Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 17, 2018

P. Hallam-Baker
Comodo Group Inc.

August 16, 2017

Mesh/Recrypt: Usable Confidentiality
draft-hallambaker-mesh-recrypt-02

Abstract

independent

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.

This document is also available online at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html .

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 17, 2018.

Table of Contents

1. 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.

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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.

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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.

2. 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.

2.1. Related Specifications

The related specifications used in the Mesh/Recrypt protocol are described in the Mesh Architecture specification [draft-hallambaker-mesh-architecture] [draft-hallambaker-mesh-architecture]

2.2. Defined Terms

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

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

Content that is subject to control of a CDC system.

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.

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

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

2.3. 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 [RFC2119] [RFC2119] .

3. 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:

Holder of Decryption Key, Creator of Recryption Keys

Holder of Encryption Key

Holder of Recryption keys

Holder of personal decryption key

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

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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.

3.1. Proxy Re-Encryption Algorithms

Proxy Re-Encryption was introduced by Blaze et. al. [Blaze98] [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.

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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.

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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.

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.

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 (pubB) to create a recryption entry for Bob {a- bx, E(bx, pubB)}.

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).

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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:

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

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.

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:

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-recrypt.html.]]

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.

3.2. 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.

3.3. 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.

3.4. 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.

3.5. 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.

3.6. 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.

4. Mesh/Recrypt/Admin Service

The Mesh/Recrypt administration service supports transactions to Add and Delete members from a group and to list all the members in a group.

_recrypt._tcp

/.well-known/recrypt

Every Recrypt Service transaction consists of exactly one request followed by exactly one response.

Mesh Service transactions MAY cause modification of the data stored in the Mesh Portal or the Mesh itself but do not cause changes to the connection state. The protocol itself is thus idempotent. There is no set sequence in which operations are required to be performed. It

is not necessary to perform a Hello transaction prior to a CreateGroup, AddMember or any other transaction.

4.1. Request Messages

A Mesh/Recrypt administration Service request consists of a payload object that inherits from the MeshRequest class. When using the HTTP binding, the request MUST specify the portal DNS address in the HTTP Host field.

4.1.1. Message: RecryptRequest

Base class for all request messages.

[None]

4.2. Response Messages

A Mesh/Recrypt administration Service response consists of a payload object that inherits from the MeshResponse class. When using the HTTP binding, the response SHOULD report the Status response code in the HTTP response message. However the response code returned in the payload object MUST always be considered authoritative.

4.2.1. Message: RecryptResponse

Base class for all response messages. Contains only the status code and status description fields.

[None]

4.3. Imported Objects

The Recrypt Administration Sercice makes use of JSON objects defined in the JOSE Signatgure and Encryption specifications.

4.4. Common classes

The following classes are referenced at multiple points in the protocol.

4.4.1. Structure: RecryptionGroup

Describes a group of recryption users.

String (Optional)

A user friendly account name in RFC821 format (user@example.com).

MemberEntry [0..Many]

Member of a recryption group

PublicKey [0..Many]

The set of past encryption keys associated with the group.

PublicKey (Optional)

The current group encryption key

4.4.2. Structure: MemberEntry

Describes a member of a recryption group

String (Optional)

UDF fingerprint of the user's master profile

String (Optional)

User friendly account name

String [0..Many]

A list of privileges assigned to the user.

Currently defined privileges are RECRYPT, ADMIN and SUPER. Recrypt grants a user the right to request decryption of data encrypted under the group key. ADMIN grants the right to add or remove users from the group. SUPER grants the right to add or remove administrators and super-administrators from the group.

Note that being granted the necessary privilege does not in itself confer the ability to decrypt messages as this requires access to the master private key.

String [0..Many]

A list of quotas assigned to the user.

Each quota is described by the UDF fingerprint of the quota service.

String (Optional)

Member status. Valid values are ACTIVE, REVOKED and SUSPENDED.

Once added to a recryption group, a user can never be 'deleted'. Instead their member record is marked as REVOKED or SUSPENDED which causes the recryption service to refuse further recryption requests.

Note that it is entirely valid for newly created recryption group to contain member entries that are inactive. This allows a key administrator to generate key material for group members in anticipation of them requiring access without immediately granting that access.

UserDecryptionEntry [0..Many]

Identifier of

4.4.3. Structure: UserDecryptionEntry

Decryption entry for a particular user and group key

String (Optional)

Fingerprint of the encryption key to which this recryption data corresponds

String (Optional)

Fingerprint of the user's key

String (Optional)

A user friendly account name in RFC821 format (user@example.com).

PublicKey (Optional)

The recryption key to be used to recrypt for this user.

JoseWebEncryption (Optional)

The user's decryption key encrypted under a one or more encryption keys held by the user. The encrypted content is a PrivateKey structure specifying the decryption key for the user.

4.4.4. Structure: CombinedToGroup

Glue that maps a combined key identifier (Encryption, Member) to a group and member entry.

String (Optional)

Fingerprint of the encryption key to which this recryption data corresponds

String (Optional)

Fingerprint of the user's key

String (Optional)

UDF fingerprint of the user's master profile

String (Optional)

A user friendly account name in RFC821 format (user@example.com).

4.5. Administrator Transactions

4.6. Transaction: Hello

Request: HelloRequest

Response: HelloResponse

Report service and version information.

The Hello transaction provides a means of determining which protocol versions, message encodings and transport protocols are supported by the service.

