Network Working Group
Internet-Draft
Expires: June 29, 2005

U. Moeller
Secardeo GmbH
L. Cottrell
Anonymizer, Inc.
P. Palfrader
The Mixmaster Project
L. Sassaman
Nomen Abditum Services
December 29, 2004

Mixmaster Protocol Version 2
draft-sassaman-mixmaster-03.txt

Status of this Memo

Copyright Notice

Abstract

Most e-mail security protocols only protect the message body, leaving useful information such as the identities of the conversing parties,

sizes of messages and frequency of message exchange open to adversaries. This document describes Mixmaster version 2, a mail transfer protocol designed to protect electronic mail against traffic analysis.

Mixmaster is based on David Chaum's mix-net concept. A mix (remailer) is a service that forwards messages, using public key cryptography to hide the correlation between its inputs and outputs. Sending messages through sequences of remailers achieves anonymity and unobservability of communications against a powerful adversary.

Table of Contents

1. Introduction

This document describes a mail transfer protocol designed to protect electronic mail against traffic analysis. Most e-mail security protocols only protect the message body, leaving useful information such as the identities of the conversing parties, sizes of messages and frequency of message exchange open to adversaries.

Message transmission can be protected against traffic analysis by the mix-net protocol. A mix (remailer) is a service that forwards messages, using public key cryptography to hide the correlation between its inputs and outputs. If a message is sent through a sequence of mixes, one trusted mix is sufficient to provide anonymity and unobservability of communications against a powerful adversary. Mixmaster is a mix-net implementation for electronic mail.

This memo describes version 2 of the Mixmaster message format, as used on the Internet since 1995.

2. The Mix-Net Protocol

The mix-net protocol [1] allows one to send messages while hiding the relation of sender and recipient from observers (unobservability). It also allows the sender of a message to remain anonymous to the recipient (sender anonymity). If anonymity is not desired, authenticity and unobservability can be achieved at the same time by transmitting digitally signed messages.

This section gives an overview of the protocol and messaging pattern. The mixing algorithm is specified in Section 3, and the message format is specified in Section 4.

Viewed from a high level, Mixmaster is like a packet network, where each node in the network is known as a "remailer." The original content is split into pieces, and an independent path is determined for each piece, with the only requirement that all paths must end at the same remailer. Each piece is multiply encrypted so that any intermediate remailer can only decrypt enough information to determine the next hop in the path. When all pieces have arrived at the final remailer, the original content is re-created and sent to its final destination.

2.1 Message Creation

In this section the terms "sender" and "user agent" are used informally.

The user agent splits the original content into chunks of 10236

bytes; if the last chunk is shorter, random padding is added. Each chunk has a four-byte length prepended, and the result is called the packet body. If sender anonymity is desired, care should be taken to not include any identifying information (such as headers or unique content from the original plaintext message) in the packets. The content may be compressed before splitting.

The sender next chooses a chain of up to 20 remailers for each packet. Each path is independent, and can be of a different length, but all paths must end at the same remailer. This final remailer is responsible for detecting and discarding duplicate packets, reconstructing the message, and doing the final delivery.

Each packet is next prepared as follows (the full details are in Section 4.3.1). For a chain of "n" remailers, headers "n + 1" through 20 are filled with random data. For headers "n" down to one, the sender generates a symmetric encryption key. This key is used to encrypt the packet body and all the following headers. The key, and other control information, is then encrypted with the public key of the "n"'th remailer in the chain.

The process is repeated, working backward through the chain until the first packet has header information encrypted for the first remailer, and the packet body has been encrypted "n" times. The packet is then sent to the first remailer on its chain.

2.2 Remailing

When a remailer receives a message, it uses its private key to decrypt the first header section. The Packet ID (see Section 4.3.1) can be used to detect duplicates. The integrity of the message is verified by checking the packet length and verifying the message digest in the packet header.

All header sections, as well as the packet body, are decrypted with the symmetric key found in the header. This reveals a public key-encrypted header section for the next remailer.

The first header section is now removed, the others are shifted up, and the last section is replaced with random bytes. Transport encoding is applied to the new message as described in Section 4.4.

In order to prevent an adversary from determining the relationship between incoming and outgoing messages (i.e., traffic analysis), the remailer must collect several encrypted messages before sending the message it has just created; see Section 3.1.

