Network Working Group
Internet-Draft
Intended status: Informational
Expires: 19 July 2020

P. M. Hallam-Baker
16 January 2020

Mathematical Mesh 3.0 Part I: Architecture Guide
draft-hallambaker-mesh-architecture-12

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.

[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-architecture.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 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 19 July 2020.

Table of Contents

1. Introduction 3
2. Definitions 5
2.1. Related Specifications 6
2.2. Defined Terms 6
2.3. Requirements Language 6
2.4. Implementation Status 6
3. Architecture 7
3.1. A Personal PKI 9
3.1.1. Device Management 9
3.1.2. Exchange of trusted credentials. 10
3.1.3. Application configuration management 10
3.1.4. The Mesh as platform 11
3.2. Mesh Architecture 11
3.2.1. Mesh Device Management 12
3.2.2. Mesh Account 13
3.2.3. Mesh Service 15
3.2.4. Mesh Messaging 15
3.3. Using the Mesh with Applications 16
3.3.1. Contact Exchange 17
3.3.2. Confirmation Protocol 17
3.3.3. Future Applications 17
4. Mesh Cryptography 18
4.1. Best Practice by Default 19
4.2. Multi-Level Security 19
4.3. Multi-Key Decryption 19
4.4. Multi-Party Key Generation 20
4.5. Data At Rest Encryption 20
4.5.1. DARE Envelope 21
4.5.2. Dare Container 21
4.6. Uniform Data Fingerprints. 22

4.6.1. Friendly Names 22
4.6.2. Encrypted Authenticated Resource Locators 23
4.6.3. Secure Internet Names 24
4.7. Personal Key Escrow 24
5. User Experience 25

5.1. Creating a Mesh Profile and Administration Device. . . . 26
5.2. Mesh Accounts 27
5.3. Using a Mesh Service 27
5.4. Connecting and Authorizing Additional Devices 28
5.4.1. Direct Connection 29
5.4.2. Pin Connection 29
5.4.3. EARL/QR Code Connection 30
5.5. Contact Requests 31
5.5.1. Remote 32
5.5.2. Static QR Code 32
5.5.3. Dynamic QR Code 33
5.6. Sharing Confidential Data in the Cloud 33
5.7. Escrow and Recovery of Keys 35
6. Security Considerations 36
7. IANA Considerations 36
8. Acknowledgements 36
9. Normative References 36

1. Introduction

The Mathematical Mesh (Mesh) is a user centered Public Key Infrastructure that uses cryptography to make computers easier to use. The Mesh provides an infrastructure that addresses the three concerns that have proved obstacles to the use of end-to-end security in computer applications:

* Device management.

* Exchange of trusted credentials.

* Application configuration management.

The infrastructure developed to address these original motivating concerns can be used to facilitate deployment and use of existing security protocols (OpenPGP, S/MIME, SSH) and as a platform for building end-to-end secure network applications. Current Mesh

applications include:

* Multi-factor authentication and confirmation

* Credential management

* Bookmark/Citation management

* Task and workflow management

A core principle of the design of the Mesh is that each person is their own source of authority. They may choose to delegate that

authority to another to act on their behalf (i.e. a Trusted Third Party) and they may choose to surrender parts of that authority to others (e.g. an employer) for limited times and limited purposes.

Thus, from the user's point of view, the Mesh is divided into two parts. The first and most important of these from their point of view being their own personal Mesh. The second being the ensemble of every Mesh to which their Mesh is connected. As with the Internet, which is a network of networks, a Mesh of Meshes has certain properties that are similar to those of its constituent parts and some that are very different.

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 [draft-hallambaker-mesh-udf]. 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 [draft-hallambaker-mesh-dare]. Describes the cryptographic message and append-only sequence formats used in Mesh applications and the Mesh Service protocol.

IV. Schema Reference [draft-hallambaker-mesh-schema]. Describes the syntax and semantics of Mesh Profiles, Container Entries and Mesh Messages and their use in Mesh Applications.

V. Protocol Reference [draft-hallambaker-mesh-protocol]. Describes the Mesh Service Protocol.

VI. The Trust Mesh [draft-hallambaker-mesh-trust]. 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 [draft-hallambaker-mesh-security] Describes the security considerations for the Mesh protocol suite.

VIII Cryptographic Algorithms [draft-hallambaker-mesh-cryptography]. 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 specification suite:

JSON-BCD Encoding [draft-hallambaker-jsonbcd]. 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
[draft-hallambaker-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 [draft-hallambaker-mesh-developer]. Describes the reference code distribution license terms, implementation status and currently supported functions.

