Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 5, 2020

P. Hallam-Baker
November 2, 2019

# Mathematical Mesh 3.0 Part VIII: Cryptographic Algorithms
## draft-hallambaker-mesh-cryptography-04

Abstract

The Mathematical Mesh 'The Mesh' is an infrastructure that facilitates the exchange of configuration and credential data between multiple user devices and provides end-to-end security. This document describes the cryptographic algorithm suites used in the Mesh and the implementation of Multi-Party Encryption and Multi-Party Key Generation used in the Mesh.

[Note to Readers]

Discussion of this draft takes place on the MATHMESH mailing list (mathmesh@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=mathmesh.

This document is also available online at http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html [1] .

Status of This Memo

## Copyright Notice

## Table of Contents

## 1. Introduction

This document describes the cryptographic algorithm suites used in the Mesh and the implementation of Multi-Party Encryption and Multi-Party Key Generation used in the Mesh.

To allow use of Mesh capabilities on the least capable computing devices currently in use, separate schedules of recommended and required algorithms are specified for Standard Devices and Constrained Devices.

The Constrained device class may be considered to include most 8-bit CPUs equipped with sufficient memory to support the necessary operations. For example an Ardunino Mega 2560 which can perform ECDH key agreement and signature operations in times ranging from 3 to 8 seconds. While such a device is clearly not suited to perform such operations routinely, a one-time connection process that takes several minutes to complete need not be of major concern.

The Standard device class may be considered to include the vast majority of general purpose and personal computing devices manufactured since 2010. Even a Raspberry Pi Zero which currently retails at $5 is capable of performing the cryptographic functions required to implement the Mesh with negligible impact on the user.

## 2. Definitions

This section presents the related specifications and standard, the terms that are used as terms of art within the documents and the terms used as requirements language.

## 2.1. 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] .

## 2.2. Defined Terms

The terms of art used in this document are described in the Mesh Architecture Guide [draft-hallambaker-mesh-architecture] .

## 2.3. Related Specifications

The architecture of the Mathematical Mesh is described in the Mesh Architecture Guide [draft-hallambaker-mesh-architecture] . The Mesh documentation set and related specifications are described in this document.

## 2.4. Implementation Status

The implementation status of the reference code base is described in the companion document [draft-hallambaker-mesh-developer] .

## 3. Recommended and Required Algorithms

To allow implementation of Mesh capabilities on the widest possible range of devices, separate algorithm requirements and recommendations are specified for four classes of device:

Administration Device A general-purpose computing device that is used for Mesh administration functions.

Mesh Device A general-purpose computing device that is not used for Mesh administration functions with sufficient memory and processing power to perform public key cryptography operations without paying particular attention to the impact on performance.

Constrained Device An embedded computing device with limited memory and computing power that offers sufficient processing capabilities to perform occasional public key operations (e.g. during device initialization) but is not suited to repeated operations.

Bridge Device A trusted device that enables Mesh Devices to interoperate with Constrained devices.

Since Administration Devices and Mesh Devices MUST support communication with Mesh Devices and Constrained devices, they MUST meet all the REQUIRED algorithms for both types of device.

### 3.1. Mesh Device

Support for the following algorithms is REQUIRED:

- SHA-2-512 [SHA-2]
- HMAC-SHA-2-512 [RFC2104]
- HMAC-based Extract-and-Expand Key Derivation Function [RFC5869]
- AES-CBC-256 Encryption [FIPS197]
- Advanced Encryption Standard (AES) Key Wrap Algorithm [RFC3394]
- Montgomery Curve Diffie-Hellman Key Agreement X25519 and X448 [RFC7748]
- Edwards-Curve Digital Signature Algorithm Ed25519 and Ed448 [RFC8032]

Support for the following algorithms is RECOMMENDED:

- AES-GCM-256 Encryption [AES-GCM]
- SHA-3-512 [SHA-3]
- KMAC SHA-3-512 [SHA-3-Derived]

While the use of GCM is generally preferred over CBC mode in IETF security protocols, this mode is not currently supported by the reference implementation platform.