2.3 Message Reassembly

When a packet is sent to the final remailer, it contains an indication that the chain ends at that remailer, and whether the packet contains the complete message or if it is part of a multi-part message. If the packet contains the entire message, the packet body is decrypted and after reordering messages, the plain text is delivered to the recipient. For partial messages, a message ID is used to identify the other parts as they arrive. When all parts have arrived, the message is reassembled, decompressed if necessary, and delivered. A final remailer may discard partial messages if all packets have not been received within a local time limit.

Note that only the final remailer can determine whether packets are part of a specific message. To all of other remailers, the packets appear to be completely independent.

3. Pool Behavior

3.1 Timed Dynamic Pool Mix Algorithm

To obfuscate the link between incoming and outgoing messages, Mixmaster uses a pooling scheme. Messages to be forwarded are stored in a pool. At regular intervals the remailer sends some random messages from the pool to either the next hop or their final recipients.

The pooling scheme is a "Timed Dynamic Pool Mix" [6], which has the following three parameters:

Name	Description
t	Mixing interval
min	Minimum number of messages in the pool
rate	Percentage of messages to be send in one round

The following steps are implemented every "t" seconds:

1. Let "n" be the number of messages currently in the pool.
2. Let "count" be the smaller of "n - min" and "n * rate", or zero if "n - min" is negative.
3. Select "count" messages from the pool at random and send them.

In its default configuration, Mixmaster has a mixing interval of 15 minutes, a minimum pool size of 45 messages, and permits a maximum of 65% of the pool to be sent in one round.

3.2 Dummy Traffic

Dummy messages (see Section 4.1) are multi-hop messages with four randomly selected remailers as the chain. The chain must be selected such that no remailer will appear twice unless two other remailers separate them.

Every time a message is placed in the pool, the remailer chooses a random number from a geometric distribution and creates that many dummy messages which are also placed in the pool.

Similarly, prior to each execution of the mixing algorithm described in Section 3.1, the remailer selects a random number from a different geometric distribution and adds that many dummy messages to the pool as well.

The parameters should be chosen so that on average the remailer creates one dummy for every 32 inbound messages and one every nine mixing rounds.

4. Message Format

4.1 Payload Format

The message payload can be an e-mail message [16], a Usenet message [8], or a dummy message.

Mail and Usenet messages are prefixed with data specifying the payload type. An additional, more restricted method of specifying message header lines is defined for reasons of backward compatibility.

The payload format is as follows:

```
Number of destination fields   [                  1 byte]
Destination fields             [            80 bytes each]
Number of header line fields   [                  1 byte]
Header lines fields            [            80 bytes each]
User data section              [ 2549K -  previous fields ]
```

Each destination field consists of a string of up to 80 ASCII characters, padded with null-bytes to a total size of 80 bytes. The following strings are defined:
null: Dummy message. The remailer will discard the message.
post: Usenet message. The remailer will post the message to Usenet.

post: [newsgroup] Usenet message. The remailer will add a "Newsgroups" header with the specified content, and post the message to Usenet.
[address] E-mail message. The remailer will add a "To" header with the specified content, and send the message as e-mail.

If no destination field is given, the payload is an e-mail message.

Message headers can be specified in header line fields. Each header line field consists of a string of up to 80 ASCII characters, padded with null-bytes to a total size of 80 bytes.

There are three types of user data sections:

- A compressed user data section begins with the GZIP identification header (31, 139). This header contains an additional user data section. The data are compressed using GZIP [RFC 1952]. The GZIP operating system field must be set to Unix, and file names must not be given. Compression may be used if the capabilities attribute of the final remailer contains the flag "C".
- An RFC 2822 user data section begins with the three bytes "##[CR]" (35, 35, 13). It contains an e-mail message or a Usenet message.
- A user data section not beginning with one of the above identification strings contains only the body of the message. When this type of user data section is used, the message header fields must be included in destination and header line fields.

The payload is limited to a maximum size of 2610180 bytes. Individual remailers may use a smaller limit.

Remailer operators can choose to remove header fields supplied by the sender and insert additional header fields, according to local policy; see Section 5.