Mesh Platform [draft-hallambaker-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 The RECOMMENDED and REQUIRED cryptographic algorithms for Mesh implementations are specified in [draft-hallambaker-mesh-cryptography].

In addition Mesh Devices used to administer non-Mesh applications must support the cryptographic algorithm suites specified by the application.

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].

The examples in this document were created on 1/6/2020 5:19:39 PM. Out of 170 examples, 70 were not functional.

[Note: Example data is now being produced using the mesh command line tool which is currently substantially less complete than the Mesh reference code it is intended to provide an interface to. As a result, the documentation currently lags the code by more than is usual.]

3. Architecture

The Mathematical Mesh (Mesh) is a user centered Public Key Infrastructure that uses cryptography to make computers easier to use. This document describes version 3.0 of the Mesh architecture and protocols.

For several decades, it has been widely noted 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 that designing applications that respect it. Real users have real work to do and have neither the time nor the inclination to use tools that will negatively impact their performance.

The Mesh is based on the principle that if the Internet is to be secure, if 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.

Today's technology requires users to put their trust in an endless variety of devices, software and services they cannot fully understand let alone control. Even the humble television of the 20^(th) century has been replaced by a 'smart' TV with 15 million lines of code. Whose undeclared capabilities may well include placing the room in which it is placed under continuous audio and video surveillance.

Every technology deployment by necessity requires some degree of trust on the owner/user's part. But this trust should be limited and subject to accountability. If manufacturers continue to fail in this regard, they risk a backlash in which users seek to restore their rights through litigation, legislation or worst of all, simply not buying more technology that they have learned to distrust through their own experience.

The Mesh is based on the principle of radical distrust, that is, if a party is capable of defecting, we assume that they will. As the Russian proverb goes: ???????, ?? ????????: trust, but verify.

In the 1990s, the suggestion that 'hackers' might seek to make financial gains from their activities was denounced as 'fear-mongering'. The suggestion that email or anonymous currencies might

be abused received a similar response. Today malware, ransomware and

spam have become so ubiquitous that they are no longer news unless the circumstances are particularly egregious.

We must dispense with the notion that it is improper or impolite to question the good faith of technology suppliers of any kind whether they be manufacturers, service providers, software authors or reviewers. Modern supply chains are complex, typically involving hundreds if not thousands of potential points of deliberate or accidental compromise. The technology provider who relies on the presumption of good faith on their part risks serious damage to their reputation when others assert that a capability added to their product may have malign uses.

Radical distrust means that we apply the principles of least principle and accountability at every level in the design:

* Cryptographic keys installed in a product during manufacture are only used for the limited purpose of putting that device under control of the user.

* Cryptographic keys and assertions related to management of devices are only visible to the user they belong to and are never exposed to external parties.

* Mesh Accounts belong to and are under control of the user they belong to and not the Mesh Service provider which the user can change at will with minimal inconvenience.

* Mesh Services do not have access to the plaintext of any Mesh Messages or Mesh Catalog data except for the Contacts catalog.

* All Mesh Messages are subject to access control by both the inbound and outbound Mesh Service to mitigate messaging abuse.

Security is risk management and not the elimination of all possibility of any risk. Radical distrust means that we raise the bar for attackers to the point where for most attackers the risk is greater than the reward.

In addition to distrusting technology providers the Mesh Architecture allows the user to limit the degree of trust they place in themselves. In the real world, devices are lost or stolen, passwords and activation codes are forgotten, natural or man-made catastrophes cause property and data to be lost. The Mesh permits but does not require use of escrow techniques that allow recovery from such situations.

3.1. A Personal PKI

The Mesh is a Public Key Infrastructure (PKI) that is designed to address the three major obstacles to deployment of end-to-end secure applications:

* Device management.

* Exchange of trusted credentials.

* Application configuration management.

Each Mesh user is the ultimate source of authority in their Personal Mesh which specifies the set of devices, contacts and applications that they trust and for what purposes.

The Mesh 1.0 architecture described a PKI designed to meet these limited requirements to enable use of existing end-to-end secure Internet protocols such as OpenPGP, S/MIME and SSH. Since these protocols only secure data in transit and the vast majority of data breaches involve data at rest, the Data At Rest Encryption (DARE) was added as a layered application resulting in the Mesh 2.0 architecture. This document describes the Mesh 3.0 architecture which has been entirely re-worked so that DARE provides the platform on which all other Mesh functions are built.

