Mesh/Recrypt: Usable Confidentiality

Mesh/Recrypt

<series>draft-hallambaker-mesh-recrypt

<status>informational

<stream>independent

<ipr>trust200902

<author>Phillip Hallam-Baker

<surname>Hallam-Baker

<initials>P. M.

<firstname>Phillip

<organization>Comodo Group Inc.

<email>[philliph@comodo.com](mailto:philliph@comodo.com)

<also>http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html

A messaging infrastructure providing full end-to end security is presented. Unlike existing approaches such as S/MIME and OpenPGP, Mesh/Recrypt uses proxy re-encryption to preserve full end-to-end security with individual user and device keys in situations such as the user having multiple decryption devices and messages being set to mailing lists.

This document shows the use of Mesh/Recrypt to address the principle use cases Mesh/Recrypt is designed to address. These include asynchronous messaging such as mail and controlled documents and synchronous messaging applications such as chat, voice and video.

# Introduction

Traditional messaging security infrastructures are difficult to configure, difficult to use and limited to one mode of communication. Digital certificates are hard to obtain and harder to maintain. Managing a Web of Trust requires a very high level of user competence. S/MIME and OpenPGP offer end-to-end email security but not streaming services such as video, voice or chat.

In recent years a number of proprietary chat systems have been extended to the point that a single application and protocol supports chat, voice, video and asynchronous communication modes such as messaging and file transfer. While such systems typically claim to offer cryptographic security, the extent to which this is achieved is difficult to determine. Even systems purporting to offer ‘end-to-end’ security have proved to be woefully inadequate when it is discovered that one of the ‘ends’ referred to is in fact the messaging infrastructure operated by the provider.

A key limitation of all the deployed messaging systems that were reviewed in the development of this paper is that true end-to-end confidentiality is only achieved for a limited set of communication patterns. Specifically, bilateral communications (Alice sends a message to Bob) or broadcast communications to a known set of recipients (Alice sends a message to Bob, Carol and Doug). These capabilities do not support communication patterns where the set of recipients changes over time or is confidential. Yet such requirements commonly occur in situations such as sending a message to a mailing list whose membership isn’t known to the sender, or creating a spreadsheet whose readership is to be limited to authorized members of the ‘accounting’ team.

<figuresvg="../Images/Recrypt-static.svg">Traditional End-to-End Encryption is static.

Mesh/Recrypt is an experimental messaging infrastructure that applies proxy re-encryption to support all the commonly used messaging modes with strong end-to-end encryption. The primary purpose of Mesh/Recrypt is to demonstrate the advantages of using the proxy re-encryption technique and to determine the feasibility of retrofitting such capabilities to legacy protocols such as SMTP, IMAP and XMPP.

<figuresvg="../Images/Recrypt-dynamic.svg">Mesh Recrypt supports End-to-End Encryption in dynamic groups.

Whether the advantages of building on an established base outweigh those of a clean slate approach for purposes of deployment are currently unknown, but there are clear advantages of using a clean slate approach for purposes of exposition.

As the name suggests, Mesh/Recrypt makes use of the Mathematical Mesh infrastructure for management of user keys. For clarity and convenience, this document describes the application of Mesh/Recrypt to a completely new protocol suite. Strategies for adding similar capabilities to existing specifications are discussed as possible future work.

# Definitions

This section presents the related specifications and standards on which Mesh/Recrypt is built, the terms that are used as terms of art within the documents and the terms used as requirements language.

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in <norm="RFC2119"/>.

## Defined Terms

The following terms are used as terms of art in this document with the meaning specified below:

Confidential Document Control (CDC)

An Access Control mechanism that uses cryptography to control read access to static content (typically documents) within its control.

Controlled Document

Content that is subject to control of a CDC system.

Proxy Re-Encryption

A cryptography mechanism that permits a party that does not have the ability to decrypt an encrypted message to transform it into a message that can be decrypted under a different private key than the original.

