Abstract

The Mathematical Mesh 'The Mesh' is an end-to-end secure infrastructure that makes computers easier to use by making them more secure. The Mesh provides a set of protocol and cryptographic building blocks that enable encrypted data stored in the cloud to be accessed, managed and exchanged between users with the same or better ease of use than traditional approaches which leave the data vulnerable to attack.

This document provides an overview of the Mesh data structures, protocols and examples of its use.

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

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 https://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 October 6, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
# Table of Contents

1. Introduction ............................................. 3
2. Definitions ............................................. 4
   2.1. Related Specifications .......................... 5
   2.2. Defined Terms .................................. 5
   2.3. Requirements Language .......................... 5
   2.4. Implementation Status .......................... 5
3. Architecture ........................................... 6
   3.1. A Personal PKI .................................... 6
   3.2. Support for Legacy and Future Applications ...... 8
      3.2.1. Confirmation Protocol .................... 8
      3.2.2. Encrypted Authenticated Resource Locators ... 8
      3.2.3. Friendly Names ............................. 9
      3.2.4. Uniform Data Fingerprints .................. 10
      3.2.5. Secure Internet Names ....................... 10
   3.3. Cryptography ...................................... 10
      3.3.1. Best Practice by Default ................ 11
      3.3.2. Multi-Level Security ...................... 12
      3.3.3. Multi-Party Cryptography ................. 12
      3.3.4. Data At Rest Encryption .................... 13
      3.3.5. Personal Key Escrow ........................ 14
   3.4. Mesh Services ..................................... 14
      3.4.1. Catalogs .................................... 15
      3.4.2. Spools ..................................... 16
      3.4.3. Future Applications ........................ 16
4. User Experience ......................................... 17
   4.1. Creating and Registering a Mesh Profile ........ 18
   4.2. Mesh Service ...................................... 18
   4.3. Mesh Catalogs ..................................... 19
   4.4. Connecting and Authorizing Additional Devices ... 19
      4.4.1. Direct Connection .......................... 20
      4.4.2. Pin Connection ............................. 20
      4.4.3. EARL/QR Code Connection .................... 20
   4.5. Contact Requests .................................. 21
      4.5.1. Remote ..................................... 21
      4.5.2. Static QR Code ............................. 21
      4.5.3. Dynamic QR Code ............................ 22
4.6. Storing and Sharing Data in the Cloud

4.6.1. EARL Exchange

4.6.2. Encryption Groups

4.7. Escrow and Recovery of Keys

5. Security Considerations

6. IANA Considerations

7. Acknowledgements

8. References

8.1. Normative References

8.2. URIs

Author’s Address

1. Introduction

The Mathematical Mesh (Mesh) is a user centered Public Key Infrastructure that uses cryptography to make computers easier to use.

This document is not normative. It provides an overview of the Mesh comprising a description of the architecture, and a discussion of typical use cases and requirements. The remainder of the document series provides a summary of the principal components of the Mesh architecture and their relationship to each other.

Normative descriptions of the individual Mesh encodings, data structures and protocols are provided in separate documents addressing each component in turn.

The currently available Mesh document series comprises:

I. Architecture (This document.) Provides an overview of the Mesh as a system and the relationship between its constituent parts.

II. Uniform Data Fingerprint. Describes the UDF format used to represent cryptographic nonces, keys and content digests in the Mesh and the use of Encrypted Authenticated Resource Locators (EARLs) and Strong Internet Names (SINs) that build on the UDF platform.

III. Data at Rest Encryption. Describes the cryptographic message and append-only sequence formats used in Mesh applications and the Mesh Service protocol.

IV. Schema Reference. Describes the syntax and semantics of Mesh Profiles, Container Entries and Mesh Messages and their use in Mesh Applications.

VI. The Trust Mesh. Describes the social work factor metric used to evaluate the effectiveness of different approaches to exchange of credentials between users and organizations in various contexts and argues for a hybrid approach taking advantage of direct trust, Web of Trust and Trusted Third Party models to provide introductions.

VII. Security Considerations. Describes the security considerations for the Mesh protocol suite.

VIII. Cryptographic Algorithms. Describes the recommended and required algorithm suites for Mesh applications and the implementation of the multi-party cryptography techniques used in the Mesh.

The following documents describe technologies that are used in the Mesh but do not form part of the Mesh standards suite:

JSON-BCD Encoding. Describes extensions to the JSON serialization format to allow direct encoding of binary data (JSON-B), compressed encoding (JSON-C) and extended binary data encoding (JSON-D). Each of these encodings is a superset of the previous one so that JSON-B is a superset of JSON, JSON-C is a superset of JSON-B and JSON-D is a superset of JSON-C.