3.1.1. Device Management

Existing PKIs were developed in an era when the 'personal computer' was still coming into being. Only a small number of people owned a computer and an even smaller number owned more than one. Today, computers are ubiquitous and a typical home in the developed world contains several hundred of which a dozen or more may have some form of network access.

The modern consumer faces a problem of device management that is considerably more complex than the IT administrator of a small business might have faced in the 1990s but without any of the network management tools such an administrator would expect to have available.

One important consequence of the proliferation of devices is that end-to-end security is no longer sufficient. To be acceptable to users, a system must be ends-to-ends secure. That is, a user must be able to read their encrypted email message on their laptop, tablet, phone, or watch with exactly the same ease of use as if the mail was

unencrypted.

Each personal Mesh contains a device catalog in which the cryptographic credentials and device specific application configurations for each connected device are stored.

Management of the device catalog is restricted to a subset of devices that the owner of the Mesh has specifically authorized for that purpose as administration devices. Only a device with access to a duly authorized administration key can add or remove devices from a personal Mesh.

3.1.2. Exchange of trusted credentials.

One of the most challenging, certainly the most contentious issues in PKI is the means by which cryptographic credentials are published and validated.

The Mesh does not attempt to impose criteria for accepting credentials as valid as no such set of criteria can be comprehensive. Rather the Mesh allows users to make use of the credential validation criteria that are appropriate to the purpose for which they intend to use them and Mesh Services provides protocol support for exchange of credentials between users and to synchronize credential information between all the devices belonging to a user.

In some circumstances, only a direct trust model is acceptable, in others, only a trusted third-party model is possible and in the vast majority of cases opportunistic approaches are more than sufficient. Both approaches may be reinforced by use of chained notary certificate (e.g. BlockChain) technology affords a means of establishing that a particular assertion was made before a certain date. The management of Trust in the Mesh is described in detail in [draft-hallambaker-mesh-trust].

3.1.3. Application configuration management

Configuration of cryptographic applications is typically worse than an afterthought. Configuration of one popular mail user agent to use S/MIME security requires 17 steps to be performed using four separate application programs. And since S/MIME certificates expire, the user

is required to repeat these steps every few years. Contrary to the public claims made by one major software vendor it is not necessary to perform 'usability testing' to recognize abject stupidity.

Rather than writing down configuration steps and giving them to the user, we should turn them into code and give them to a machine. Users should never be required to do the work of the machine. Nor should any programmer be allowed to insult the user by casting their effort aside and requiring it to be re-entered.

While most computer professionals who are required to do such tasks on a regular basis will create a tool for the purpose, most users do not have that option. And of those who do write their own tools, only a few have the time and the knowledge to do the job without introducing security vulnerabilities.

3.1.4. The Mesh as platform

Meeting the core objectives of the Mesh required new naming, communication and cryptographic capabilities provided to be developed. These capabilities may in turn be used to develop new end-to-end secure applications.

For example, a DARE Catalog is a cryptographic container in which the entries represent a set of objects which may be added, updated and deleted over time. The Mesh Service protocol allows DARE Catalogs to be synchronized between devices connected to a Mesh Account. DARE Catalogs are used as the basis for the device and contacts catalogs referred to above.

The Mesh Credentials Catalog uses the same DARE Catalog format and Mesh Service protocol to share passwords between devices with end-to-end encryption so that no password data is ever left unencrypted in the cloud.

3.2. Mesh Architecture

The Mesh has four principal components:

Mesh Device Management Each user has a personal Mesh profile that is used for management of their personal devices. A user may connect devices to or remove devices from their personal Mesh by use of a

connected device that has been granted the 'administration' role.

Mesh Account A Mesh account is similar in concept to a traditional email or messaging account but with the important difference that it belongs to the user and not a service provider. Users may maintain multiple Mesh accounts for different purposes.

Mesh Service A Mesh Service provides a service identifier (e.g. "alice@example.com") through which devices and other Mesh users may interact with a Mesh Account. It is not necessary for a Mesh Account to be connected to a Mesh Service and users may change their Mesh service provider at any time. It is even possible for a Mesh Account to be connected to multiple services at the same time but only one such account is regarded as the primary account at a given time.

Mesh Messaging Mesh Messaging allows short messages (less than 32KB) to be exchanged between Mesh devices connected to an account and between Mesh Accounts. One of the key differences between Mesh Messaging and legacy services such as SMTP is that every message received is subjected to access control.