4.7. Transaction: CreateGroup

Request: CreateGroupRequest

Response: CreateGroupResponse

Create a new recryption group.

4.7.1. Message: CreateGroupRequest

- Inherits: RecryptRequest

Request creation of a recryption group. The only request parameter describes the group to be created.

RecryptionGroup (Optional)

The Recryption Group to create

4.7.2. Message: CreateGroupResponse

- Inherits: RecryptResponse

Reports the success or failure of a CreateGroup request. The operation either succeeds or fails, there are no returned parameters

[None]

4.8. Transaction: UpdateGroup

Request: UpdateGroupRequest

Response: UpdateGroupResponse

Update the information describing a recryption group.

4.8.1. Message: UpdateGroupRequest

- Inherits: RecryptRequest

Request an update to a recryption group.

Note that the update process is currently limited to 'strike and replace'. This is likely to become cumbersome if groups with very large numbers of entries are being maintained. It is likely that a future version of the protocol will support update requests that implement commonly occurring tasks such as updates to add a new encryption key, etc.

RecryptionGroup (Optional)

The Recryption Group to create

4.8.2. Message: UpdateGroupResponse

- Inherits: RecryptResponse

Reports the success or failure of a UpdateGroup request. The operation either succeeds or fails, there are no returned parameters

[None]

4.9. Transaction: AddMember

Request: AddMemberRequest

Response: AddMemberResponse

Add a member or members to an existing recryption group.

4.9.1. Message: AddMemberRequest

- Inherits: RecryptRequest

  String (Optional)

The UDF fingerprint of the recryption group to add the member to.

  MemberEntry [0..Many]

Describes the member(s) to add

4.9.2. Message: AddMemberResponse

- Inherits: RecryptResponse

Reports the success or failure of a AddMember request. The operation either succeeds or fails, there are no returned parameters

[None]

4.10. Transaction: UpdateMember

Request: UpdateMemberRequest

Response: UpdateMemberResponse

Update a one or more member entries

This transaction may be used to make member entries inactive by posting REVOKED or SUSPENDED status to their member entry.

4.10.1. Message: UpdateMemberRequest

- Inherits: RecryptRequest

  String (Optional)

The UDF fingerprint of the recryption group in which the member entries is to be updated

  MemberEntry [0..Many]

Describes the member(s) to add

4.10.2. Message: UpdateMemberResponse

o Inherits: RecryptResponse

Reports the success or failure of a UpdateMember request. The operation either succeeds or fails, there are no returned parameters

[None]

4.11. Future work

At present the protocol does not provide a mechanism for modifying administrator privileges or requesting statistics on use of recryption services. These are obviously important. Whether these should be part of the base protocol or a separate protocol is another matter.

5. User Service

The only transaction supported by the user facing service at this point is the ability to request a recryption operation.

5.1. Transaction: RecryptData

Request: RecryptDataRequest

Response: RecryptDataResponse

Request that the service provide a recryption result for the specified encrypted data and return it encrypted under the user's public key.

5.1.1. Message: RecryptDataRequest

o Inherits: RecryptRequest

Request that the service provide a recryption result for the specified encrypted data and return it encrypted under the user's public key.

String (Optional)

The recryption group in which the member entries is to be updated

Recipient (Optional)

The Jose Web Encryption recipient information to be partially decrypted.

5.1.2. Message: RecryptDataResponse

- Inherits: RecryptResponse

  JoseWebEncryption (Optional)

The partial decryption information to use to complete the decryption encrypted under the user's key.

  JoseWebEncryption (Optional)

The decryption key to use to complete the decryption encrypted under the user's key.

6. Acknowledgements

7. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982] [RFC6982] . The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982] [RFC6982] , "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. Reference Implementation

Organization: Comodo Group Inc.

Implementer: Phillip Hallam-Baker

Maturity: Experimental Prototype

This implementation was used to produce the reference section and all the examples in this document. Since the conversion of specification

to code is automatic, there is a high degree of assurance that the reference implementation is consistent with this document.

7.1.1. Coverage:

The draft-xx branch describes the code used to create version xx of this document.

The main current limitations are that the code only supports RSA key pairs and for ease of development the server does not persist keys across sessions. Nor does the implementation currently support the HTTP payload authentication and encryption layer or make use of TLS. These could be easily fixed.

The client and server are implemented as libraries that may be called from a multi-protocol server. A standalone server will be provided in a future release.

Only the JSON encoding is currently implemented. The JSON-B, JSON-C, ASN.1 and TLS Schema implementations are all supported by the code generation tool but not currently implemented as the build tool bindings for those encodings have not yet been finalized or documented.

The key restrictions for TLS key exchange have not yet been implemented.

The code has only been tested on Windows 10 but passed compatibility testing for both Mono and dotNetCore 10 run times which should in theory permit use on Linux and OSX platforms.

7.1.2. Licensing

The code is released under an MIT License

Source code is available from GitHub at https://github.com/hallambaker/Mathematical-Mesh

7.1.3. Implementation Experience

The implementation and specification documentation were developed in Visual Studio using the PHB Build Tools suite.

7.1.4. Contact Info

Contact Phillip Hallam-Baker phill@hallambaker.com

8. Security Considerations

[This is just a sketch for the present.]

9. IANA Considerations

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

10. References

10.1. Normative References

[Blaze98] "[Reference Not Found!]".

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh: Architecture", draft-hallambaker-mesh-architecture-03 (work in progress), May 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

10.2. Informative References

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013.

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com