4.2 Cryptographic Algorithms

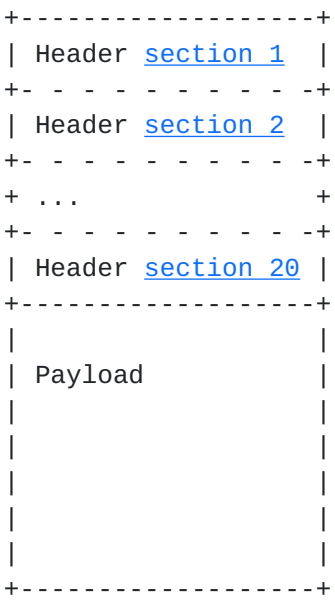
The asymmetric encryption mechanism is RSA with 1024 bit RSA keys and the PKCS #1 v1.5 (RSAES-PKCS1-v1_5) padding format [13]. The symmetric encryption mechanism is EDE 3DES with cipher block chaining (24-byte key, 8-byte initialization vector) [7]. MD5 [9] is used as the message digest algorithm.

4.3 Packet Format

A Mixmaster packet consists of a header containing information for the remailers, and a body containing payload data. To ensure that packets are indistinguishable, the fields are all of fixed size.

The packet header consists of 20 header sections (specified in

Section 4.3.1) of 512 bytes each, resulting in a total header size of 10240 bytes. The header sections (except for the first one) and the packet body are encrypted with symmetric session keys specified in the first header section.



4.3.1 Header Section Format

Packet layout

```
[ Public key ID                 16 bytes ]
[ Length of RSA-encrypted data   1 byte  ]
[ RSA-encrypted session key    128 bytes ]
[ Initialization vector          8 bytes ]
[ Encrypted header part        328 bytes ]
[ Random padding                31 bytes ]
 Total size: 512 bytes
```

To generate the RSA-encrypted session key, a 24-byte Triple-DES key is encrypted with RSAES-PKCS1-v1_5, resulting in 128 bytes (1024 bits) of encrypted data. This Triple-DES key and the initialization vector provided in clear are used to decrypt the encrypted header part. They are not used at other stages of message processing.

The 328 bytes of data encrypted to form the encrypted header part are as follows:

```
[ Packet ID                          16 bytes ]
[ Triple-DES key                     24 bytes ]
[ Packet type identifier              1 byte  ]
[ Packet information     depends on packet type ]
[ Timestamp                           7 bytes ]
[ Message digest                     16 bytes ]
[ Random padding                    as needed ]
 Total size: 328 bytes
```

The fields are defined as follows:

Packet ID: randomly generated packet identifier.

Triple-DES key: used to encrypt the following header sections and the packet body.

Packet type identifier: The type identifiers are:

Value	Type
0	Intermediate hop
1	Final hop, complete message
2	Final hop, partial message

Timestamp: A timestamp is introduced with the byte sequence (48, 48, 48, 48, 0). The following two bytes specify the number of days since January 1, 1970 (00:00 UTC), in little-endian byte order. A random number between one and three, inclusive, may be subtracted from the number of days in order to obscure the origin of the message.

Message digest: MD5 digest computed over the preceding elements of the encrypted header part.

The packet information depends on the packet type identifier, as follows:

Packet type 0 (intermediate hop):

```
  [ 19 Initialization vectors  152 bytes ]
  [ Remailer address            80 bytes ]
```

Packet type 1 (final hop):

```
  [ Message ID                  16 bytes ]
  [ Initialization vector        8 bytes ]
```

Packet type 2 (final hop, partial message):

```
  [ Chunk number                 1 byte  ]
  [ Number of chunks             1 byte  ]
  [ Message ID                  16 bytes ]
  [ Initialization vector        8 bytes ]
```

Initialization vectors: For packet type 1 and 2, the IV is used to symmetrically encrypt the packet body. For packet type 0, there is one IV for each of the 19 following header sections. The IV for the last header section is also used for the packet body.
Remailer address: E-mail address of next hop.
Message ID: Identifier unique to (all chunks of) this message.
Chunk number: Sequence number used in multi-part messages, starting with one.
Number of chunks: Total number of chunks.

In the case of packet type zero, header sections two through twenty, and the packet body, each are decrypted separately using the respective initialization vectors. In the case of packet types one and two, header sections two through twenty are ignored, and the packet body is decrypted using the given initialization vector.

4.3.2 Body Format

The message payload Section 4.1 is split into chunks of 10236 bytes. Random padding is added to the last chunk if necessary. The length of each chunk (not counting the padding), is prepended to the chunk as a four-byte little-endian number. This forms the body of a Mixmaster packet.

A message may consist of up to 255 packets.