DNS Web Service Discovery. Describes the means by which prefixed DNS SRV and TXT records are used to perform discovery of Web Services.

The following documents describe aspects of the Mesh Reference implementation:

Mesh Developer. Describes the reference code distribution license terms, implementation status and currently supported functions.

Mesh Platform. Describes how platform specific functionality such as secure key storage and trustworthy computing features are employed in the Mesh.

2. Definitions

This section presents the related specifications and standards on which the Mesh is built, the terms that are used as terms of art within the Mesh protocols and applications and the terms used as requirements language.
2.1. Related Specifications

Besides the documents that form the Mesh core, the Mesh makes use of many existing Internet standards, including:

Cryptographic Algorithms  Mesh applications use the cryptographic algorithm suites specified by the application. The cryptographic algorithms used in the Mesh itself are limited to SHA-2 [SHA-2] and SHA-3 [SHA-3] digest functions and AES Encryption [FIPS197] and RSA Signature.

The Ed25519 and Ed448 algorithms are currently used for both signature [RFC8032] and encryption. The Edwards Curve being preferred over the Montgomery for Encryption as it afforded a more straightforward implementation of techniques such as co-generation of public key pairs and proxy re-encryption.

Transport  All Mesh Services make use of multiple layers of security. Protection against traffic analysis and metadata attacks are provided by use of Transport Layer Security [RFC5246]. At present, the HTTP/1.1 [RFC7231] protocol is used to provide framing of transaction messages.

Encoding  All Mesh protocols and data structures are expressed in the JSON data model and all Mesh applications accept data in standard JSON encoding [RFC7159]. The JOSE Signature [RFC7515] and Encryption [RFC7516] standards are used as the basis for object signing and encryption.

2.2. Defined Terms

TBS

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 RFC 2119 [RFC2119].

2.4. Implementation Status

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

The Mathematical Mesh (Mesh) is a user centered Public Key Infrastructure that uses cryptography to make computers easier to use.

For several decades, it has been widely observed that most users are either unwilling or unable to make even the slightest efforts to protect their security, still less those of other parties. Yet despite this observation being widespread, the efforts of the IT security community have largely focused on changing this user behavior rather than designing applications that respect it.

The Mesh is based on the principle that if the Internet is to be secure, it must become effortless to use applications securely. Rather than beginning the design process by imagining all the possible modes of attack and working out how to address these with the least possible inconvenience, we must reverse the question and ask how much security can be provided without requiring any effort on the user’s part to address it.

3.1. A Personal PKI

The Mesh provides a personal PKI to which a user can connect all their devices so they work seamlessly together. Each connected device has a unique set of keys used for encryption, signature and authentication that are bound to that device.

Processing of Mesh assertions is similar to processing of PKIX certificates and SAML assertions but with important semantic differences. Mesh assertions do not have predetermined expiry time and the processing model is based on keys rather than identity.

As in SAML, the basic unit in the Mesh PKI is a signed assertion. A Mesh Profile is a self-signed Mesh Assertion. Currently, three types of Mesh Profile are defined:

Device Profile  Specifies the public keys to be used by a device for Encryption, Signature and Authentication and additional device description information.

Master Profile  Specifies a master signature key that is used as the axiom of trust for a ‘life-long’ cryptographic identity and specifies a set of administration keys authorized to sign Mesh assertions on behalf of that identity.

Service Profile  Binds a Master profile to a Mesh Service account that acts as a co-ordination service for all the devices connected
to that profile and a point of presence at which Mesh Messages may be exchanged with other Mesh Services.

Devices are connected to the user’s personal profile by means of a connection entry consisting of one or more connection assertions and one or more device private data entries describing device specific application configuration data.

Connection Assertion  A Mesh Assertion signed by an authorized administration key binding a device to a Mesh profile by means of the UDF Content Digest of its Device profile. A device may have multiple connection assertions for different purposes which may include authorizations for specific actions (e.g. permission to read or write specific types of data in a Mesh Service Account).

Device Private Data  A DARE Message encrypted under the device encryption key containing device specific configuration information allowing it to connect to one of more applications.

