Internet-Draft
Expires: August 31, 2004

Ulf Moeller

Lance Cottrell
Anonymizer Inc.
Peter Palfrader
The Mixmaster Project
Len Sassaman
Nomen Abditum Services
February 2004

Mixmaster Protocol Version 2
draft-sassaman-mixmaster-00.txt

Status of This Memo

Copyright Notice

Abstract

Most e-mail security protocols only protect the message body, leaving useful information such as the 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 D. Chaum's 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. 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 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 [Chaum 1981] allows one to send messages while hiding the relation of sender and recipient from observers (unobservability). It also provides the sender of a message with the ability 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 over the protocol used in Mixmaster. The mixing algorithm is specified in section 3, and the message format is specified in section 4.

2.1 Message Creation

To send a message, the user agent splits it into parts of fixed size, which form the bodies of Mixmaster packets. If sender anonymity is desired, care should be taken not to include identifying information in the message. The message may be compressed.

The sender chooses a sequence of up to 20 remailers for each packet. The final remailer must be identical for all packets.

The packet header consists of 20 sections. For a sequence of n remailers, header sections n+1, ... , 20 are filled with random data. For each section i := n down to 1, the sender generates a symmetric encryption key, which is used to encrypt the body and all following header sections. This key, together with other control information for the remailer, is included in the i-th header section, which is then encrypted with the remailer's public key. The resulting message is sent to the first remailer in an appropriate transport encoding.

To increase reliability, redundant copies of the message may be sent through different paths. The final remailer must be identical for all paths, so that duplicates can be detected and the message is delivered only once.

2.2 Remailing

When a remailer receives a message, it decrypts the first header section with its private key. By keeping track of a packet ID, the remailer verifies that the packet has not been processed before. The

integrity of the message is verified by checking the packet length and verifying message digests included in the packet. Then the first header section is removed, the others are shifted up by one, and the last section is filled with random padding. All header sections and 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 at the top, and removes the old top header section. Transport encoding is applied to the resulting message.

The remailer collects several encrypted messages before sending the resulting messages in random order. Thus the relation between the incoming and outgoing messages is obscured to outside adversaries even if the adversary can observe all messages sent. The message is effectively anonymized by sending it through a chain of independently operated remailers.

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 a complete message or 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. If the parts do not arrive within a time limit, the message is discarded.

Only the last remailer in the chain can determine whether packets are part of a certain message. To all the others, they are completely independent.

2.4 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.

If the remailer receives a Mixmaster message that, when decrypted, reveals a message of another protocol which the remailer speaks, it should process the

message as though it had been delivered in the other protocol format initially.

3. Pool Behavior

3.1 Timed Dynamic Pool Mix Algorithm

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

The pooling scheme used in Mixmaster is a "Timed Dynamic Pool Mix", which has the following three parameters:

t	mixing interval
min	minimum number of messages in the pool
rate	percentage of messages to be send in one round

Every t seconds:

- Let n be the number of messages currently in the pool
- count is the smaller of (n - min) and (n * rate), or 0 if (n - min) is negative.
- Select count random messages from the pool and send them.

3.2. Dummy Traffic

"Dummy messages" are multi-hop messages of type NULL with four randomly selected nodes as their chain. Chains are selected such that no node will appear twice in the chain unless separated by two other nodes in the chain.

Older versions of Mixmaster (2.0.4 - 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.

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 distributions' parameters are chosen so that on average the remailer creates one dummy for every 32 messages coming in and one every 9 mixing rounds.

4. Message Format

4.1 Payload Format

The Mixmaster message payload can be an e-mail message, a Usenet message or a dummy message.

The messages use the formats specified in [RFC 822] and [RFC 1036] respectively, prepended 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              [ up to ~2.5 MB]
```

Each destination field consist 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
  post:                Usenet message
  post: [newsgroup]    Usenet message
  [address]            e-mail message
```

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

If the destination field is "post: [newsgroup]", a "Newsgroups: [newsgroup]" field is added to the header of the resulting message. If the destination field is of the fourth type, a "To: [address]" field is added to the header of the resulting message. [address] and [newsgroup] are strings of ASCII characters. If the message is a dummy message the node should discard it.

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). It contains another 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 822 user data section begins with the identification "##<CR>" (35, 35, 13). It contains an e-mail message or a Usenet message as

specified in [RFC 822] and [RFC 1036]. This type cannot be used if the final remailer uses a Mixmaster software version prior to 2.0.4.

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 maximal 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 operation in Mixmaster version 2 uses RSA with 1024 bit RSA keys and the PKCS #1 v1.5 (RSAES-PKCS1-v1_5) padding format [RFC 2437]. The symmetric encryption uses EDE 3DES with cipher block chaining (24 byte key, 8 byte initialization vector) [Schneier 1996]. MD5 [RFC 1321] 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 size of these encrypted data fields is fixed.