4.4 Mail Transport Encoding

Mixmaster packets are sent as standard email messages [16]. The message body has the following format:

```
##
Remailer-Type: Mixmaster [version number]

-----BEGIN REMAILER MESSAGE-----
[packet length ]
[message digest]
[encoded packet]
-----END REMAILER MESSAGE-----
```

The length field always contains the decimal number "20480", since the size of Mixmaster packets is constant. An MD5 message digest [9] of the packet prior to Base-64 encoding is encoded in Base-64.

The packet itself is encoded in Base-64 encoding [10], with line-breaks every 40 characters.

5. Key Format

Remailer public key files consist of a list of attributes and a public RSA key:

[attributes list]

-----Begin Mix Key-----
[key ID]
[length]
[encoded key]
-----End Mix Key-----

The attributes are listed in one line separated by spaces. Individual attributes must not contain whitespace, and are defined as follows:

identifier: A human readable string identifying the remailer
address: The remailer's Internet mail address
key ID: Public key ID
version: Software version number
capabilities: Flags indicating additional remailer capabilities
validity date: Date from which the key is valid
expiration date: Date of the key's expiration

The identifier consists of lowercase alphanumeric characters, beginning with an alphabetic character. The identifier should be no more than eight characters in length.

The key ID is the MD5 message digest of the representation of the RSA public key (not including the length bytes). It is encoded as a hexadecimal string.

The version field consists of the protocol version number followed by a colon and the software version information, limited to the ASCII alphanumeric characters, plus dot (.) and dash (-). All implementations of the protocol specified here should prepend the software version with "2:". Existing implementations lacking a protocol version number imply protocol version 2.

The capabilities field is optional. It is a list of flags represented by a string of ASCII characters. Clients should ignore unknown flags. The following flags are defined:

Flag	Meaning
C	Accepts compressed messages
M	Will forward messages to another mix when used as final hop
Nm	Supports posting to Usenet through a mail-to-news gateway
Np	Supports direct posting to Usenet

The date fields are optional. They are ASCII date stamps in the format YYYY-MM-DD. The first date indicates the date from which the key is first valid; the second date indicates its expiration. If only one date is present, it is treated as the key creation date. (The date stamp implies 00:00 UTC).

The version, capabilities, and date fields must each be no longer than 125 characters.

The encoded key part consists of two bytes specifying the key length (1024 bits) in little-endian byte order, and of the RSA modulus and the public exponent in big-endian form using 128 bytes each, with preceding null bytes for the exponent if necessary. The packet is encoded in Base-64 [10], with line-breaks every 40 characters. Its length (258 bytes) is given as a decimal number.

Digital signatures [14] should be used to ensure the authenticity of the key files.

6. Implementation Notes

This section discusses various implementation issues.

6.1 Remixing

Some remailers may understand multiple remailer protocols. In the interest of creating a unified anonymity set, remailers which speak multiple remailer protocols should attempt to remix messages that use the older protocols whenever possible.

When a remailer receives a message in the older protocol format, it should determine if the message destination is another remailer which also speaks the Mixmaster protocol. If the remailer knows the Mixmaster public key for the next hop, it should process the message normally, but instead of sending the message to its next hop, treat the processed message as opaque data which will comprise the body of a Mixmaster message. The remailer should then create a Mixmaster

message with this body to be delivered to the next hop remailer.

Ensuring that a remailer's keyring contains up to date copies of the public keys for other remailers is the responsibility of the given remailer's operator. Utilities such as Echolot [2] can be used to assist in automating this task.

If the remailer receives a Mixmaster message that, when decrypted, contains a message in an alternate protocol supported by the remailer, it should process the message as though it had initially been delivered in the alternate protocol format.

6.2 Administrative Commands

The existing remailer software understands a number of specific administrative commands. These commands are sent via the Subject: line of an e-mail to the email address of the remailer:

remailer-help: Returns information about using the remailer. The remailer may support a suffix consisting of a dash and a two-letter ISO 639 country code. For example, remailer-help-ar will return a help file in Arabic, if available. Supported languages should be listed at the beginning of the "remailer-help" response.

remailer-key: Returns the remailer's public key as described in Section 5. It may also return the keys and attributes of other remailers it knows about.

remailer-stats: Returns information about the number of messages the remailer has processed per day (again, a day starts at 00:00 UTC).

remailer-conf: Returns local configuration information such as software version, supported protocols, filtered headers, blocked newsgroups and domains, and the attribute strings for other remailers the remailer knows about.

remailer-adminkey: Returns the OpenPGP [14] key of the remailer's operator.

