;
6. checks if "`hash(sender domain)`" is in the database;
7. checks if "`hash(->recipient)`" is in the database;
8. checks if "`hash(->recipient domain)`" is in the database;
9. checks if "`hash(sender->recipient domain)`" is in the database;
10. and checks if "`hash(sender->recipient)`" is in the database.

If any key is found in the database then the mail is discarded by the mail system.

The following sample global block list is implemented in the following manner. Here is the plaintext block list:

```
alice@freedom.net,bob@hotmail.com
aol.com
,foo@bar.net
```

The mail system will convert it into a CDB archive. The example block list prohibits:

1. alice@freedom.net from sending to bob@hotmail.com.
2. anyone from aol.com from having their mail delivered to the recipient.
3. anyone from sending mail to foo@bar.net.

Typically the nym->Internet block list will contain the first type of entry (i.e.: blocking specific conversations) where the sender is a particular nym and the recipient is either an email address of a domain.

Typically, a nym block list will contain the second type of entry (i.e.: blocking specific senders) because the recipient is denoted by the owner of the list, and is therefore redundant. The sender may be either a specific email address or a domain.

4.9 Authentication Headers

When a nym sends to the Internet, we cannot know which envelope recipients are Bccs and which are supposed to know about each other. As a result, we must generate separate authentication headers for each possible recipient and separate them out during remote delivery. This process was initially called "exploding" the mail, before we learned that qmail's remote delivery policy is to always deliver each recipient separately along its own connection. Our task is made simpler: we simply strip out, in each delivery, the authentication headers which do not correspond to the current envelope recipient; ensure the `From` field matches the sending nym. The `X-Freedom-Envelope-Sig` header can only be used to verify that the sender did or did not originate a message to the recipient. It cannot guarantee the other `To` or `Cc` fields are accurate, nor can it guarantee the message hasn't been tampered with.

To facilitate this task, we have designed the net2nym authentication header to be easy to grep for. It takes the following form:

```
X-Freedom-Envelope-Sig: recipient@isp.net
```

Thus, an email message being sent to bob, alice and fred might appear something like this:

```
X-Freedom-Envelope-Sig: bob@secret.gov oi3hj4b0b0ihjbrojkgfb0gdhbv97863764ghjbg0
X-Freedom-Envelope-Sig: alice@evil.com 03iub0ufb08734bo3hb09u3bvr9ubfi0u3br0fb03
X-Freedom-Envelope-Sig: fred@naive.org 3rfg9u43ygr934gf9ugr9uhfb394gf934ugf9uyfg
From: Joe the Mischievous <fred@trouble.net>
To: Bob The Intrepid <bob@secret.gov>
Cc: Alice the Wealthy <alice@evil.com>
Subject: How evil can you get?

hey, I was wondering, have any of you been doing really bad stuff to fred
recently?
-joe
```

which would subsequently be delivered as the 3 separate messages:

```
X-Freedom-Envelope-Sig: bob@secret.gov oi3hj4b0b0ihjbrojkgfb0gdhbv97863764ghjbg0
From: Joe the Mischievous <fred@trouble.net>
To: Bob The Intrepid <bob@secret.gov>
Cc: Alice the Wealthy <alice@evil.com>
Subject: How evil can you get?

hey, I was wondering, have any of you been doing really bad stuff to fred
recently?
-joe
```

```
X-Freedom-Envelope-Sig: alice@evil.com 03iub0ufb08734bo3hb09u3bvr9ubfi0u3br0fb03
From: Joe the Mischievous <fred@trouble.net>
To: Bob The Intrepid <bob@secret.gov>
Cc: Alice the Wealthy <alice@evil.com>
Subject: How evil can you get?

hey, I was wondering, have any of you been doing really bad stuff to fred
recently?
-joe
```

```
X-Freedom-Envelope-Sig: fred@naive.org 3rfg9u43ygr934gf9ugr9uhfb394gf934ugf9uyfg
From: Joe the Mischievous <fred@trouble.net>
To: Bob The Intrepid <bob@secret.gov>
Cc: Alice the Wealthy <alice@evil.com>
Subject: How evil can you get?

hey, I was wondering, have any of you been doing really bad stuff to fred
recently?
-joe
```

4.10 Tagline Inserter

For nym to Internet mail, if the sending nym has not turned off the tagline option, this function appends/inserts the Freedom tagline into the message.