For example, Alice has a laptop, a desktop and a mobile connected to her personal profile each of which is authorized to control the garage door by means of an authenticated interaction with the door control computer:

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

Alice’s Personal PKI

If its purpose requires, a connection assertion MAY be public. This allows a device to authenticate to Web Services and authenticate messages using its embedded public key credentials by presenting a suitable connection assertion.

For example, Alice submits an update to a source code repository. The device keys are used to authenticate access to the service and then sign them. The source code repository only has knowledge of Alice’s Master Profile fingerprint but can verify the access request and code signature by validating these keys under the device profile and the connection assertion provided.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [3].]]
3.2. Support for Legacy and Future Applications

The Mesh provides an infrastructure for supporting existing Internet security applications and a set security features that may be used to build new ones.

For example, Alice uses the Mesh to provision and maintain the keys she uses for OpenPGP, S/MIME, SSH and IPSEC. She also uses the credential catalog for end-to-end secure management of the usernames and passwords for her Web browsing and other purposes:

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

The Mesh design is highly modular allowing components that were originally designed to support a specific requirement within the Mesh to be applied generally.

3.2.1. Confirmation Protocol

The basic device connection protocol requires the ability for one device to send a connection request to the Mesh service hosting the user’s profile. To accept the device connection, the user connects to the service using an administration device, reviews the pending requests and creates the necessary device connection assertion if it is accepted.

The confirmation protocol generalizes this communication pattern allowing any authorized party to post a short accept/reject question to the user who may (or may not) return a signed response. This feature can be used as improvement on traditional second factor authentication providing resistance to man-in-the-middle attacks and providing a permanent non-repudiable indication of the user’s specific intent.

3.2.2. Encrypted Authenticated Resource Locators

Various schemes have been used to employ QR Codes as a means of device and/or user authentication. In many of these schemes a QR code contains a challenge nonce that is used to authenticate the connection request.

The Mesh supports a QR code connection mode employing the Encrypted Authenticated Resource Locator (EARL) format. An EARL is a type of UDF URI that contains a domain name and a master key:
The EARL may be expressed as a QR code:

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

QR Code representation of the EARL

An EARL is resolved by presenting the content digest fingerprint of the encryption key to a Web service hosted at the specified domain. The service returns a DARE Message whose payload is encrypted and authenticated under the specified master key. Since the content is stored on the service under the fingerprint of the key and not the key itself, the service cannot decrypt the plaintext. Only a party that has access to the QR code can decrypt the message.

3.2.3. Friendly Names

Internet addressing schemes are designed to provide a globally unique (or at minimum unambiguous) name for a host, service or account. In the early days of the Internet, this resulted in addresses such as 10.2.3.4 and alice@example.com which from a usability point of view might be considered serviceable if not ideal. Today the Internet is a global infrastructure servicing billions of users and tens of billions of devices and accounts are more likely to be alice.lastname.1934@example.com than something memorable.

Friendly names provide a user or community specific means of identifying resources that may take advantage of geographic location or other cues to resolve possible ambiguity. If Alice says to her voice activated device "close the garage door" it is implicit that it is her garage door that she wishes to close. And should Alice be fortunate enough to own two houses with a garage, it is implicit that it is the garage door of the house she is presently using that she wishes to close.

The Mesh Contacts Catalog provides a directory mapping friendly names to devices that is available to all Alice’s connected devices so that she may give and instruction to any of her devices using the same friendly name and expect consistent results.
3.2.4. Uniform Data Fingerprints.

The Uniform Data Fingerprint (UDF) format provides a compact means of presenting cryptographic nonces, keys and digest values using Base32 encoding that resists semantic substitution attacks. UDF provides a convenient format for data entry. Since the encoding used is case-insensitive, UDFs may if necessary be read out over a voice link without excessive inconvenience.

The following are examples of UDF values:

```
```

UDF content digests are used to support a direct trust model similar to that of OpenPGP. Every Mesh Profile is authenticated by the UDF fingerprint of its signature key. Mesh Friendly Names and UDF Fingerprints thus serve analogous functions to DNS names and IP Addresses. Like DNS names, Friendly Names provide the basis for application-layer interactions while the UDF Fingerprints are used as to provide the foundation for security.

3.2.5. Secure Internet Names

Secure Internet Names bind an Internet address such as a URL or an email address to a Security Policy by means of a UDF content digest of a document describing the security policy. This binding enables an Internet client to ensure that the security policy is applied when connecting to the address. For example, ensuring that an email sent to an address must be end-to-end encrypted under a particular public key or that access to a Web Service requires a particular set of security enhancements.

```
```

3.3. Cryptography

All the cryptographic algorithms used in the Mesh are either industry standards or present a work factor that is provably equivalent to an industry standard approach.