### 3.2. Constrained Device

Support for the following algorithms is REQUIRED:

- SHA-2-512 [SHA-2]
- HMAC-SHA-2-512 [RFC2104]
- HMAC-based Extract-and-Expand Key Derivation Function [RFC5869]
- Poly1035 Authenticated Encryption [RFC8439]
- ChaCha20 Encryption [RFC8439]
- Advanced Encryption Standard (AES) Key Wrap Algorithm [RFC3394]

- Edwards-Curve Digital Signature Algorithm Ed25519 [RFC8032]

- Edwards-Curve Diffie-Hellman Key Agreement Ed25519 [RFC8032]

Use of the Edwards Curves for Signature and Key Agreement allows both functions to be supported by a single library with no reduction in security.

## 4. Multi-Party Cryptography

The multi-party key generation and multi-party decryption mechanisms used in the Mesh protocols are made possible by the fact that Diffie Hellman key agreement and elliptic curve variants thereof support properties we call the Key Combination Law and the Result Combination Law.

Let {X, x}, {Y, y}, {E, e} be {public, private} key pairs.

The Key Combination law states that we can define an operator ? such that there is a keypair {Z, z} such that:

Z = X ? Y and z = (x + y) mod o (where o is the order of the group)

The Result Combination Law states that we can define an operator ? such that:

(x ? E) ? (y ? E) = (z ? E) = (e ? Z).

### 4.1. Application to Diffie Hellman (not normative)

For the Diffie Hellman system in a modular field p, o = p-1 and a ? b = a ? b = a.b mod p.

Proof:

By definition, X = e^x mod p, Y = e^y mod p, and Z = e^z mod p.

Therefore,

Z = e^z mod p = e^x+y mod p = (e^xe^y) mod p = e^x mod p.e^y mod p = X.Y

A similar proof may be constructed for the operator ?.

### 4.2. Multi-Party Key Generation

The Key Combination Law provides the basis for the Key Co-Generation technique used to ensure that the cryptographic keys used in devices connected to a Mesh profile are sufficiently random and have not been compromised by malware or other 'backdoor' compromise to the machine during or after manufacture.

For the Diffie Hellman system, the Key Combination law provides all the mechanism needed to implement a Key Co-Generation mechanism. If the Device key is {X, x}, the administration device can generate a Co-Generation Key Pair {Y, y} and generate a Device Connection Assertion for the final public key E calculated from knowledge of X and Y alone. Passing the value y to the device (using a secure channel) allows it to calculate the corresponding private key e required to make use of the Device Connection Assertion.

This approach ensures that a party with knowledge of either x or y but not both obtains no knowledge of e.

Section REF _Ref5309729 \w \h 5 describes the implementation of these schemes in the Mesh

### 4.3. Multi-Party Decryption

The Key Combination Law and Result Combination Law provide the basis for the Multi-Party Decryption technique used to support Mesh Encryption Groups.

Section REF _Ref5309538 \w \h 6 describes the application of this technique in the Mesh

### 4.4. Mutually Authenticated Key Exchange.

The Result Combination Law is used to provide a Key Exchange mechanism that provides mutual authentication of the parties while preserving forward secrecy.

### 4.5. Implementation

For elliptic curve cryptosystems, the operators ? and ? are point addition.

Implementing a robust Key Co-Generation for the Elliptic Curve Cryptography schemes described in [RFC7748] and [RFC8032] requires some additional considerations to be addressed.

- The secret scalar used in the EdDSA algorithm is calculated from the private key using a digest function. It is therefore necessary to specify the Key Co-Generation mechanism by reference to operations on the secret scalar values rather than operations on the private keys.

- The Montgomery Ladder traditionally used to perform X25519 and X448 point multiplication does not require implementation of a function to add two arbitrary points. While the steps required to create such a function are fully constrained by the specification, the means of satisfying the constraints is not.

### 4.5.1. Implementation for Ed25519 and Ed448

The data structures used to implement co-generation of public keys are defined in the main Mesh Reference Guide. This document describes only the additional implementation details.

Note that the 'private key' described in [RFC8032] is in fact a seed used to generate a 'secret scalar' value that is the value that has the function of being the private key in the ECDH algorithm.

To provision a new public key to a device, the provisioning device:

1. Obtains the device profile of the device(s) to be provisioned to determine the type of key to perform co-generation for. Let the device {public, private} key be {D, d}.