6.3 Dummy Traffic

Older versions of Mixmaster (2.0.4 through 2.9.0) allowed for the creation of dummy message cover traffic, but provided no automated means for introducing this dummy traffic into the system. Beginning in version 3.0, Mixmaster employs an internal dummy policy.

6.4 Key Rotation

Beginning with version 3.0, Mixmaster offers automatic key rotation. Care must be taken to minimize the possibility for partitioning attacks during the key rotation window.

Keys are generated with a validity date and an expiration date. User agents should only display valid keys which have not expired.

Keys are valid for a 13 month period. A remailer must generate a new key when the existing key's expiration date is one month or less in the future. When queried, a remailer must report the most recently generated key as its key, effectively giving each key a 12 month service period.

Remailers must continue to decrypt and process mail encrypted to expired keys for one week past the expiration date on the key. One week after expiration, an expired remailer key should be securely destroyed.

6.5 Delivery of Anonymous Messages

When anonymous messages are forwarded to third parties, remailer operators should be aware that senders might try to supply header fields that indicate a false identity or to send unauthorized Usenet control messages. This is a problem because many news servers accept control messages automatically without any authentication.

For these reasons, remailer software should allow the operator to disable certain types of message headers, and to insert headers automatically.

Remailers usually add a "From:" field containing an address controlled by the remailer operator to anonymous messages. Using the word "Anonymous" in the name field allows recipients to apply scoring mechanisms and filters to anonymous messages. Appropriate additional information about the origin of the message can be inserted in the "Comments:" header field of the anonymous messages.

Anonymous remailers are sometimes used to send harassing e-mail. To prevent this abuse, remailer software should allow operators to block destination addresses on request. Real-life abuse and attacks on anonymous remailers are discussed in [3].

7. Security Considerations

The security of the mix-net relies on the assumption that the underlying cryptographic primitives are secure. In addition, specific attacks on the mix-net need to be considered; [5] contains a more detailed analysis of these attacks.

Passive adversaries can observe some or all of the messages sent to mixes. The users' anonymity comes from the fact that a large number of messages are collected and sent in random order. For that reason

remailers should collect as many messages as possible while keeping the delay acceptable.

Statistical traffic analysis is possible even if single messages are anonymized in a perfectly secure way: an eavesdropper may correlate the times of Mixmaster packets being sent and anonymized messages being received. This is a powerful attack if several anonymous messages can be linked together (by their contents or because they are sent under a pseudonym). To protect themselves, senders must mail Mixmaster packets stochastically independent of the actual messages they want to send. This can be done by sending packets at regular intervals, using a dummy message whenever appropriate. To avoid leaking information, the intervals should not be smaller than the randomness in the delay caused by trusted remailers.

There is no anonymity if all remailers in a given chain collude with the adversary, or if they are compromised during the lifetime of their keys. Using a longer chain increases the assurance that the user's privacy will be preserved, but at the same time causes lower reliability and higher latency. Sending redundant copies of a message increases reliability but may also facilitate attacks. An optimum must be found according to the individual security needs and trust in the remailers.

Active adversaries can also create, suppress or modify messages. Remailers must check the packet IDs to prevent replay attacks. To minimize the number of packet IDs that the remailer must retain, packets which bear a timestamp more than a reasonable number of days in the past may be discarded. Implementors should consider that packets maybe up to three days younger than indicated by the timestamp, and select an expiration value which allows sufficient time for legitimate messages to pass through the network. The number of packet IDs that the remailer must retain can be further minimized by discarding packet IDs for packets encrypted to a key which has expired more than a week in the past.

The use of a link-level encryption protocol with an ephemeral key, such as STARTTLS with SMTP [15], provides for forward secrecy and further aids against replay attacks. Remailer operators should be encouraged to deploy such solutions at the MTA level whenever possible.

Early implementations of Mixmaster did not generate a timestamp packet. Implementors should be aware of the partitioning attack implications if they chose to permit processing of packets without timestamps. Mixmaster versions 2.0.5 and greater in the 2.0.x tree as well as Mixmaster 3.0 in the 3.x tree do not permit processing of such packets.

Message integrity must be verified to prevent the adversary from performing chosen ciphertext attacks or replay attacks with modified packet IDs, and from encoding information in an intercepted message in a way not affected by decryption (e.g. by modifying the message length or inducing errors). This version of the protocol does not provide integrity for the packet body. Because the padding for header section is random, in this version of the protocol it is impossible for a remailer to check the integrity of the encrypted header sections that will be decrypted by the following remailers. Chosen ciphertext attacks and replay attacks are detected by verifying the message digest included in the header section.