Existing Internet security protocols are based on approaches developed in the 1990s when performance tradeoffs were a prime consideration in the design of cryptographic protocols. Security was focused on the transport layer as it provided the best security possible given the available resources.
With rare exceptions, most computing devices manufactured in the past ten years offer either considerably more computing power than was typical of 1990s era Internet connected machines or considerably less. The Mesh architecture is designed to provide security infrastructure both classes of machine but with the important constraint that the less capable ‘constrained’ devices are considered to be ‘network capable’ rather than ‘Internet capable’ and that the majority of Mesh related processing will be offloaded to another device.

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

Constrained Devices connected through a Mesh Hub.

3.3.1. Best Practice by Default

Except where external applications demand otherwise, the Mesh requires that the following ‘best practices’ be followed:

Industry Standard Algorithms  All cryptographic protocols make use of the most recently adopted industry standard algorithms.

Strongest Work Factor  Only the strongest modes of each cipher algorithm are used. All symmetric encryption is performed with 256-bit session keys and all digest algorithms are used in 512-bit output length mode.

Key Hygiene  Separate public key pairs are used for all cryptographic functions: Encryption, Signature and Authentication. This enables separate control regimes for the separate functions and partitioning of cryptographic functions within the application itself.

Bound Device Keys  Each device has a separate set of Encryption, Signature and Authentication key pairs. These MAY be bound to the device to which they are assigned using hardware or other techniques to prevent or discourage export.

No Optional Extras  Traditional approaches to security have treated many functions as being ‘advanced’ and thus suited for use by only the most sophisticated users. The Mesh rejects this approach noting that all users operate in precisely the same environment facing precisely the same threats.
3.3.2. Multi-Level Security

All Mesh protocol transactions are protected at the Transport, Message and Data level. This provides security in depth that cannot be achieved by applying security at the separate levels independently. Data level encryption provides end-to-end confidentiality and non-repudiation, Message level authentication provides the basis for access control and Transport level encryption provides a degree of protection against traffic analysis.

3.3.3. Multi-Party Cryptography

Existing Internet security protocols only make use of cryptographic operations that involve two parties, one of which has knowledge of the private key and the other which knows only the public key. The Mesh makes use of multi-party cryptographic techniques in which the public operation is performed in the usual manner but the private key operations are performed by multiple parties.

For example, Alice uses secure Mesh messaging on her mobile device. Since the device might be lost, she does not want to store her Mesh messaging decryption device on the device itself. Nor does she want to store that key in the cloud. Instead, the key is split into two parts, one of which is (securely) provisioned to the device and the other to a cloud service.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [7].]]

Split Key Decryption.

This configuration allows the device to be used to decrypt messages if and only if the cloud service allows. If the device is lost, the cloud service is notified and all subsequent decryption requests are refused. The cloud service cannot decrypt messages under any circumstances, a fact which can be demonstrated by using a random number generated without access to the decryption key as the private key of the service and only using the decryption key to generate the corresponding device decryption key.

Multi-party techniques are also used in key generation. This allows an administration application to ensure that devices use key pairs that are sufficiently random without having access to the private key information itself.
Key Pair Co-Generation.

3.3.4. Data At Rest Encryption

The Data At Rest Encryption (DARE) format is used for all confidentiality and integrity enhancements. The DARE format is based on the JOSE Signature and Encryption formats and the use of an extended version of the JSON encoding allowing direct encoding of binary objects.

3.3.4.1. DARE Message

The DARE Message format offers similar capabilities to existing formats such as OpenPGP and CMS without the need for onerous encoding schemes. DARE Assertions are presented as DARE Messages.

A feature of the DARE Message format not supported in existing schemes is the ability to encrypt and authenticate sets of data attributes separately from the payload. This allows features such as the ability to encrypt a subject line or content type for a message separately from the payload.

3.3.4.2. Dare Container

A DARE Container is an append-only sequence of Entries. A key feature of the DARE Container format is that entries MAY be encrypted and/or authenticated incrementally. Individual entries MAY be extracted from a DARE Container to create a stand-alone DARE Message.

Containers may be authenticated by means of a Merkle tree of digest values on the individual frames. This allows similar demonstrations of integrity to those afforded by Blockchain to be provided but with much greater efficiency.