2. Generates a private key m with the specified number of bytes (32 or 57].

3. Calculates the corresponding public key M.

4. Calculates the Application public key A = D+M where + is point addition.

5. Constructs the application device entry containing the private key value m and encrypts under the device encryption key d.

On receipt, the device may at its option use its knowledge of the secret scalar corresponding to d and m to calculate the application secret scalar a or alternatively maintain the two secrets separately and make use of the result combination law to perform private key operations.

### 4.5.2. Implementation for X25519 and X448

While the point addition function can be defined for any elliptic curve system, it is not necessary to implement point addition to support ECDH key agreement.

In particular, point multiplication using the Montgomery ladder technique over Montgomery curves only operate on the x co-ordinate and only require point doubling operations.

For expediency, the current implementation of the Mesh reference code uses the Edwards curves for both signature and encryption pending announcement of platform support for both algorithms.

## 5. Multi-Party Key Generation

Multi-Party Key Generation is a capability that is used in the Mesh to enable provisioning of application specific private key pairs to connected devices without revealing any information concerning the application private key of the device.

For example, Alice provisions the confirmation service to her watch. The provisioning device could generate a signature key for the device and encrypt it under the encryption key of the device.  But this means that we cannot attribute signatures to the watch with absolute certainty as the provisioning device has had knowledge of the watch signature key.  Nor do we wish to use the device signature key for the confirmation service.

Multi-Party Key Generation allows an administration device to provision a connected device with an application specific private key that is specific to that application and no other such that the application can determine the public key of the device but has no knowledge of the private key.

Provisioning an application private key to a device requires the administration device to:

- Generate a new application public key for the device.

- Construct and publish whatever application specific credentials the device requires to use the application.

- Providing the information required to make use of the private key to the device.

Note that while the administration device needs to know the device application public key, it does not require knowledge of the device application private key.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html [2].]]

Two party key pair generation.

### 5.1. Example: Provisioning the Confirmation Service

For example, Alice provisions the confirmation service to her watch. The device profile of the watch specifies an Ed25519 signature key. Note that for production use, Ed448 is almost certainly prefered but Ed25519 has the advantage of more compact presentation.

```
Device Key

UDF Seed:       ZAAA-G4F5-G34W-HHLC-24AP-SZKI-VGO7-UHI
Private Key:
  B7 48 FF 8A  1B D2 04 BC  BA 53 70 80  91 66 08 8E
  26 15 B5 52  ED C9 C4 CE  80 D7 75 A6  BD A9 4A 1C
Secret Scalar:
56125651237600354228798699639619403779855821496596976661382393571903134631504
Public Key:
  2B B3 8F D1  93 16 E1 5E  24 44 BA 82  9F 4A D3 7D
  F6 7F B5 A0  92 65 A1 7B  F4 8C B6 51  6D F3 9F DC
Fingerprint:    MANZ-N7DM-BYZH-EQIO-E7YL-C6SX-GONP
```

The provisioning device could generate a signature key for the device and encrypt it under the encryption key of the device. But this means that we cannot attribute signatures to the watch with absolute certainty as the provisioning device has had knowledge of the watch signature key. Nor do we wish to use the device signature key for the confirmation service.

Instead, the provisioning device generates an overlay keypair:

```
Device Key

UDF Seed:       ZAAA-HIGJ-GGBY-NTBF-YS2T-ABNO-4WT2-SIY
Private Key:
  F9 44 69 38  A7 E2 52 A1  F6 1D 9D 9B  29 D1 5D 52
  77 23 26 0C  AF F3 14 A6  4E 95 CB 76  5B 24 B1 67
Secret Scalar:
49437431830830005043554286843388551057265319605872247754572631754571997461496
Public Key:
  C7 EF 8B 19  BE D9 E7 F7  45 17 73 8E  B3 92 6E 86
  C2 0F A4 06  84 58 BF 3B  FE EE 87 A5  B8 EB 62 0A
Fingerprint:    MBQO-UETO-EL5H-EQBI-PM4Y-QOE7-E6PH
```

The provisioning device can calculate the public key of the composite keypair by adding the public keys of the device profile and the companion public key:

```
   Composite public key = Device + Overlay:
     0A FD 34 5B  D7 49 47 BF  C3 CC 2E 4B  AB A6 BF FE
     E1 7A B1 3C  74 C6 49 EE  04 9C FF C0  4B D3 05 F6
   Fingerprint:    MBAK-3KK5-F3AN-E6AJ-EIIT-IIL6-7KTP
```

The provisioning device encrypts the private key of the comanion keypair (or the seed from which it was generated) under the encryption key of the device.