The packet header consists of 20 header sections (specified in section 3.3.1) 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

```
     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]
     Padding                      [  31 bytes]
```

Total size: 512 bytes

To generate the RSA-encrypted session key, a random 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               [fill to 328 bytes]
```

The possible packet type identifiers are:

```
 Intermediate hop           0
 Final hop                  1
 Final hop, partial message  2
```

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]
```

Packet ID: randomly generated packet identifier.

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

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: randomly generated identifier unique to (all chunks of) this message.

Chunk number: Sequence number used in multi-part messages, starting with 1.

Number of chunks: Total number of chunks.

Timestamp: A timestamp is introduced with the byte sequence (48, 48, 48, 48, 0). The following two bytes specify the number of days since Jan 1, 1970, given in little-endian byte order. A random number of up to 3 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.

In the case of packet type 0, header sections 2 .. 20 and the packet body each are decrypted separately using the respective initialization vectors. In the case of packet types 1 and 2, header sections 2 .. 20 are ignored, and the packet body is decrypted using the given initialization vector.

4.3.2 Body Format

The message payload (section 3.1) is split into chunks of 10236 bytes. To each chunk, its length is prepended as a 4 byte little-endian number to form 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 text messages [RFC 822]. The RFC 822 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 [RFC 1321] of the (un-encoded) packet is encoded in base64.

The packet itself is encoded in base 64 encoding [RFC 1421], broken into lines of 40 characters (except that the last line is shorter).

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:

identifier:	a human readable string identifying the remailer
address:	the remailer's Internet mail address
key ID:	public key ID
version:	the Mixmaster 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 consist of no more than eight characters in length.

The encoded key packet 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 [RFC 1421], and broken into lines of 40 characters each (except that the last line is shorter). Its length (258 bytes) is given as a decimal number.

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 this protocol should prepend the software version with "2:" to indicate Mixmaster protocol version 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 used in version 2.0.4 through 3.0:

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 introduced in version 3.0 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).

Digital signatures [RFC 2440] should be used to ensure the authenticity of the key files.

5.1 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. Mixmaster clients only display keys which are currently valid and not expired.

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

Remailers 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. Administrative Commands

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

```
remailer-help:     returns information on using the remailer
remailer-key:      returns the remailer's public key
remailer-stats:    returns information on the number of messages processed
remailer-conf:     returns local configuration information
remailer-adminkey: returns the public key of the remailer operator
```

Upon receiving an administrative request, the remailer returns its reply to the email address specified in the Reply-To: or From: header of the request.

The remailer-help request should return basic information on using the remailer. Remailers may also accept remailer-help requests with an ISO 639 two-letter language code appended following a hyphen. 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.

The remailer-key request returns the remailer key as described above. It may also return keys and capability information for other remailer protocols supported by the remailer.

The remailer-stats request returns statistics on the number of messages processed per day by the remailer.

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

The remailer-adminkey request returns a public OpenPGP key for use in communication with the remailer operator.

7. 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 Usenet control messages [RFC 1036] unauthorized, which 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.

If the recipient does not wish to receive anonymous messages, unobservability of communications and authenticity can be achieved at the same time by the remailer verifying that the message is cryptographically signed [RFC 2440] by a known sender.

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 [Mazieres 1998].

8. 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 ([Moeller 1998] 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 in 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 in 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 allowsd 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.

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.

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.

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.

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 7.

9. Acknowledgments

Several people contributed ideas and source code to the Mixmaster v2 software. "Antonomasia" <ant@notatla.demon.co.uk>, Adam Back <adam@cypherspace.org> and Bodo Moeller <bmoeller@acm.org> suggested improvements to this document.

10. References

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

[Mazieres 1998] Mazieres, D., and Kaashoek, F., "The Design, Implementation and Operation of an Email Pseudonym Server",
5th ACM Conference on Computer and Communications Security, 1998.
<URL: ftp://cag.lcs.mit.edu/pub/dm/papers/mazieres:pnym.ps.gz>.

[Moeller 1998] Moeller, U., "Anonymisierung von Internet-Diensten",

Studienarbeit, University of Hamburg, January 1998.
<URL: http://agn-www.informatik.uni-hamburg.de/people/3umoelle/st.ps>.

[RFC 822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[RFC 1036] Horton, M., and Adams, R., "Standard for Interchange of USENET Messages", RFC 1036, December 1987.

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

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

[RFC 1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

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

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

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

[Schneier 1996] Schneier, B., "Applied Cryptography", 2nd Edition, Wiley, 1996.

11. Authors' Addresses

Ulf Moeller

EMail: mail@ulfm.de

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

EMail: loki@infonex.com

Peter Palfrader

EMail: peter@palfrader.org

Len Sassaman
Nomen Abditum Services
P.O. Box 99282
Emeryville, CA 94662-9282

EMail: rabbi@abditum.com