Workgroup: Network Working Group
Internet-Draft:
draft-hallambaker-mesh-architecture
Published: 2 November 2020
Intended Status: Informational
Expires: 6 May 2021
Authors: P. M. Hallam-Baker
ThresholdSecrets.com

Mathematical Mesh 3.0 Part I: Architecture Guide

Abstract

The Mathematical Mesh is a Threshold Key Infrastructure that makes computers easier to use by making them more secure. Application of threshold cryptography to key generation and use enables users to make use of public key cryptography across multiple devices with minimal impact on the user experience.

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 6 May 2021.

Copyright Notice

Table of Contents

1. Introduction

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

In 1977, Public Key cryptography laid out a powerful proposition: If Alice and Bob have private keys on their devices and each knows the public key of the other, Alice and Bob can communicate with confidentiality and integrity. The realization of this proposition at Internet scale was vested in a technology called Public Key Infrastructure (PKI) whose principal function is to provide a trustworthy means by which Alice and Bob can discover each other's public key.

Yet despite the power of PKI, Internet security remains a work in progress. While PKI has proved an effective means of authenticating services to users, attempts to apply PKI to the equally important task of authenticating users to services and securing data at rest have been confined to the margins. One critical reason for that failure is that *Public* Key Infrastructure has only provided effective tools for managing *public* keys. If we are to achieve comprehensive Internet security, we must provide every user with the ability to manage private keys across their devices with zero effort on their part.

Threshold cryptography is a sub-field of public key cryptography that defines operations on cryptographic keys, including operations on private keys. Threshold cryptography allows Key generation and key use operations may be split between multiple devices. These tools make zero effort management of private keys practical.

The Mesh is a TKI that addresses the three principal 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 *autonomy*. That is each user has full control over their digital environment and 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) without surrendering their autonomy. Delegation of authority is always for limited times and limited purposes.

Thus, from the user's point of view, the Mesh is divided into two parts: The part of the Mesh that belongs to them and everything else. 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 quite 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 Mesh Discovery Service [draft-hallambaker-mesh-discovery]. Describes the Mesh Discovery Service that supports mapping of Mesh names to the corresponding Mesh Service Provider.

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:

IX. 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.

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.

Threshold Modes in Elliptic Curves [draft-hallambaker-threshold]. Describes threshold key generation and key agreement operations for the Ed25519, Ed448, X25519 and X448 elliptic curves.

Threshold Signatures in Elliptic Curves [draft-hallambaker-threshold-sigs]. Describes creation of threshold signatures using the Ed25519 and Ed448 elliptic curves.

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 11/2/2020 4:47:04 PM. Out of 51 examples, 16 failed.

3. Requirements

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

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 as best as possible without unnecessary inconvenience to the user, we must reverse the question and ask how much security can be provided without requiring any effort whatsoever from the user. This principle is called **Zero Effort Security**.

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 20th 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 not compromise the user's autonomy. Delegation of 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. In 1949, Edward A. Murphy Jr. proposed his now eponymous law which states, 'Anything that can go wrong will go wrong'. We must now apply a similar principle to Internet security: 'Anything that can be made to go wrong is already being made to go wrong and will only get worse until something is done to stop it.'

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 to the design of the Mesh:

- 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 threshold catalog used by the service as the source of access control policy.

- 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. It does not demand that we immediately address every issue with perfection or delay deployment of technologies that are capable of controlling *many* risks until we have achieved the control of *every* risk.

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. The Device Management Challenge

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. In these circumstances, it arguably sufficed to provision a user with a single private key on the single device they were likely to use.

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 were unencrypted. A cryptographic security control that impedes the user is a control that is not going to be used.

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.

3.2. 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. Here there are two different challenges.

Developing an infrastructure that provides a mapping to a cryptographic key from a name that serves no other purpose than identifying the key is relatively easy. Developing an infrastructure that maps existing names with semantics that are already established is considerably harder.

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 provides an internal trust infrastructure that makes use of a *direct trust* model similar to that of PGP fingerprints to which external names may be mapped using whatever validation criteria users consider are appropriate to the purpose for which they intend to use them.

The principles of providing extended trust management in the Mesh are further described in [draft-hallambaker-mesh-trust].

3.3. 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.

3.4. 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, the Mesh Catalogs used to maintain collections of device descriptions, bookmarks, credentials, etc. might be used in an electronic records infrastructure to maintain chain of custody of digital evidence.