A user's Personal Mesh is the set of their Personal Mesh Profiles, Mesh Accounts and the Mesh Services to which they are bound.

For example, Figure X shows Alice's personal Mesh which have separate accounts for her personal and business affairs. She has many devices, two of which are shown here. Both are linked to her personal account but only one is linked to her business account. Besides allowing Alice to separate work and pleasure, this separation means that she does not need to worry about her business affairs being compromised if the device "Alice2" is stolen.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 1

Alice's ProfileMaster contains a Master Signature Key used to sign the profile itself and one or more Administrator Signature Keys used to sign assertions binding devices and/or assertions to her Mesh.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 2

If desired, Alice can escrow the master key associated with her Profile Master and delete it from her device(s), thus ensuring that compromise of the device does not result in a permanent compromise of her personal Mesh. Recovery of the Master Signature Key and the associated Master Encryption Escrow keys (not shown) allows Alice to recover her entire digital life.

To eliminate the risk of hardware failure, the escrow scheme offered by the Mesh itself uses key shares printed on paper and an encrypted escrow record stored in the cloud. Mesh users are of course free to use alternative escrow means of their choice.

3.2.1. Mesh Device Management

Mesh devices are added to or removed from a user's personal Mesh by adding or removing Device catalog entries from the CatalogDevice associated with the Master Profile.

Device catalog entries are created by devices that have been provisioned with an administration key specified in the corresponding ProfileMaster

The keying material used by a device in the context of a user's personal Mesh comes from two separate sources:

* Keying material specified in the ProfileDevice which is either generated on the device itself or installed during manufacture.

* Keying material provided by the Administration Device during the connection process.

This approach mitigates the risk of keying material used by the device being compromised during or after manufacture and the risks associated with use of weak keys. The key combination mechanism is shown in section XX below and described in detail in [draft-hallambaker-mesh-cryptography].

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 3

In accordance with the principle of maintaining cryptographic hygiene, separate keys are generated for signature, authentication and encryption purposes.

3.2.2. Mesh Account

Mesh Accounts comprise a collection of persistent data stores associated with a particular persona associated with a personal Mesh. The connection between a Mesh Account and the personal Mesh to which it belongs may or may not be public. For example, Alice might allow her contacts to know that her business and personal accounts belong to the same personal Mesh and thus the same person but Bob might not.

Mesh Accounts afford similar functionality to the accounts provided by traditional Internet protocols and applications but with the important distinction that they belong to the user and not the service provider. A Mesh Account may be connected to one, many or no Mesh Services and the user may add or delete service providers at any time.

A Mesh Account that is not connected to a Service is called an offline account. Offline accounts cannot send or receive Mesh Messages and cannot be synchronized using the Mesh Service protocol (but may be synchronized through other means).

When a Mesh Account is connected to multiple services, only the first service is normally regarded as being primary with the others being secondary accounts for use in case of need.

Alice's personal account is connected to two devices and two services ("alice@example.com" and "alice@example.net").

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 4

As with the connection of the device to Alice's personal Mesh, the connection of each device to each account requires the creation of a separate set of keying using the same key combination mechanism described above. This information is contained in the ActivationAccount record corresponding to the account in the CatalogEntryDevice.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 5

Note that even though Alice's personal account is connected to two separate Mesh Services, the same cryptographic keys are used for both. However separate keys are used for her personal and business accounts so as to prevent these accounts being linked through use of the same device keys.

3.2.2.1. Account Catalogs

Mesh Catalogs are a DARE Containers 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.

Application

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.

Configuration information for applications including mail (SMTP,

IMAP, OpenPGP, S/MIME, etc) and SSH.

The Devices and Contacts catalogues have special functions in the Mesh as they describe the set of devices and other users that a Mesh user interacts with.

These catalogs are also used as the basis for providing a consistent set of friendly names to the users devices and contacts that is accessible to all her devices. This (in principle) allows Alice to give a voice command to her car or her watch or her phone to call a person or open a door and expect consistent results.

3.2.3. Mesh Service

Each Mesh Service is described by a ProfileService signed by a long-lived signature key. As with the ProfileMaster, a separate set of Administrator keys is used to sign the Assertion Host objects used to credential Service Hosts.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 6

Note that the Mesh Service Authentication mechanism only provides trust after first use. It does not provide a mechanism for secure introduction. A Mesh Service SHOULD be credentialed by means of a validation process that establishes the accountability. For example, the CA-Browser Forum Extended Validation Requirements.