Recryption

The term ‘recryption’ is used as a synonym for Proxy Re-Encryption in this document.

Recryption Key

A cryptographic key that is used to enable a different party to decrypt an encrypted message that does not grant decryption capability.

## Related Specifications

The related specifications used in the Mesh/Recrypt protocol are described in the Mesh Architecture specification <norm="draft-hallambaker-mesh-architecture"/>

## Implementation Status

The implementation status of the reference code base is described in the companion document <info="draft-hallambaker-mesh-developer"/>.

# Proxy Re-Encryption

Proxy re-encryption provides a technical capability that meets the needs of such communication patterns. Conventional symmetric key cryptography uses a single key to encrypt and decrypt data. Public key cryptography uses two keys, the key used to encrypt data is separate from the key used to decrypt. Proxy re-encryption introduces a third key (the recryption key) that allows a party to permit an encrypted data packet to be decrypted using a different key without permitting the data to be decrypted.

The introduction of a recryption key permits end-to-end confidentiality to be preserved when a communication pattern requires that some part of the communication be supported by a service.

The introduction of a third type of key, the recryption key permits two new roles to be established, that of an administrator and recryption service. There are thus four parties:

Administrator

Holder of Decryption Key, Creator of Recryption Keys

Sender

Holder of Encryption Key

Recryption Service

Holder of Recryption keys

Receiver

Holder of personal decryption key

The communication between these parties is shown in Figure X below:

<figuresvg="../Images/Recrypt-parties.svg">Mesh/Recrypt Parties

The chief advantage of recryption is that the recryption service does not have the ability to decrypt messages and does not need to be trusted at the same level as a recipient. A recryption service may be implemented as a cloud service on an untrusted host or managed in house by a system administrator who is only partially trusted.

## Proxy Re-Encryption Algorithms

Proxy Re-Encryption was introduced by Blaze et. al. <norm="Blaze98"/> in 1998. In this paper, we make use of the Diffie Hellman based mechanism described in this paper. While this approach does not have capabilities such as reversibility or transitivity offered in later work, such features do not appear to offer any practical advantages in developing protocols for the intended applications and may well introduce significant disadvantages.

The use of the Diffie Hellman based approach has the considerable advantages of being compatible with the recently developed CFRG Elliptic Curve algorithms and being minimally unencumbered by IPR claims.

Recall that in the Diffie Hellman key agreement algorithm, shared parameters e and p are generated, these being an exponent value (e) and a modulus value (p). To create a shared key, two parties (Alice and Bob) generate private keys a, b being positive integers in the interval [2 ... p-1]. The corresponding public keys are then ea mod p and eb mod p. Thus, knowledge of either {eb mod p, a} or {ea mod p, b} is sufficient to calculate the shared secret value s = eab mod p.

<figuresvg="../Images/Recrypt-traddh.svg">Traditional Diffie-Hellman

When applying Diffie Hellman to a messaging protocol, it is typically desirable to ensure that a unique shared value is created for each exchange. If the protocol only requires authentication of the receiver, the sender may ensure that each shared value is unique by generating a new key pair {t, et mod p} for each exchange. Alternatively, mutual authentication may be preserved if the shared secret is formed from three values s = eabt mod p, where a and b are the validated public keys of the sender and receiver and t is a temporary key generated by the sender that has a nonce-like function.

To adapt Diffie Hellman to a recryption mechanism, we note that just as the value s = ebt mod p may be calculated as either (eb mod p)t mod p or (et mod p)b mod p, it can also be calculated as ((et mod p)b-x mod p . (et mod p)x mod p) mod p. This equivalence is used to create the recryption protocol.

Figure XX shows Bob calculating the shared secret with the aid of a Recryption service. Bob's private key for decryption is now x and the Recryption service has the corresponding recryption key b-x. The recryption service can provide Bob with the additional information needed to decrypt the message but cannot decrypt the message itself.