3.5. Security

The Mesh is designed to provide the greatest practical level of security that does not detract from the user experience. The usual CIA triad is considered:

Confidentiality The confidentiality of user content should be protected at all times and against all unauthorized parties including their MSP.

Reasonable efforts should be taken to protect user data against traffic analysis and metadata attacks. It is not necessary to consider disclosure of this information to MSPs. Metadata must be shielded from external parties but controls to prevent traffic analysis may be left to implementers.

Integrity
The design should consider unauthorized modification of data to be at least as serious as disclosure.

Availability The design should consider loss of data likely to be at least as serious as disclosure.

In addition to protecting the user's data, the Mesh is designed to protect the user's autonomy. While the use of any electronic device or service entails a degree of trust, the user should have the right to decide which devices and which service providers to trust and to have the practical ability to revoke that trust at any time they choose.

3.6. Enterprise Deployment

Development of PKI has traditionally focused on the needs of large enterprises. The Mesh is focused on the individual user. While this change of focus is in part a recognition of the need to reverse the traditional bias, it is also a recognition of the fact that we must understand the needs of the individual user before attempting to understand the additional needs of an enterprise IT department serving a large number of users.

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 further details and additional examples.

For clarity and compactness of exposition, these use cases are illustrated using the command line tool *meshman*, a tool that makes the cryptographic operations explicit. This does not represent the ideal user experience in which Zero-effort security is achieved. Such a user experience requires that the Mesh operations be seamlessly integrated into the user's applications so that instead of using the meshman tool to encrypt or decrypt document, the word processor application itself would be extended to read and write documents encrypted in the DARE format.

4.1. Creating a Mesh Account

From the user's perspective, their personal Mesh consists of a collection of devices that communicate seamlessly and securely through a Mesh account serviced by a Mesh Service Provider (MSP).

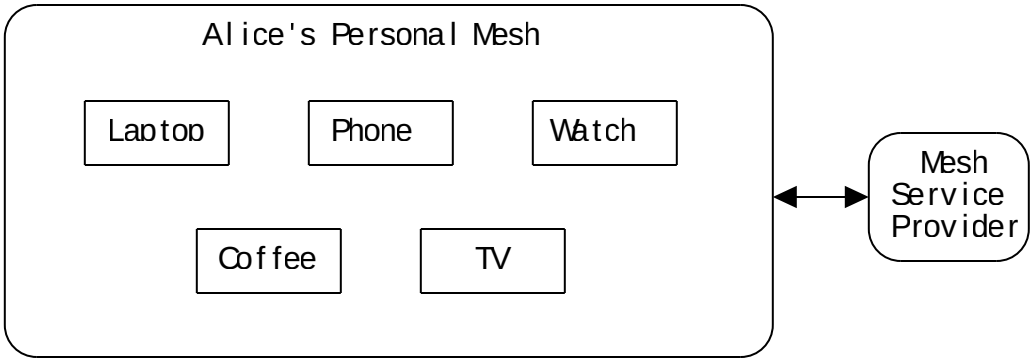

Figure 1

As with an email service provider, the user is only likely to be aware of their interactions with their MSP in the case of a service interruption. As far as the user is concerned the data is replicated across their devices automatically unless there is a problem.

While the term 'account' is used because it is the term a user is familiar with that most closely describes its functions, Mesh accounts are different from traditional Internet accounts in one important respect: In order to realize the principle of 'autonomy', Mesh accounts are created by and *belong to* the user and not the service provider. Should a serious problem occur, a user may opt to change their MSP. But unlike a changing an SMTP email provider, this change is made seamless and cost free.

Another important difference between the Mesh and SMTP is that all Mesh data is encrypted end to end. The MSP does not have access to any user content and does not have access to any user meta-data except that which is strictly necessary to service the account.

The only Mesh catalogs associated with a Mesh account that can be read by an MSP are the Access Catalog which serves as the basis for specifying and enforcing access control policy on the resources associated with the account and the Publications Catalog which is an index of encrypted data published through the account.

To create a Mesh account, the user need only specify the account name and the initial MSP:

The user specifies the initial account address to be used (alice@example.com). Use of this address is of course dependent on authorization by the Mesh Service Provider (example.com) and is likely to require authentication and possibly payment.

```
Alice> account create alice@example.com
Account=MCQ4-CSYK-2LAY-3XXW-72CL-6P65-X6CQ
```

The command returns the value of Alice's Mesh Account fingerprint . This value is used as a unique identifier that is cryptographically bound to the signature key used to authenticate the account profile.