Unlike traditional encryption formats which require a new public key exchange for each encrypted payload, the DARE Container format allows multiple entries to be encrypted under a single key exchange operation. This is particularly useful in applications such as encrypting server transaction logs. The server need only perform a single key exchange operation is performed each time it starts to establish a master key. The master key is then used to create fresh symmetric keying material for each entry in the log using a unique nonce per entry. Integrity is provided by a Merkle tree calculated...
over the sequence of log entries. The tree apex is signed at regular intervals to provide non-repudiation.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [9].]]

**DARE Container containing a transaction log.**

### 3.3.5. Personal Key Escrow

One of the core objectives of the Mesh is to make data level encryption ubiquitous. While data level encryption provides robust protection of data confidentiality, loss of the ability to decrypt means data loss.

For many Internet users, data availability is a considerably greater concern than confidentiality. Ten years later, there is no way to replace pictures of the children at five years old. Recognizing the need to guarantee data recovery, the Mesh provides a robust personal key escrow and recovery mechanism. Lawful access is not supported as a requirement.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [10].]]

**Key escrow and recovery**

Besides supporting key recovery in the case of loss, the Mesh protocols potentially support key recovery in the case of the key holder’s death. The chief difficulty faced in implementing such a scheme being developing an acceptable user interface which allows the user to specify which of their data should survive them and which should not. As the apothegm goes: Mallet wants his beneficiaries to know where he buried Aunt Agatha’s jewels but not where he buried Aunt Agatha.

### 3.4. Mesh Services

Mesh Services provide an ‘always available’ network presence for one or more Mesh profiles. This allows a device that is connected to a Mesh profile but not always available on the network to receive Mesh messages from other Mesh users and to share information with other devices connected to the same profile.
The Mesh currently supports eight separate application profiles through operations on two basic data collection types: Catalogs and Spools. Both of which are implemented as DARE containers. This allows for a remarkably compact application protocol since the only operation needed to perform the functionality of all the Mesh applications is the ability to synchronize a DARE container from a master copy held on the service to full or partial copies on the connected devices.

While such economy of mechanism supporting a wide variety of functions is of course desirable in its own right, it is in part a consequence of the fact that a host supporting a true end-to-end service with data level encryption cannot be expected to support a rich set of operations on data it is unable to read.

3.4.1. Catalogs

Mesh Catalogs are a DARE Container whose entries represent a set of objects with no inherent ordering. Examples of Mesh catalogs include:

- Devices The devices connected to the corresponding Mesh profile.
- Contacts Logical and physical contact information for people and organizations.
- Bookmarks Web bookmarks and citations.
- Credentials Username and password information for network resources.
- Calendar Appointments and tasks.
- Network Network access configuration information allowing access to WiFi networks and VPNs.
- Application Configuration information for applications including mail (SMTP, IMAP, OpenPGP, S/MIME, etc) and SSH.

The first two of these catalogs have special functions within the Mesh. The Devices Catalog tracks the devices connected to a Mesh Profile and the Contacts Catalog is used to support the use of Friendly Names in place of Internet addresses and to perform access control on inbound messages.
3.4.2. Spools

Mesh Spools are DARE Container whose entries comprise a sequence of Mesh Messages.

The inbound spool contains Messages received from other parties and the outbound spool contains Messages queued to be sent to other Mesh users.

The Mesh Messaging model enforces a four-corner approach in which all messages exchanged between devices pass through an inbound and outbound Mesh Service. This approach permits ingress and egress controls to be enforced on the messages and that every message is properly recorded in the appropriate spools.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [11].]]

For efficiency and to limit the scope for abuse, all inbound Mesh Messages are subject to access control and limited in size to 64KB or less. This limit has proved adequate to support transfer of complex control messages and short content messages. Transfer of data objects of arbitrary size may be achieved by sending a control message containing a URI for the main content which may then be fetched using a protocol such as HTTP.

This approach makes transfers of very large data sets (i.e. multiple Terabytes) practical as the ‘push’ phase of the protocol is limited to the transfer of the initial control message. The bulk transfer is implemented as a ‘pull’ protocol allowing support for features such as continuous integrity checking and resumption of an interrupted transfer.

3.4.3. Future Applications

Since a wide range of network applications may be reduced to synchronization of sets and lists, the basic primitives of Catalogs and Spools may be applied to apply end-to-end security to an even wider variety of applications.

For example, a Spool may be used to maintain a mailing list, track comments on a Web forum or record annotations on a document. Encrypting the container entries under a multi-party encryption group allows such communications to be shared with a group of users while maintaining full end-to-end security and without requiring every
party writing to the spool to know the public encryption key of every recipient.