3.2.4. Mesh Messaging

The Mesh Messaging layer supports the exchange of short (less than 32KB) messages. Mesh devices connected to the same Mesh profile may exchange Mesh Messages directly. Messages exchanged between Mesh Users MUST be mediated by a Mesh Service for both sending and receipt. This 'four corner' pattern permits ingress and egress controls to be enforced on the messages and that every message is properly recorded in the appropriate spools.

For example, To send a message to Alice, Bob posts it to one of the Mesh Services connected to the Mesh Account from which the message is to be sent. The Mesh Service checks to see that both the message and Bob's pattern of behavior comply with their acceptable use policy and

if satisfactory, forwards the message to the receiving service example.com.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 7

The receiving service uses the recipient's contact catalog and other information to determine if the message should be accepted. If accepted, the message is added to the recipient's inbound message spool to be collected by her device(s) when needed.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 8

For efficiency and to limit the scope for abuse, all inbound Mesh Messages are subject to access control and limited in size to 32KB 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.

3.3. Using the Mesh with 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:

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 9

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

3.3.1. Contact Exchange

One of the chief concerns in any PKI is the means by which the public keys of other users are obtained and validated. This is of particular importance in the Mesh since every Mesh Message is subject to access control and it is thus necessary for Alice to accept Bob's credentials before Bob send most types of message to Alice.

The Mesh supports multiple mechanisms for credential exchange. If Alice and Bob meet in person and are carrying their smart phones, a secure mutual exchange of credentials can be achieved by means of a QR code mechanism. If they are at separate locations, Alice can choose between accepting Bob's contact information with or without additional verification according to the intended use.

3.3.2. 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.

3.3.3. 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 achieve end-to-end security in 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. Mesh 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.

For example, Alice uses her Desktop and Laptop to exchange end-to-end secure Mesh Messages and documents but her Internet-of-Things food blender and light bulb are limited in the range of functions they support and the telemetry information they provide. The IoT devices connect to a Mesh Hub which acts as an always-on point of presence for the device state and allows complex cryptographic operations to be offloaded if necessary.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 10

4.1. Best Practice by Default

Except where support for 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.

4.2. 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.

4.3. Multi-Key Decryption

Traditional public key encryption algorithms have two keys, one for encryption and another for decryption. The Mesh makes use of threshold cryptography techniques to allow the decryption key to be split into two or more parts.

For example, if we have a private key _z_, we can use this to perform a key agreement with a public key _S_ to obtain the key agreement value A. But if _z_ = (_x+y_) mod _g_ (where g is the order of the group). we can obtain the exact same result by applying the private keys _x_ and _y_ to _S_ separately and combining the results:

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 11

The approach to Multi-Key Decryption used in the Mesh was originally inspired by the work of Matt Blaze et. al. on proxy re-encryption. But the approach used may also be considered a form of Torben Pedersen's Distributed Key generation.

This technique is used in the Mesh to allow use of decryption key held by a user to be controlled by a cloud service without giving the cloud service the ability to decrypt by itself.

4.4. Multi-Party Key Generation

The mathematics that support multi-key decryption are also the basis for the multi-party key generation mechanism that is applied at multiple levels in the Mesh. The basis for the multi-party key generation used in the Mesh is that for any Diffie-Hellman type cryptographic scheme, given two keypairs { _x_, _X_ }, { _y_, _Y_ }, we calculate the public key corresponding to the private key _x_+ _y_ using just the public key values _X_and _Y_.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 12

Multi-party key generation ensures that keys used to bind devices to a personal Mesh or within a Mesh account are 'safe' if any of the contributions to the generation process are safe.

4.5. 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.

4.5.1. DARE Envelope

The DARE Envelope 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 Envelopes.

A feature of the DARE Envelope 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.

4.5.2. Dare Container

A DARE Container is an append-only sequence of DARE Envelopes. 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 Envelope.

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.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 13

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.

Three types of DARE Containers are used in the mesh

Catalogs A DARE Container whose entries track the status of a set of related objects which may be added, updated or deleted.

Spools A DARE Container whose entries track the status of a series of Mesh Messages.

Archives A DARE Container used to provide a file archive with optional confidentiality and/or integrity enhancements.