The basic approach is to send all outgoing mail thru a pipeline, which pipes the message to stdout if they have turned off this option, and otherwise pipes to a MIME unpacker, followed by a concatenation of the tagline and a MIME re-packer.

The tagline inserter first converts its input to a MIME multipart message (if needed). It then examines the MIME parts of the email and appends the supplied tagline to the appropriate places. If the MIME type of the message is multipart/alternative then the tagline is appended to the first plain text and the first HTML parts of the message. If the MIME type is single part text or HTML then the tagline is appended only to the appropriate part.

If a text/plain or text/html is base64 encoded, that part is first decoded, the tagline is appended, and the part is re-encoded.

4.11 Shared Nym-Mail System MAC Key

In order to save public key signature operations in the mail system, we have a secret shared between the mail system and each nym. This symmetric secret is generated by the nym server at nym creation time, and sent back with the mail system authentication certificate. The nym server erases its copy after transmission to the client. Please see Section 4.12, Mail Certificate, for more information on the Mail Certificate.

The client stores the shared secret with the same security as private keys, the mail system certificate, and so on. It never needs to transmit the shared secret; it only uses it to verify MACs on downloaded mail from the Internet.

The key is generated, by the nym server, like so:

```
key = SHA-1(secret || entity)
```

where secret is the secret shared between the mail system and the nym system, and entity is the entity identifier of the newly created nym (i.e.: entity type and name).

The mail system is able to generate this key on the fly, as it knows both the secret and the entity.

Should the secret change (e.g. after a detected compromise), the nym will no longer be able to authenticate itself to the mail system because its certificate is invalid. It will connect to the nym server to get a new certificate, and at the same time get a new MAC key. MAC verifications on existing mail will fail, but this is the proper behavior, assuming a compromise.

4.12 Mail Certificate

The certificate is generated by the Nym Server at nym creation time, and sent to the user. This certificate is used as a password in interactions with the NMTA and the POP Server. At the same time, the nym server also generates and sends the MAC key shared by the

mail system and the nym. Please see Section 4.11, Shared Nym-Mail System MAC Key, for further information.

If the secret key shared between the Freedom Mail System and the Freedom Nym Server is compromised, the Freedom server operators will change it upon detection. This will, of course, invalidate all the certificates and shared Mail-Nym MAC keys. Upon the Freedom Client's next authentication attempt, the mail server check-password function tells the user that the user name and authentication failed. Then the nym re-authenticates itself to the nym server using its private signature key, and obtains a replacement certificate using the new secret key.

The certificate can only be validated by the Nym Server and the Mail System, because it is signed by the Mail/PKI shared secret. A certificate looks like this:

pkt-vers (1)	pkt-type (1)	ent	vers (2)	exp (8)	type (1)	mail (4)	vol (4)	size (4)	cross (4)	mac (10)

where the fields are:

pkt-vers A 1 byte value indicating the version of this packet format, currently 1.

pkt-vers A 1-byte value indicating the type of this packet, currently 1.

ent The name of the nym, preceded by a 1-byte type.

vers The version of the mail certificate.

exp The expiration date of the certificate (from a ZkClock object).

type The type of the account of the entity, (e.g.: premium).

mail The mail-limit (maximum number of mail a user can send).

vol The volume-limit (maximum number of bytes a user can send).

size The pop-quota (maximum size, in bytes, of the owner's mailbox).

cross The cross-post limit (maximum number of simultaneous newsgroup the user can send).

mac The 10-byte mac over the preceding bytes.

All multi-byte data fields are considered to be in network byte order (big endian).