<figuresvg="../Images/Recrypt-recryptdh.svg">Diffie-Hellman with Recryption

Applying this approach to Proxy Re-Encryption directly is unacceptable since the administrator of the recryption group must know Bob's private key. To avoid this problem, the administrator generates a new public key pair for each member of the group and encrypts the decryption portion under the public key of the member.

In the following example, Alice is the administrator of the recryption group and Bob and Carol are recipients.

Bob

Generates a public key encryption pair (b, B). The algorithm used for this does not matter, as the only functions used are encryption and decryption.

Bob publishes his public key B.

Alice (Administrator)

Generates public key pair {a, ea mod p}.

Publishes the public key value for the recryption group ea mod p

To enable Bob to receive messages, Alice generates a recryption keypair for Bob {a-bx, bx } and encrypts the decryption key (bx) using Bob’s public key (Bpub) to create a recryption entry for Bob {a- bx, E(bx, Bpub)}.

The recryption entry is sent to the recryption service.

At this point Alice, Bob and the Recryption Service have the information they need to receive encrypted messages (figure X).

<figuresvg="../Images/Recrypt-protocol-init.svg">Mesh/Recrypt Administration Protocol

Having established the necessary keying material, Carol (or any other party who knows the recryption group encryption key) can encrypt a message:

Carol (Sender)

Generates a temporary key pair {t, et mod p} and uses this and the public key of the recryption group (ea mod p) to create a shared secret s = eat mod p that is used to encrypt the message.

Sends the encrypted message and temporary public key (et mod p) to the recryption service

Recryption Service

Receives the message and retrieves the list of intended recipients, this currently has just a single entry for Bob {a-bx, E(bx, B)}

Calculates (et mod p)a-bx mod p = eta-tbx mod p

Sends the encrypted message, the original temporary public key generated by Carol (et mod p), the recryption value eta-tbx mod p and the encrypted decryption key E(bx, B) to Bob.

Bob

Receives the message

Decrypts the E(bx, B) using his private key b to obtain bx

Uses bx and et mod p to calculate etbx mod p

Calculates (eta-tbx mod p. etbx mod p) mod p = eta mod p = s

Uses s to decrypt the message

This protocol is illustrated in figure X:

<figuresvg="../Images/Recrypt-protocol.svg">Mesh/Recrypt Decryption Protocol

Note that Alice is not a participant in the recryption protocol. Administrator actions are only required when adding or removing recipients to the recryption group.

Alice can add additional recipients to the group at any time by creating a recryption pair, encrypting the decryption key under the new user's public key and sending the information to the recryption service, just like she did for Bob.

Alice can remove a user from the recryption group by telling the recryption service to no longer recrypt messages to the removed user’s recryption key. This requires the recryption service to be trusted not to forward messages to the deleted user. To restore the untrusted status of the recryption service it is necessary for the administrator to create a new encryption key and a full set of recryption keys for the continuing users.

One major limitation in the trust model of the recryption scheme described is that while it is not possible for either the recryption service or individual recipients to decrypt arbitrary messages the recryption service and a recipient may do so if they collude. This particular limitation in the trust model is an inescapable consequence of the fact that the function of the recryption service is to enable a recipient to decrypt a message and cannot be avoided without introducing additional parties. This limitation is not considered to be a serious limitation for the intended application.

## Applying Mesh/Recrypt

This document describes the Mesh/Recrypt algorithm and protocol. To make use of the capability it provides, it is necessary to make use of it in an application protocol. Mesh/Recrypt MAY be used in any application that supports data level encryption. This includes mailing lists, conferencing systems offering voice or chat and confidential document control.

## Mailing Lists

One of the earliest uses proposed for recryption is to support end-to-end security for a confidential mailing list in which the membership of the list is not disclosed to its members. In this application, the mail server is a recryption service and trusted to maintain the confidentiality of the mailing list membership but not the messages themselves. This offers many advantages over existing approaches:

* Messages are encrypted end-to-end
* It is not necessary for senders to know the membership of the list.
* New members added to the list can read messages sent before they joined.