Another interesting possibility is the use of DARE Containers as a file archive mechanism. A single signature on the final Merkle Tree digest value would be sufficient to authenticate every file in the archive. Updates to the archive might be performed using the same container synchronization primitives provided by a Mesh Service. This approach could afford a robust, secure and efficient mechanism for software distribution and update.

4. User Experience

This section describes the Mesh in use. These use cases described here are re-visited in the companion Mesh Schema Reference [draft-hallambaker-mesh-schema] and Mesh Protocol Reference [draft-hallambaker-mesh-protocol] with additional examples and details.

For clarity and for compactness, these use cases are illustrated using the command line tool meshman.

The original design brief for the Mesh was to make it easier to use the Internet securely. Over time, it was realized that users are almost never prepared to sacrifice usability or convenience for security. It is therefore insufficient to minimize the cost of security, if secure applications are to be used securely they must be at least as easy to use as those they replace. If security features are to be used, they must not require the user to make any additional effort whatsoever.

The key to meeting this constraint is that any set of instructions that can be written down to be followed by a user can be turned into code and executed by machine. Provided that the necessary authentication, integrity and confidentiality controls are provided. Thus, the Mesh is not just a cryptographic infrastructure that makes use of computer systems more secure, it is a usability infrastructure that makes computers easier to use by providing security.

The user experience is thus at the heart of the design of the Mesh and a description of the Mesh Architecture properly begins with consideration of the view of the system that matters most: that of the user.

The principle security protocols in use today were designed at a time when most Internet users made use of either a single machine or one of a number of shared machines connected to a shared file store. The problem of transferring cryptographic keys and configuration data
between machines was rarely considered and when it was considered was usually implemented badly. Today the typical user owns or makes use of multiple devices they recognize as a computer (laptop, tablet) and an even greater number of devices that they do not recognize as computers but are (almost any device with a display).

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [12]].]

4.1. Creating and Registering a Mesh Profile

The first step in using the Mesh is to create a personal profile. From the user’s point of view a profile is a collection of all the configuration data for all the Mesh enabled devices and services that they interact with.

```` Example ArchitectureCreate ````

Note that the user does not specify the cryptographic algorithms to use. Choice of cryptographic algorithm is primarily the concern of the protocol designer, not the user. The only circumstance in which users would normally be involved in algorithm selection is when there is a transition in progress from one algorithm suite to another.

4.2. Mesh Service

A Mesh Service provides an ‘always available’ point of presence that is used to exchange data between devices connected to the connected profile and send and receive Mesh Messages to and from other Mesh users.

To use a Mesh Service, a user creates a Mesh Service account. This is analogous to an SMTP email service but with the important distinction that the protocol is designed to allow users to change their Mesh Service provider at any time they choose with minimal impact.

The account is created by sending an account registration request to the chosen Mesh Service. If accepted, the Mesh Service creates a new account and creates containers to hold the associated catalogs and spools:

```` Example ArchitectureRegister ````

As with any other Internet service provision, Mesh Service providers may impose constraints on the use of their service such as the amount
of data they send, store and receive and charge a fee for their service.

4.3. Mesh Catalogs

A Mesh Catalog contains a set of entries, each of which has a unique object identifier. Catalog entries may be added, updated or deleted. By default, all catalog entries are encrypted. Applying the Default Deny principle, in normal circumstances, the Mesh Service is not capable of decrypting any catalog excepting the Contacts catalog which is used as a source of authorization data in the Access Control applied to inbound messaging requests.

For example, the entries in the credentials catalog specify username and password credentials used to access Internet services. Adding credentials to her catalog allows Alice to write scripts that access password protected resources without including the passwords in the scripts themselves:

```` Example ArchitectureCredential ````

4.4. Connecting and Authorizing Additional Devices

Having established a Mesh profile, a user may connect any number of devices to it. Connecting a device to a Mesh profile allows it to share data with, control and be controlled by other devices connected to the profile.

Although any type of network capable device may be connected to a Mesh profile, some devices are better suited for use with certain applications than others. Connecting an oven to a Mesh profile could allow it to be controlled through entries to the user’s recipe and calendar catalogs and alert the user when the meal is ready but attempting to use it to read emails or manage Mesh profiles.

Three connection mechanisms are currently specified, each of which provides strong mutual authentication: Direct, PIN and QR.

Since approval of a connection request requires the creation of a signed Connection Assertion, requests must be approved by a device that has access to an administration key authorized by the user’s Master Profile. Such devices are referred to as Administration devices. Administration devices must have data entry (e.g. keyboard) and output (e.g. display) affordances to support any of the currently defined connection mechanisms. The QR code connection mechanism also requires a suitable camera.
It will be noted that the process of connecting a device that contains a preconfigured set of device keys might in principle expose the user to the risk that the manufacturer has retained knowledge of these keys and that this might be used to create a 'backdoor'.

This risk is controlled using the key co-generation technique described earlier. The original device profile is combined with a device profile provided by the user to create a composite device profile. This ensures that every device uses a unique profile even if they are initialized from a shared firmware image containing a fixed set of device key pairs.

4.4.1. Direct Connection

The direct connection mechanism requires that both the administration device and the device originating the connection request have data entry and output affordances and that it is possible for the user to compare the authentication codes presented by the two devices to check that they are identical.