The provisioning device calculates the private key of the composite keypair by adding the two private key values modulo the order of the group and verifies that scalar multiplication of the base point returns the composite public key value.

```
Composite Secret Scalar = Device + Overlay:
3696218815216775247999600833593893946854897602495339232735523069524455498173
Fingerprint:    MBAK-3KK5-F3AN-E6AJ-EIIT-IIL6-7KTP
```

## 6.  Multi-Party Decryption

A key limitation of most deployed messaging systems 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://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html [3].]]

Traditional End-to-End Encryption is static.

The mathematical approach that makes key co-generation possible may be applied to support a public key encryption mode in which encryption is performed as usual but decryption requires the use of multiple keys. This approach is variously described in the literature as distributed key generation and proxy re-encryption [Blaze98] .

The approach specified in this document borrows aspects of both these techniques. This combined approach is called 'recryption'. Using recryption allows a sender to send a message to a group of users whose membership is not known to the sender at the time the message is sent and can change at any time.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html [4].]]

Recryption supports End-to-End Encryption in dynamic groups.

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:

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html [5].]]

Mesh/Recrypt Parties

The information stored at the recryption service is necessary but not sufficient to decrypt the message. Thus, no disclosure of the message plaintext occurs even in the event that an attacker gains full knowledge of all the information stored by the recryption service.

### 6.1. Mechanism

The mechanism used to support recryption is the same as the mechanism used to support key co-generation except that this time, instead of combining two keys to create one, the private component of a decryption key (i.e. the private key) is split into two parts, a recryption key and a decryption key.

Recall that the key combination law for Diffie Hellman crypto-systems states that we can add two private keys to get a third. It follows that we can split the private key portion of a keypair {G, g} into two parts by choosing a random number that is less than the order of the Diffie-Hellman group to be our first key x. Our second key is y = g - r mod o, where o is the order of the group.

Having generated x, y, we can use these to perform private key agreement operations on a public key E and then use the result combination law to obtain the same result that we would have obtained using g.

One means of applying this mechanism to recryption would be to generate a different random value x for each member of the group and store it at the recryption service and communicate the value y to the member via a secure channel. Applying this approach, we can clearly see that the recryption service gains no information about the value of the private key since the only information it holds is a random number which could have been generated without any knowledge of the group private key.

[RFC8032] requires that implementations derive the scalar secret by taking a cryptographic digest of the private key. This means that either the client or the service must use a non-compliant implementation. Given this choice, it is preferable to require that the non-standard implementation be required at the service rather than the client. This limits the scope of the non-conformant key derivation approach to the specialist recryption service and ensures that the client enforce the requirement to generate the private key component by means of a digest.

### 6.2. Implementation

Implementation of recryption in the Mesh has four parts:

- Creation and management of the recryption group.

- Provisioning of members to a recryption group.

- Message encryption.

- Message decryption.

These operations are all performed using the same catalog and messaging infrastructure provided by the Mesh for other purposes.

Each recryption group has its own independent Mesh account. This has many advantages:

- Administration of the recryption group may be spread across multiple Mesh users or transferred from one user to another without requiring specification of a separate management protocol to support these operations.

- The recryption account address can be used by Mesh applications such as group messaging, conferencing, etc. as a contact address.

- The contact request service can be used to notify members that they have been granted membership in the group.

#### 6.2.1. Group Creation

Creation of a Recryption group requires the steps of:

- Generating the recryption group key pair

- Creating the recryption group account

- Generating administrator record for each administrator.

- Publishing the administrator records to the recryption catalog.

Note that in principle, we could make use of the key combination law to enable separation of duties controls on administrators so that provisioning of members required multiple administrators to participate in the process. This is left to future versions.

### 6.2.2. Provisioning a Member

To provision a user as a member of the recryption group, the administrator requires their current recryption profile. The administrator MAY obtain this by means of a contact service request. As with any contact service request, this request is subject to access control and MAY require authorization by the intended user before the provisioning can proceed.