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.

4.1.1. Encrypting and Decrypting files.

Having created an account, Alice can use it to encrypt files and decrypt them on the same machine.

Alice encrypts the text file plaintext.txt to create an encrypted version readable only by Alice:

```
Alice> type plaintext.txt
This is a test
Alice> dare encode plaintext.txt ciphertext.dare /encrypt ^
    alice@example.com
Alice> dare verify ciphertext.dare
File: ciphertext.dare
    Bytes: 0
    Encryption Algorithm: A256CBC
        Recipient: MBNX-5MOX-L2P6-6B33-QAU4-V3Y3-3ZM4
    Digest Algorithm: S512
    Payload Digest:
```

Alice can recover the file at any time using the decryption command:

```
Alice> dare decode ciphertext.dare plaintext1.txt
Alice> type plaintext1.txt
This is a test
```

Although the encrypted file can be accessed by Alice with precisely the same ease as the plaintext version, the contents of the encrypted file are not readable by any other user of the machine unless Alice explicitly grants access. The encrypted file may be stored on a shared drive, cloud file system or removable storage without disclosing the contents.

While encrypting and decrypting files using a tool provides the desired functionality, it does not meet our objectives for usability. These capabilities should be integrated into applications or the platform itself.

4.1.2. Catalogs

Every Mesh account is created with a set of catalogs and spools. For example, the bookmarks catalog maintains a list of the user's Web bookmarks. The credentials catalog maintains a list of the user's usernames and passwords for the various network services they use. As with the file encryption example, these capabilities are clearly going to be most effective when incorporated into the user's applications, (i.e. their Web browser).

Alice adds the username and password she uses to access her weather service account to her credentials catalog:

```
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]
```

As with all Mesh Catalogs, the catalog data is encrypted and cannot be accessed by any unauthorized party including the Mesh Service Provider.

If needed, she can retrieve the credentials from the catalog by specifying the network resource to which access is required:

```
Alice> password get ftp.example.com
alice1@ftp.example.com = [password]
```

This capability provides a means of preventing one of the most common causes of enterprise password breach in which a system administrator encodes the access credentials for a service into a script used to access the service. A script containing a command to extract the credentials from a Mesh catalog will only work for a user authorized to access the credentials in the Mesh.

4.2. Adding devices

Computers have become ubiquitous and inexpensive. Most people living in affluent countries interact with several dozen computer systems every day. Every household appliance from the television to the coffee pot has become or is in the process of becoming a computer.

It is this circumstance that has exposed the critical flaw in traditional PKI: The lack of practical means of managing private keys across multiple devices.

The Mesh allows users to connect all their devices together so that they may be considered part of a single entity whose component parts communicate and interact seamlessly and securely.

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. The Mesh allows the principle of least privilege when connecting a device granting precisely the set of capabilities required to perform its intended function.

Multiple connection mechanisms are specified, each of which provides strong mutual authentication. In each case, the connection request must be approved by a device provisioned with the Mesh administration privilege:

Direct The connection request is initiated on the device being requested and approved on the administration device. Authentication of the connection request is performed by comparing witness values presented on the connecting device and the administration device.

PIN A PIN code is generated on an administration device and passed to the connecting device out of band. The connecting device provides proof of knowledge of this PIN code when making the connection request allowing an administration device to approve the request automatically without further user interaction.

Dynamic QR This connection mechanism is a variation of the PIN connection mechanism in which administration device presents the PIN code value to the connecting device in the form of a QR code. This allows a connecting device with a camera to connect with minimal user effort.

Static QR This connection method is designed to support connection of constrained IoT devices that lack a camera or display capability. An administration device equipped with a camera reads a static QR code printed on the device that provides the information used to enable the administration device to establish a local network connection (e.g. WiFi, Bluetooth, strobe, IR) that can be used to complete the connection.