```` Example ArchitectureConnectDirect ````

4.4.2. PIN Connection

The PIN Connection mechanism is similar to the Direct connection mechanism except that the process is initiated on an administration device by requesting assignment of a new authentication PIN. The PIN is then input to the connecting device to authenticate the request.

```` Example ArchitectureConnectPIN ````

If the Device Profile fingerprint is known at the time the PIN is generated, this can be bound to the PIN authorization assertion to permit connection of a specific device.

4.4.3. EARL/QR Code Connection

The EARL/QR code connection mechanisms are used to connect a constrained device to a Mesh profile by means of an Encrypted Authenticated Resource Locator, typically presented as a QR code on the device itself or its packaging.

[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-architecture.html [13].]
Since the meshman tool does not support QR input, it is decoded using a separate tool to recover the UDF EARL which is presented as a command line parameter:

```` Example ArchitectureConnectQR ````

4.5. Contact Requests

As previously stated, every inbound Mesh message is subject to access control. The user’s contact catalog is used as part of the access control authentication and authorization mechanism.

By default, the only form of inbound message that is accepted without authorization in the contact catalog is a contact request. Though for certain Mesh users (e.g. politicians, celebrities) even contact requests might require some form of prior approval (e.g. endorsement by a mutual friend).

A Mesh Contact Assertion may be limited to stating the user’s profile fingerprint and Mesh Service Account(s). For most purposes however, it is more convenient to present a Contact Assertion that contains at least as much information as is typically provided on a business or calling card:

```` Example ArchitectureContactDefinition ````

User’s may create multiple Contact Assertions for use in different circumstances. A user might not want to give their home address to a business contact or their business address to a personal friend.

4.5.1. Remote

In the most general case, the participants are remote from each other and one user must make a contact request of the other:

```` Example ArchitectureContactRequest ````

4.5.2. Static QR Code

A DARE contact entry may be exchanged by means of an EARL UDF. This is typically presented by means of a QR code which may be created using the meshman tool and a QR code generator:

```` Example ArchitectureContactQR ````

The resulting QR code may be printed on a business card, laser engraved on a luggage tag, etc.
QR Code representation of the EARL

To accept the contact request, the recipient merely scans the code with a Mesh capable QR code reader. They are asked if they wish to accept the contact request and what privileges they wish to authorize for the new contact.

4.5.3. Dynamic QR Code

If it is possible for the device to generate a new QR code for the contact request, it is possible to support bi-directional exchange of credentials with strong mutual authentication.

For example, Alice selects the contact credential she wishes to pass to Bob on her mobile device which presents an EARL as a QR code. Bob scans the QR code with his mobile device which retrieves Alice’s credential and asks if Bob wishes to accept it and if he wishes to share his credential with Alice. If Bob agrees, his device makes a Remote Contact request authenticated under a key passed to his device with Alice’s Contact Assertion.

The Dynamic QR Code protocol may be applied to support exchange of credentials between larger groups. Enrolling the contact assertions collected in such circumstances in a notarized append only log (e.g. a DARE Container) provides a powerful basis for building a Web of Trust that is equivalent to but considerably more convenient than participation in PGP Key Signing parties.

4.6. Storing and Sharing Data in the Cloud

The DARE Message format supports the use of two novel data encryption capabilities that are particularly suited to ‘cloud computing’ environments in which the owner of a data collection outsources the task of managing it.

4.6.1. EARL Exchange

An EARL is a form of URI that specifies a means of locating and decrypting a DARE Message stored on a Web Service.
Preparing data for delivery using an EARL is a task more typically performed by a document management system than an individual user.

The meshman encryption command can be used to encrypt a file as a DARE Message with the appropriate file name and return the corresponding master key required to create the corresponding EARL:

```` Example ArchitectureEARL ````

4.6.2. Encryption Groups

As previously discussed, the Mesh makes use of multi-party encryption techniques to mitigate the risk of a device compromise leading to disclosure of confidential data. The Mesh also allows these techniques to be applied to provide Confidential Document Control.