Having obtained the user's recryption profile, the administration tool generates a decryption private key for the user and encrypts it under the member's key to create the encrypted decryption key entry.

The administration tool then computes the secret scalar from the private key and uses this together with the secret scalar of the recryption group to compute the recryption key for the member. This value and the encrypted decryption key entry are combined to form the recryption group membership record which is published to the catalog.

### 6.2.3. Message Encryption and Decryption

Encryption of a messages makes use of DARE Message in exactly the same manner as any other encryption. The sole difference being that the recipient entry for the recryption operation MUST specify the recryption group address an not just the key fingerprint. This allows the recipient to determine which recryption service to contact to perform the recryption operation.

To decrypt a message, the recipient makes an authenticated recryption request to the specified recryption service specifying:

- The recipient entry to be used for decryption

- The fingerprint of the decryption key(s) the device would like to make use of.

- Whether or not the encrypted decryption key entry should be returned.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html [6].]]

Two key decryption.

The recryption service searches the catalog for the corresponding recryption group to find a matching entry. If found and if the recipient and proposed decryption key are dully authorized for the purpose, the service performs the key agreement operation using the recryption key specified in the entry and returns the result to the recipient.

The recipient then decrypts the recryption data entry using its device decryption key and uses the group decryption key to calculate the other half of the result. The two halves of the result are then added to obtain the key agreement value that is then used to decrypt the message.

### 6.3. Example: Messaging group

NB: The current code implements encryption in the Elliptic Curve Ed25519, not the Montgomery Curve X.25519 as it should. This will be lifted in the near future.

Alice creates an encryption keypair.

```
Group Key:
  51 C5 D1 84  5F C3 31 97  D3 80 2A 25  CE 7F 82 AF
  F0 30 5B 7D  52 D0 B6 CE  3C 1C 95 A0  33 CB 89 98
Value:
32231959352602868314725639964100760889451477817969178194652896778421852742616
```

To verify the proper function of the group, Alice creates a test message and encrypts it under the group key.

```
Message = This is a test as UTF8:
  54 68 69 73  20 69 73 20  61 20 74 65  73 74
```

```
[{
    "enc": "A256CBC",
    "Salt": "HmQcfDk3qgOudmzqLZdtEw",
    "recipients": [{
        "kid": "MCGR-HO5N-2FPS-VUFQ-DEKF-7TRA-JHM6",
        "epk": {
          "PublicKeyECDH": {
            "crv": "Ed25519",
            "Public": "T3NLoRxqQEyUPOPQJIp0aVDN4tH2f3BOVXqgCq5SyDs"}},
        "wmk": "jjSkEuWVIJZx8zLpedPMzJ9HyXzOcIqJh2i8MdRAHRnYa7E1oMKYyQ"}]},
  "Sjd7SsWo4o3OPfM-xBbnIQ"]
```

Alice decides to add Bob to the group.  She creates an encryption key for Bob: The decryption key is specified in the same way as any other Ed25519 private key using the hash of a private key seed value:

```
Bob's Member Key:
  C1 3D 09 F2  D0 46 2B ED  34 43 A2 06  EB C9 C5 6D
  68 92 08 3C  31 FC E7 35  EB 3D 65 82  82 66 92 1C
Value:
39765297387767177229798479396757873493329325191929553639423911645504436683984
```

The the recryption key is the group secret scalar minus (mod p) the secret scalar of Bob's private key:

```
Bob's Service Key:
   [Not specified as a digest input value]
Value:
6940673119500215512873533693428875877836385344799439767232887009488324560610
```

To decrypt:

```
   Member Agreement Value:
     1D 32 8A 93  79 65 29 58  89 0A D7 A4  2E 57 C6 F6
     95 B4 78 71  F2 21 EE 63  4F 1D C7 FB  A9 52 28 59

   Service Agreement Value:
     B8 C1 13 4C  1B 22 21 38  BD 2E 7E 96  39 2C 7B 21
     56 69 3B 58  6C FE 1E E9  5A 09 09 AD  30 39 09 DC

   Key Agreement IKM:
     E7 88 C7 64  12 02 0E E8  66 38 60 F8  C8 A8 DB 36
     FF DC E5 63  DF 64 61 D7  3E 41 E8 0E  8D FC 65 62
```

This value allows the test message to be decrypted.