4.6. 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:

```
NDDI-FY6J-IOGW-Z6WZ-QBKV-FDNY-ALDQ
EA3P-NYHM-5E36-QNRK-NDB3-WJT6-YHCA
SAQB-TIMR-OIPM-CVGV-3WQA-726M-WLDR-6
MB5S-R4AJ-3FBT-7NHO-T26Z-2E6Y-WFH4
KCM5-7VB6-IJXJ-WKHX-NZQF-OKGZ-EWVN
AA7B-A4WX-KRJE-LVPL-XEDA-36PQ-RRQ3
```

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.

4.6.1. 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 Device 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.

4.6.2. 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 an identifier which allows an encrypted data object to be retrieved and decrypted. In this case, the encrypted data object contains the information needed to complete the interaction.

An EARL contains the domain name of the service providing the resolution service and an encryption master key:

udf://example.com/EC4X-PWKB-JNOA-FRW7-DBBT-ZAYU-3EVA-FR

The EARL may be expressed as a QR code:

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 14

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 Envelope 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 encryption key in the QR code can decrypt the message.

4.6.3. 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 a SIN-aware 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.

alice@example.com Alice's regular email address (not a SIN).

alice@mm--uuuu-uuuu-uuuu.example.com A strong email address for Alice that can be used by a regular email client.

alice@example.com.mm--uuuu-uuuu-uuuu A strong email address for Alice that can only used by an email client that can process SINs.

Using an email address that has the Security Policy element as a prefix allows a DNS wildcard element to be defined that allows the address to be used with any email client. Presenting the Security Policy element as a suffix means it can only be resolved by a SIN-aware client.

4.7. 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.

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.

The Mesh supports use of Shamir Secret Sharing to split a secret key into a set of shares, a predetermined number of which may be used to recover the original secret. For convenience secret shares are represented using UDF allowing presentation in Base32 (i.e. text format) for easy transcription or QR code presentation if preferred.

A Mesh Profile is escrowed by creating a recovery record containing the private keys corresponding to the master signature and master escrow keys associated with the profile. A master secret is then generated which is used to generate a symmetric encryption key used to encrypt the recovery record and to generate the desired number of recovery shares. For example, Alice escrows her Mesh Profile creating three recovery shares, two of which are required to recover the master secret:

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 15

To recover the master secret, Alice presents the necessary number of key shares. These are used to recover the master secret which is used to generate the decryption key.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 16

A user may choose to store their encrypted recovery record themselves or make use of the EARL mechanism to store the information at one or more cloud services using the fingerprint of the master secret as the locator.

5. 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).

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 17

5.1. Creating a Mesh Profile and Administration Device.

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.

```
Alice> mesh create
Device Profile UDF=MBPL-MIIT-KOHN-FC6P-6V5Y-ZQ4R-3NKY
Personal Profile UDF=MCSC-2POG-PH7T-ODJX-HOCA-B4XY-AFSK
```

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.

5.2. Mesh Accounts

Add an account to the personal Mesh:

```
Alice> account create personal
Account=MC23-X3CT-EUNB-DR3M-QIBI-W2IJ-ATEP
```

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:

```
Alice> password add ftp.example.com alice1 password
alice1@ftp.example.com = [password]
Alice> password add www.example.com alice@example.com newpassword
alice@example.com@www.example.com = [newpassword]
Alice> password list
alice1@ftp.example.com = [password]
alice@example.com@www.example.com = [newpassword]
Alice> password add ftp.example.com alice1 newpassword
alice1@ftp.example.com = [newpassword]
Alice> password get ftp.example.com
alice1@ftp.example.com = [newpassword]
```

5.3. Using a 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:

```
Alice> account register alice@example.com
Account=MC23-X3CT-EUNB-DR3M-QIBI-W2IJ-ATEP
```

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.

5.4. 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.

5.4.1. 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.

The connection request is initiated on the device being connected:

```
Alice2> device request alice@example.com
   Witness value = NGPY-QAYV-OCQD-W6JD-U3HP-3OAQ-57OW
   Personal Mesh = MCSC-2POG-PH7T-ODJX-HOCA-B4XY-AFSK
```

Using her administration device, Alice gets a list of pending requests. Seeing that there is a pending request matching the witness value presented by the device, Alice accepts it:

```
Alice> device pending
Alice> device accept NCPO-L452-CMZY-MLQO-TM52-KQYW-EFGP
```

The new device will now synchronize automatically in response to any Mesh commands. For example, listing the password catalog:

```
Alice2> password list
ERROR - Object reference not set to an instance of an object.
```

5.4.2. 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.