To apply recryption to a mailing list server, a recryption keyset is created for each mailing list managed by the server and the administrator responsible for maintaining the membership of the list is also the administrator of the corresponding recryption key set.

## Chat rooms and other streaming data.

The application of recryption to a chat room application is similar to the mailing list application except that the administrator may be either an offline party as before or a participant in the conversation. In the latter case, the protocol should permit the administrator to pass their role to another participant should they need to leave.

One major constraint on the use of recryption to support streamed audio or video is that since the messaging service cannot decrypt the data stream, it can hardly be expected to perform transcoding services such as producing lower resolution versions of a video stream to support participants with low bandwidth connections. Either all the participants must receive the exact same data feed or transcoding services must be provided by a trusted party granted access by the administrator.

## Confidential Document Control

Confidential Document Control (CDC) uses cryptography to enforce access control. Unlike Digital Rights Management and related technologies, CDC only provides a means to permit or deny access to confidential data while it is under protection. A CDC infrastructure does not attempt to control the use made of that data by an authorized recipient, in particular, a CDC infrastructure does not necessarily prevent redistribution of data by a party permitted to read it.

The application of recryption to CDC maps naturally to the use of ‘security labels’ to control access to confidential documents in government and military applications. Each security label (e.g. secret#example.com) has an associated recryption key set. The administrator of the recryption key set is responsible for managing the parties authorized to read documents controlled under that label.

Recryption may be used to support the use of multiple labels. Combining appropriate cryptographic operations permits a document author to require recipients to be granted access for all the labels specified or for any of the labels specified. For example, the designation (Accounting#example.com + Executive#example.com) might indicate that a recipient must be a member of the Accounting and Executive teams while the designation (Accounting#example.com | Executive#example.com) would enable members of either team to read the material.

While the recryption algorithm used in Mesh/Recrypt allows the use of conjunctions and disjunctions to implement the equivalent of an ACL entry granting access, it is not possible to implement the equivalent of an ACL entry denying access to a group of users. The recryption service can be instructed to refuse recryption to a group of users but this restriction is not cryptographically enforced.

Since users must request a recryption key from the recryption service for each document accessed, the recryption service is a Policy Control Point and is thus potentially a point at which additional accountability and/or access controls may be introduced. An enterprise recryption service might maintain a log of all access requests from users and restrict access to users whose requests exceed some form of quota. Attempts to access particularly sensitive documents might raise flags requiring review by a supervisor.

## Multiple Devices

When the S/MIME and OpenPGP email encryption schemes were developed in the 1990s the machines of the day, if movable at all were ‘portable’ rather than ‘mobile’. Contemporary users demand access to their communications applications from a wide variety of devices including desktops, laptops, tablets, phones and even watches. The need for a single user to access their email on multiple devices is now the norm rather than the exception.

Use of multiple devices and in particular mobile devices introduces obvious security concerns. A device may be lost or stolen; a machine may be sold without destroying data stored on it. Such circumstances very frequently result in disclosure of private keys to an attacker. Maintaining separate private keys on each device allows the consequences of such loss to be mitigated and further compromise prevented.

To apply recryption to this use case, the email recipient establishes a personal recryption keyset on a machine that they consider at least risk of compromise. A separate recryption key entry is then created for each device and the recryption keyset uploaded to a suitable recryption server host (e.g. the presence service of a chat application, inbound mail server, etc.)

One difficulty that arises in this approach is that while a non-transitive recryption mechanism can be applied in either a sender side context such as a mailing list or a receiver side context such as supporting multiple devices, enabling the use of both at the same time requires additional effort.

<include="..\Generated\ExamplesRecrypt.md">

<include="..\Generated\SchemaRecrypt.md">

# Acknowledgements

# Security Considerations

[This is just a sketch for the present.]

# IANA Considerations

[TBS list out all the code points that require an IANA registration]