A Mesh Encryption Group is a special type of Mesh Service Account that is controlled by one of more group administrators. The Encryption Group Key is a normal ECDH public key used in the normal manner. The decryption key is held by the group administrator. To add a user to the group, the administrator splits the group private key into two parts, a service key and a user key. These parts are encrypted under the public encryption keys of their assigned parties. The encrypted key parts form a decryption entry for the user is added to the Members Catalog of the Encryption Group at the Mesh Service.

```` Example ArchitectureRecrypt ````

When a user needs to decrypt a document encrypted under the group key, they make a request to the Mesh Service which checks to see that they are authorized to read that particular document, have not exceeded their decryption quota, etc. If the request is approved, the service returns the partial decryption result obtained from the service’s key part together with the encrypted user key part. To complete the decryption process, the user decrypts their key part and uses it to create a second partial decryption result which is combined with the first to obtain the key agreement value needed to complete the decryption process.

```` Example ArchitectureRecrypt ```
Should requirements demand, the same principle may be applied to achieve separation of duties in the administration roles. Instead of provisioning the group private key to a single administrator, it may be split into two or more parts. Adding a user to the group requires each of the administrators to create a decryption entry for the user and for the service and user to apply the appropriate operations to combine the key parts available to them before use.

These techniques could even be extended to support complex authorization requirements such as the need for 2 out of 3 administrators to approve membership of the group. A set of decryption entries is complete if the sum of the key parts is equal to the private key (modulo the order of the curve).

Thus, if the set of administrators is A, B and C and the private key is k, we can ensure that it requires exactly two administrators to create a complete set of decryption entries by issuing key set {a} to A, the key set {k-a, b} to B and the key set {k-a, k-b} to C (where a and b are randomly generated keys).

4.7. Escrow and Recovery of Keys

One of the chief objections made against deployment of Data Level encryption is that although it provides the strongest possible protection of the confidentiality of the data, loss of the decryption keys means loss of the encrypted data. Thus, a robust and effective key escrow mechanism is essential if use of encryption is to ever become commonplace for stored data.

The use of a ‘life-long’ Mesh profiles raises a similar concern. Loss of a Master Signature Key potentially means the loss of the ability to control devices connected to the profile and the accumulated trust endorsements of other users.

Because of these requirements, Mesh users are strongly advised but not required to create a backup copy of the private keys corresponding to their Master Profile Signature and Escrow keys.

Users may use the key escrow mechanism of their choice including the escrow mechanism supported by the Mesh itself which uses Shamir Secret Sharing to escrow the encryption key for a DARE Message containing the private key information.

To escrow a key set, the user specifies the number of key shares to be created and the number required for recovery.

```` Example ArchitectureEscrow ```"
Recovery of the key data requires the key recovery record and a quorum of the key shares:

```` Example ArchitectureRecovery ````

Having recovered the Master Signature Key, the user can now create a new master profile authorizing a new administration device which can be used to authenticate access to the Mesh Service Account(s) connected to the master profile.

5. 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].

6. IANA Considerations

This document does not contain actions for IANA

7. Acknowledgements

Comodo Group: Egemen Tas, Melhi Abdulhayolu, Rob Stradling, Robin Alden.

8. References

8.1. Normative References

[draft-hallambaker-jsonbcd]

[draft-hallambaker-mesh-algorithms]
"[Reference Not Found!!]."

[draft-hallambaker-mesh-dare]

[draft-hallambaker-mesh-developer]
[draft-hallambaker-mesh-platform]

[draft-hallambaker-mesh-protocol]
"[Reference Not Found!]".

[draft-hallambaker-mesh-schema]
"[Reference Not Found!]".

[draft-hallambaker-mesh-security]
"[Reference Not Found!]".

[draft-hallambaker-mesh-trust]

[draft-hallambaker-mesh-udf]

[draft-hallambaker-web-service-discovery]

[FIPS197]  "[Reference Not Found!]".


8.2. URIs


Author’s Address

Phillip Hallam-Baker

Email: phill@hallambaker.com