4.13 Mail Certificate Authentication Protocol

In the POP mail system, we are building upon the existing authentication mechanisms of POP, and ESMTP. However, these mechanisms typically involve the user sending his username and password to the server, and the server having a database of user names and passwords, which case we want to avoid.

The following authentication protocol is designed to remove a database lookup, and to allow the user's password to securely pass attributes about the user. In particular, we

want to avoid having to contact the nym server or key server in order to authenticate the user, as these transactions waste time and processing power, as well as inhibit scalability.

The secret key is shared by the mail system and the Nym Server. When the user connects to the mail server using ESMTP or POP, he sends his nym name as the USER and the certificate as the PASS field. In the future, the mail system will store the nym's current certificate version in a file in its mail directory, enabling it to cheaply verify the certificate using the check-password function, which will just verify that the MAC is valid, and that the version numbers are correct.

The certificate may never be sent in the clear to the mail server, as someone may snoop on the connection and get it. The certificate authentication mechanism is vulnerable to replay if used directly so must always be used inside an encrypted tunnel, which has replay protection, or inside trusted networks. For this reason, it is only used by the Freedom Client in the context of unauthenticated routes through the Freedom Cloud.

If the secret key shared between the Freedom Mail System and the Freedom Nym Server is compromised, the Freedom server operators will change it upon detection. This will, of course, invalidate all the certificates and shared Mail-Nym MAC keys. Upon the Freedom Client's next authentication attempt, the mail server check-password function tells the user that the user name and authentication failed. Then the nym re-authenticates itself to the nym server using its private signature key, and obtains a replacement certificate using the new secret key.

4.14 Mail Message Format

Following RFC 822, a mail header consists of two parts: the field-name, and the field-body. For example, in

```
From: bob@freedom.net
```

`From` is the field-name, and `bob@freedom.net` is the field-body.

We assume the user's mail user agent (MUA) produces a message like the following:

`From:` N_{snd}
`To:` N_{rcv_t}, I_{rcv_t}
`Cc:` N_{rcv_c}, I_{rcv_c}
`Bcc:` N_{rcv_b}, I_{rcv_b}

`text`

where N represents a nym, and I is a non-Freedom (Internet) user (there can be zero or more than one recipient for each type). Therefore, N_{rcv_c} would be the nym recipient from the Cc header. Call this M_{orig}, the original message, which includes both the RFC 822 headers and the text.

4.14.1 Pre-Processing Stage

These are the steps performed by the Client and IMEPB for all mail they handle. After this stage has been completed, the next set of rules to follow depends on the recipient type.

Client-Side Operations

1. Sanitize the headers on the original message, removing anything that might identify the Freedom user (as opposed to the nym). Call the resulting sanitized message M_{san}.
2. Break up the recipients into the following classes:
- each nym recipient, whether in a `To`, `Cc` or `Bcc` header becomes the sole member of its own class;
- each non-nym recipient in a `Bcc` header becomes the sole member of its own class;
- all non-nym recipients in `To` and `Cc` headers become members of the same class.

This classification is necessary because the format of the message differs according to whether the recipient is a nym or not (mail to nyms needs to be publicly encrypted with their key; mail to an Internet user doesn't), and because non-`Bcc` recipients must not be aware of `Bcc` recipients.

3. For each class of recipient make a copy of M_{san}, using the exploder functionality. (Section 4.9, Authentication Headers.)
4. For each class of recipient generate a list of their names for the recipients header of the message envelope.
5. If the recipient is an Internet address, create an `X-Freedom-Envelope-Sig` header containing a base-64 encoded signature over the field-bodies of the two headers: `From` and `To`. Add this header to M_{san}. This signature can only be used to verify that the sender did or did not originate a message to the recipient. It cannot guarantee the other `To` or `Cc` fields are accurate, nor can it guarantee the message hasn't been tampered with. In the nym recipient case, the client verifies this signature. In the Internet recipient case, Zero-Knowledge's Abuse department will verify signatures on a case-by-case basis.

IMEPB-Side Operations:

1. Break up the recipients into the following classes:
- each nym recipient, whether in a `To`, `Cc` or `Bcc` header becomes the sole member of its own class.