## 7. Mutually Authenticated Key Agreement

Diffie Hellman key agreement using the authenticated public keys of the principals provides mutual authentication of those principals.

For example, if Alice's key pair is {a, A} and Bob's key pair is {b, B}, the Diffie Hellman key agreement value DH (a, B) = DH (b, A) can only be generated from the public information if a or b is known.

The chief disadvantage of this approach is that it only allows Alice and Bob to establish a single shared secret that will never vary and does not provide forward secrecy.  To avoid this, cryptographic protocols usually perform the key agreement against an ephemeral key and either accept that the client key is not authenticated or perform multiple key agreements and combine the results.

Using the Result Combination Law allows a key agreement mechanism to combine the benefits of mutual authentication with the use of ephemeral keys without the need for multiple private key operations or additional round trips.

In its simplest form, the key exchange has two parties which we refer to as the client and the server.  The client being the party that initiates the protocol exchange and the server being the party that responds.  Let the public key pair of the client be {a, A} and that of the server {b, B}.

Two versions of the key agreement mechanism are specified:

Client ephemeral  The client contributes an ephemeral key pair {n_A, N_A}. The effective public key of the client is A ? N_A.

The server uses its public key B.

The key agreement value is DH (a + n_A, B) = DH (b, A ? N_A)

Dual ephemeral  The client contributes an ephemeral key pair {n_A, N_A}. The effective public key of the client is A ? N_A.

The server contributes an ephemeral key pair {n_B, N_B}. The effective public key of the client is B ? N_B.

The key agreement value is DH (a + n_A, B ? N_B) = DH (b + n_B, A ? N_A)

The function of the ephemeral key is effectively that of a nonce but it is shared with the counter-party as a public key value.

The dual ephemeral approach has the advantage that it limits the scope for side channel attacks as both sides have contributed unknown information to the key agreement value. The disadvantage of this approach is that the key agreement value can only be calculated after the server has provided its ephemeral.

Implementations MAY take advantage of the result combination law to enable private key operations involving the authenticated key (or a contribution to it) to be performed in trustworthy hardware.

An advantage of this key exchange mechanism over the traditional TLS key exchange approach is that no signature operation is involved, thus ensuring that either party can repudiate the exchange and thus the claim that they were in communication.

The master secret is calculated from the key agreement value in the usual fashion. For ECDH algorithms, this comprises the steps of converting the key agreement value to an octet string which forms the input to a Key Derivation Function.

## 8. Security Considerations

The security considerations for use and implementation of Mesh services and applications are described in the Mesh Security Considerations guide [draft-hallambaker-mesh-security] .

## 9. IANA Considerations

This document requires no IANA actions.

## 10. Acknowledgements

A list of people who have contributed to the design of the Mesh is presented in [draft-hallambaker-mesh-architecture] .

## 11. Examples

### 11.1. Key Combination

#### 11.1.1. Ed25519

#### 11.1.2. Ed448

#### 11.1.3. X25519

#### 11.1.4. X448

### 11.2. Group Encryption

#### 11.2.1. X25519

#### 11.2.2. X448

## 12. References

### 12.1. Normative References

[AES-GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", November 2007.

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part I: Architecture Guide", draft-hallambaker-mesh-architecture-11 (work in progress), October 2019.

[draft-hallambaker-mesh-security]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VII: Security Considerations", draft-hallambaker-mesh-security-02 (work in progress), October 2019.

[FIPS197] NIST, "Advanced Encryption Standard (AES)", November 2001.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997.

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

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018.

[SHA-2] NIST, "Secure Hash Standard", August 2015.

[SHA-3] Dworkin, M., "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", August 2015.

[SHA-3-Derived] Kelsey, J., Chang, S., and R. Perlner, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash SHARE", December 2016.

## 12.2. Informative References

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

[draft-hallambaker-mesh-developer] Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", draft-hallambaker-mesh-developer-09 (work in progress), October 2019.

## 12.3. URIs

[1] http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html

[2] http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html

[3] http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html

[4] http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html

[5] http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html

[6] http://mathmesh.com/Documents/draft-hallambaker-mesh-cryptography.html

## Author's Address

Phillip Hallam-Baker

Email: phill@hallambaker.com