The adversary can trace a message if he knows the decryption of all other messages that pass through the remailer at the same time. To make it less practical for an attacker to flood a mix with known messages, remailers can store received messages in a reordering pool that grows in size while more than average messages are received, and periodically choose at random a fixed fraction of the messages in the pool for processing. There is no complete protection against flooding attacks in an open system, but if the number of messages required is high, an attack is less likely to go unnoticed. Additional work has been done in the field of active flooding attack protection; future mix-net protocols may wish to take advantage of this work [4].

If the adversary suppresses all Mixmaster messages from one particular sender and observes that anonymous messages of a certain kind are discontinued at the same time, that sender's anonymity is compromised with high probability. There is no practical cryptographic protection against this attack in large-scale networks. The effect of a more powerful attack that combines suppressing messages and re-injecting them at a later time is reduced by using timestamps.

Manipulation of the distribution of remailer keys, capabilities, and statistics can lead to powerful attacks against a remailer network. Sensitive information such as this should be distributed in a secure manner.

The lack of accountability that comes with anonymity may have implications for the security of a network. For example, many news servers accept control messages automatically without any cryptographic authentication. Possible countermeasures are discussed in Section 6.5.

8. Acknowledgments

Several people contributed ideas and source code to the Mixmaster v2

software.

"Antonomasia" <ant@notatla.org.uk>, Adam Back <adam@cypherspace.org>, Marco A. Calamari <marcoc@dada.it>, Colin Tuckley <colin@tuckley.org>, Bodo Moeller <bmoeller@acm.org>, and Jon Orbeton <jono@networkcommand.com> suggested improvements to this document. Rich Salz <rsalz@datapower.com> contributed significantly to this document by XMLifying it and rewording many ambiguous sections. Myles Conley <miles@tenhand.com> also reworded many ambiguous sections and contributed important suggestions for improving clarity.

9 References

[1] Chaum, D., "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", Communications of the ACM 24:2, 1981.

[2] Palfrader, P., "Echolot: A Pinger for Anonymous Remailers", 2003, <http://http://www.palfrader.org/echolot/>.

[3] Mazieres, D. and F. Kaashoek, "The Design, Implementation and Operation of an Email Pseudonym Server", 1998, <ftp://cag.lcs.mit.edu/pub/dm/papers/mazieres:pnym.ps.gz>.

[4] Danezis, G. and L. Sassaman, "Heartbeat Traffic to Counter (n-1) Attacks", October 2003, <http://www.cl.cam.ac.uk/users/gd216/p125_danezis.pdf>.

[5] Moeller, U., "Anonymisierung von Internet-Diensten", January 1998, <http://agn-www.informatik.uni-hamburg.de/people/3umoelle/st.ps>.

[6] Serjantov, A., Dingledine, R. and P. Syverson, "From a Trickle to a Flood: Active Attacks on Several Mix Types", October 2002, <http://freehaven.net/doc/batching-taxonomy/taxonomy.pdf>.

[7] Menezes, A., van Oorschot, P. and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[8] Horton, M. and R. Adams, "Standard for interchange of USENET messages", RFC 1036, December 1987.

[9] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[10] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, February 1993.

[11] Deutsch, P., Gailly, J-L., Adler, M., Deutsch, L. and G. Randers-Pehrson, "GZIP file format specification version 4.3", RFC 1952, May 1996.

[12] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L. and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, March 1998.

[13] Kaliski, B. and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0", RFC 2437, October 1998.

[14] Callas, J., Donnerhacke, L., Finney, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[15] Hoffman, P., "SMTP Service Extension for Secure SMTP over TLS", RFC 2487, January 1999.

[16] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

Authors' Addresses

Ulf Moeller
Secardeo GmbH
Betastr. 9a
85774 Unterfoehring
Germany

EMail: ulf.moeller@secardeo.com

Lance Cottrell
Anonymizer, Inc.
5694 Mission Center Road #426
San Diego, CA 92108-4380
USA

EMail: loki@infonex.com

Peter Palfrader
The Mixmaster Project
Hoettinger Auffahrt 1
6020 Innsbruck
Austria

EMail: peter@palfrader.org
URI: http://www.palfrader.org/

Len Sassaman
Nomen Abditum Services
P.O. Box 900481
San Diego, CA 92190-0481
USA

EMail: rabbi@abditum.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.