The PIN connection mechanism begins with the issue of the PIN:

```
Alice> account pin
PIN=NCAX-BVXY-5O3Y-CPZS-OU (Expires=2020-01-07T17:19:33Z)
```

The PIN code is transmitted out of band to the device being connected:

```
Alice3> device request alice@example.com /pin=NCAX-BVXY-5O3Y-CPZS-OU
   Witness value = RDPZ-WAM2-QD3P-YNJE-R3BV-NXOF-OEFC
   Personal Mesh = MCSC-2POG-PH7T-ODJX-HOCA-B4XY-AFSK
```

Since the request was pre-authorized, it is not necessary for Alice to explicitly accept the connection request but the administration device is needed to create the connection assertion:

```
Alice> device pending
```

We can check the device connection by attempting to synchronize to the profile account:

```
Alice3> account sync
ERROR - Object reference not set to an instance of an object.
```

Note that this connection mechanism could be addapted to allow a device with a camera affordance to connect by scanning a QR code on the administration device.

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.

5.4.3. 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.

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:

To use the device QR code connection mechanism, we require a Web service that will host the connection document example.com and a MeshService account that the device will attempt to complete the connection by requesting synchronization devices@example.com.

To begin the process we generate a new random key and combine it with the service to create an EARL:

```
udf://example.com/EC4X-PWKB-JNOA-FRW7-DBBT-ZAYU-3EVA-FR
```

Next a device profile is created and preregistered on with the Mesh Service that will provide the hailing service. Since we are only preparing one device it is convenient to do this on the device

itself. In a manufacturing scenario, these steps would typically be performed offline in bulk.

```
Alice4> device pre devices@example.com /key=udf://example.com/EC4X-PW
    KB-JNOA-FRW7-DBBT-ZAYU-3EVA-FR
ERROR - Object reference not set to an instance of an object.
```

Once initialized the device attempts to poll the service for a connection each time it is powered on, when a connection button affordance on the device is pressed or at other times as agreed with the Mesh Service Provider:

```
Alice4> account sync
ERROR - Object reference not set to an instance of an object.
```

To connect the device to her profile, Alice scans the device with her administration device to obtain the UDF. The administration device retrieves the connection description, decrypts it and then uses the information in the description to create the necessary Device Connection Assertion and connect to the device hailing Mesh Service Account to complete the process:

```
Alice> device earl udf://example.com/EC4X-PWKB-JNOA-FRW7-DBBT-ZAYU-3E
    VA-FR
ERROR - Object reference not set to an instance of an object.
```

When the device next attempts to connect to the hailing service, it receives the Device Connection Assertion:

```
Alice4> account sync
ERROR - Object reference not set to an instance of an object.
```

5.5. 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:

Alice creates a contact entry for herself:

```
Alice> contact self email alice@example.com
{
  "Self": true,
  "Key": "NDMW-OM2B-66IV-DWFH-MQXO-X322-IP74",
  "EnvelopedContact": [{},
    "ewogICJDb250YWN0IjogewogICAgIkFkZHJlc3Nlcy
  I6IFt7CiAgICAgICAgIlVSSSI6ICJtYWlsdG86e2VtYWlsfSJ9XX19"]}
```

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.

5.5.1. Remote

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

Bob requests Alice add him to her contacts catalog:

```
Bob> message contact alice@example.com
```

When Alice next checks her messages, she sees the pending contact request from Bob and accepts it. Bob's contact details are added to her catalog and Bob receives a response containing Alice's credentials:

```
Alice> message pending
Alice> message accept tbs
```

5.5.2. 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. The resulting QR code may be printed on a business card, laser engraved on a luggage tag, etc.

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.

5.5.3. 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.

5.6. Sharing Confidential Data in the Cloud

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. This provides data encryption capabilities that are particularly suited to 'cloud computing' environments.

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.

(Artwork only available as svg: No external link available, see draft-hallambaker-mesh-architecture-12.html for artwork.)

Figure 18

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.