We don't need to worry about any of the other classifications we saw in the Client-side operations because the IMEP will not accept mail destined for non-Freedom users.

2. For each class of recipient make a copy of the message and call it M_{san}, using the exploder functionality (Section 4.9, Authentication Headers).
3. For each class of recipient generate a list of their names for the recipients header of the message envelope. Since each nym is put in its own class, the `To` header of these message envelopes should each contain one and only one name.

4.14.2 Internet Recipient

If the **Pre-processing stage** has been followed by the client, the mail is now ready for delivery to the recipient and is passed off to the NMTA.

4.14.3 Nym Recipient (Including News)

This section describes the format of the message destined to a nym, either from a non-Freedom user (via the IMEP), or from another nym. It is assumed that the previous steps have been followed by the client or IMEPB, and that we now have a copy of M_{san}.

1. Create a special binary header, H_{msg}, containing the following fields

    sender [+ *Internet email address*]
    recipient
    hash(M_{san})
    flag
    {$S_{N_{snd}}$}(hash(*sender*, *recipient*, hash(M_{san})))

In the Internet sender case, the sender field will be the NMTAB and there will be an Internet email address field based on the message envelope sender (which will be used for the blocking list). For the nym sender case, the client ensures that the sender field is the sending nym's name. Since the mail is destined to a nym, there can be only one recipient. The flag field indicates whether the message is authenticated with a MAC (value is 0) or a signature (value is 1). When a nym sends mail to another nym it will use a signature, signing with its private signing key, while the NMTAB will generate a MAC using the secret shared with the recipient nym. (Section 4.11, Shared Nym-Mail System MAC Key.)

2. Encrypt M_{san} using the recipient nym's public encryption key.

3. For the nym sender case, the client generates a bounce message, M_{bounce} that will be delivered to the sender in case of an error. The bounce message has a header, H_{bounce}, that is similar to H_{msg}, with the exception that the sender and recipient fields are both the sending nym's name. The bounce header and message are encrypted with the sending nym's public encryption key. More information on the bounce message may be found in Section 4.6, Bounce Handler.

4.Compose all these pieces into a MIME message

    base64(M_{san})
    base64(M_{bounce})

If the message cannot be delivered, it is reformed keeping just the bounce section and placed in the sending nym's mailbox, and the rest of the message is be thrown away. If the message can be delivered, it is reformed with the bounce section removed.

Reference

Cameron Liard's notes on MTAs
http://starbase.neosoft.com/~claird/comp.mail.misc/MTA_comparison.html
"Life with qmail". Dave Sill http://Web.InfoAve.Net/~dsill/lwq.html
http://stommel.tamu.edu/~baum/linux/LDP/HOWTO/mini/Mail2News-9.html
http://www.tibus.net/pgregg/projects/qmail/single-uid-howto.html
safecat, http://www.nb.net/~lbudney/linux/software/safecat.html
maildrop, http://www.flounder.net/~mrsam/maildrop/
ORBS, http://www.orbs.org
mail-abuse.net, http://www.mail-abuse.net

Endnotes

[1] "Freedom 2.0 Security Issues and Analysis", Adam Back, Ian Goldberg, Adam Shostack, Zero-Knowledge Systems, Inc.
[2] “Freedom System 2.0 Architecture”, Philippe Boucher, Adam Shostack, Ian Goldberg, Zero-Knowledge Systems, Inc.
[3] The NetWorking Dog Glossary
http://www.networkingdog.net.au/Glossary/MessageAuthenticationCode.htm
[4] qmail D. J. Bernstein http://www.qmail.org
[5] "The PC-Mac TCP/IP & NFS faq" (comp.protocols.nfs)
http://www.rtd.com/pcnfsfaq/SecA.html
[6] http://www.tibus.net/pgregg/projects/qmail/single-uid-howto.html
[7] http://rblcheck.sourceforge.net/
[8] http://www.qmail.org/man/man8/qmail-command.html
[9] http://www.qmail.org/man/man8/qmail-local.html