Alice creates the recryption group groupw@example.com to share confidential information with her closest friends:

```
Alice> group create groupw@example.com
{
  "Profile": {
    "KeyOfflineSignature": {
      "UDF": "MB2A-I332-A75U-OHD7-WLHR-XRZK-CLRU",
      "PublicParameters": {
        "PublicKeyECDH": {
          "crv": "Ed448",
          "Public": "V9z0yscYdi-vi2cqWrehpdZhy45EjB-AIaXWC7iIVtmHckZA
              xSlK
  pDd-vDVDwBgF50egU0gEvMUA"}}},
```

```
    "KeyEncryption": {
      "UDF": "MB2W-EYG3-EFXZ-QGEL-BCFJ-BA76-IF6H",
      "PublicParameters": {
        "PublicKeyECDH": {
          "crv": "Ed448",
          "Public": "8XTv_F07n1odENFoyPvc8gXF95iZoOp1sLZ3HeO4jwxy42Qe
              TfE8
  2HtQWT59vPV5X8uSg0iKdloA"}}}}}
```

Bob encrypts a test file but he can't decrypt it because he isn't in the group:

```
Bob> dare encodeTestFile1.txt /out=TestFile1-group.dare /encrypt=grou
    pw@example.com
ERROR - The command  is not known.
Bob> dare decode  TestFile1-group.dare
ERROR - Could not find file 'C:\Users\hallam\Test\WorkingDirectory\Te
    stFile1-group.dare'.
```

Since she is the group administrator, Alice can decrypt the test file using the group decryption key:

```
Alice> dare decode  TestFile1-group.dare
ERROR - Could not find file 'C:\Users\hallam\Test\WorkingDirectory\Te
    stFile1-group.dare'.
```


Adding Bob to the group gives him immediate access to any file encrypted under the group key without making any change to the encrypted files:

```
Alice> dare decode  TestFile1-group.dare
ERROR - Could not find file 'C:\Users\hallam\Test\WorkingDirectory\Te
    stFile1-group.dare'.
```

Removing Bob from the group immediately withdraws his access.

```
Alice> group delete groupw@example.com bob@example.com
ERROR - The feature has not been implemented
```

Bob cannot decrypt any more files (but he may have kept copies of files he decrypted earlier).

```
Alice> dare decode  TestFile1-group.dare
ERROR - Could not find file 'C:\Users\hallam\Test\WorkingDirectory\Te
    stFile1-group.dare'.
```

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).

5.7. 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 Envelope 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.

```
Alice> mesh escrow
ERROR - The cryptographic provider does not permit export of the priv
    ate key parameters
```

Recovery of the key data requires the key recovery record and a quorum of the key shares:

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.

6. 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]

7. IANA Considerations

This document does not contain actions for IANA

8. Acknowledgements

Comodo Group: Egemen Tas, Melhi Abdulhayo?lu, Rob Stradling, Robin Alden.

9. Normative References

[draft-hallambaker-jsonbcd]

Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", Work in Progress, Internet-Draft, draft-hallambaker-jsonbcd-15, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-jsonbcd-15>.

[draft-hallambaker-mesh-cryptography]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VIII: Cryptographic Algorithms", Work in Progress, Internet-Draft, draft-hallambaker-mesh-cryptography-04, 1 November 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-cryptography-04>.

[draft-hallambaker-mesh-dare]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part III : Data At Rest Encryption (DARE)", Work in Progress, Internet-Draft, draft-hallambaker-mesh-dare-05, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-dare-05>.

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-hallambaker-mesh-developer-09, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-developer-09>.

[draft-hallambaker-mesh-platform]
Hallam-Baker, P., "Mathematical Mesh: Platform Configuration", Work in Progress, Internet-Draft, draft-hallambaker-mesh-platform-05, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-platform-05>.

[draft-hallambaker-mesh-protocol]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part V: Protocol Reference", Work in Progress, Internet-Draft, draft-hallambaker-mesh-protocol-03, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-protocol-03>.

[draft-hallambaker-mesh-schema]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part IV: Schema Reference", Work in Progress, Internet-Draft, draft-hallambaker-mesh-schema-04, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-schema-04>.

[draft-hallambaker-mesh-security]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VII: Security Considerations", Work in Progress, Internet-Draft, draft-hallambaker-mesh-security-02, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-security-02>.

[draft-hallambaker-mesh-trust]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VI: The Trust Mesh", Work in Progress, Internet-Draft, draft-hallambaker-mesh-trust-03, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-trust-03>.

[draft-hallambaker-mesh-udf]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part II: Uniform Data Fingerprint.", Work in Progress, Internet-Draft, draft-hallambaker-mesh-udf-08, 6 January 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-udf-08>.

[draft-hallambaker-web-service-discovery]
Hallam-Baker, P., "DNS Web Service Discovery", Work in Progress, Internet-Draft, draft-hallambaker-web-service-discovery-03, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-web-service-discovery-03>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/rfc/rfc7159>.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/rfc/rfc7231>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/rfc/rfc7516>.