For example, Alice connects a second device using the direct connection mechanism:

The connection request is initiated on the device being connected:

```
Alice2> device request alice@example.com
   Device UDF = MCR5-EJWB-KRGF-3SYW-ASKC-DWUP-FIQQ
   Witness value = RQUH-LNHP-XVR6-UHQ5-WSCZ-WCZC-Q5PK
```

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> message pending
MessageID: DGVB-YMZP-LJCN-XY3D-LGWE-2G33-WXFE
        Connection Request::
        MessageID: DGVB-YMZP-LJCN-XY3D-LGWE-2G33-WXFE
        To:  From:
        Device:  MDJ2-URNT-WXPZ-5PLB-J3XM-MJK5-AEWG
        Witness: DGVB-YMZP-LJCN-XY3D-LGWE-2G33-WXFE
MessageID: NBAC-LLBY-E4EU-7ZRF-I47O-ZYHZ-PBCW
        Confirmation Request::
        MessageID: NBAC-LLBY-E4EU-7ZRF-I47O-ZYHZ-PBCW
        To: alice@example.com From: console@example.com
        Text: start
MessageID: NBKU-OVBZ-YZRN-FEB4-ARMW-VUVI-2JSG
        Contact Request::
        MessageID: NBKU-OVBZ-YZRN-FEB4-ARMW-VUVI-2JSG
        To: alice@example.com From: bob@example.com
        PIN: AD6Q-2HLS-M3HL-MYNB-43SW-OXCM-QFSA
Alice> account sync /auto
ERROR - Cannot access a closed file.
Alice> device accept RQUH-LNHP-XVR6-UHQ5-WSCZ-WCZC-Q5PK
ERROR - Cannot access a closed file.
```

Alice can now synchronize her newly connected device to her account:

```
Alice2> device complete
```

These connection mechanisms are described in detail in the Mesh Protocol Reference [draft-hallambaker-mesh-protocol].

4.2.1. Decrypting files on the new device

Having connected a second device and granted it 'Web' rights, Alice can use it to decrypt files and access her bookmark and password catalogs in exactly the same fashion as the first. If a password is changed on one device, all her connected devices receive the update.

For example, Alice can now decrypt the file she encrypted on her first device and access her credential catalog from the new device:

```
Alice2> password get ftp.example.com
ERROR - No decryption key is available
Alice2> dare decode ciphertext.dare plaintext2.txt
ERROR - No decryption key is available
```

Should the new device be lost, stolen or simply broken, Alice can prevent further use of the device to decrypt her data by disconnecting it from her Mesh:

Alice disconnects the new device:

```
Alice> device delete TBS
ERROR - The feature has not been implemented
```

The device can no longer access the password catalog:

```
Alice2> dare decode ciphertext.dare plaintext2.txt
ERROR - No decryption key is available
```

The use of threshold decryption allows the Mesh Service Provider to control the use of decryption by Alice's devices without having the ability to decrypt the content itself.

4.2.2. Applications

Connected devices can also make use of connected applications for which they are granted the necessary rights.

Alice creates an SSH profile within her Mesh on the administrative device.

Missing example 1

After configuring an SSH server to accept her new SSH credential, she can use any of her devices that has been granted the SSH right to connect to it.

In this case Alice has chosen to use an SSH configuration in which a single client key is shared across multiple devices. The Mesh is in principle capable of supporting more sophisticated configurations in which use of the client key is under control of a threshold service and/or each device has its own individual private. Consideration of these configuration modes is currently outside the scope of work for the Mesh and is probably more usefully considered as part of an effort to integrate Mesh functionality into the SSH system. This would also allow support for features such as recording SSH server key fingerprints in the Mesh Contacts catalog.

Alice could enable use of OpenPGP and S/MIME on her connected devices that have been granted the messaging right in a similar way. All the network and security configuration data required to use one of her email accounts is stored in her Mesh applications catalog. The Mesh client performs all the steps required to obtain and install CA issued certificates. As with the SSH example, while it is quite possible to support all the necessary functionality through the Mesh alone, a better result is likely to be achieved by modifying the SMTP email clients and Certificate Authority infrastructures.

4.3. Mesh Messaging

The Mesh Messaging system is a push messaging system analogous to SMTP, but its purpose is limited to secure exchange of control plane messages. This leads to some important differences:

- Every message is signed and end-to-end encrypted
- The only communication pattern supported is a four-corner model in which users exchange messages through their respective MSPs.
- Every message is subject to access control at the inbound and outbound MSP.
- Message content is limited to 32KB.

This size restriction ensures that exchange of Mesh Messages does not impose an undue burden on the inbound and outbound MSP. It is not necessary for a sender to transfer multiple MB message before the receiver decides to refuse it for some reason. Connected devices may efficiently synchronize their message spools even over limited bandwidth connections. A short message is never blocked by a larger one.

Should exchange of longer messages be desired, a pull model is employed. A Mesh message is used to send a message advising the recipient's client of the location from which the full content may be obtained. This approach has many benefits over the SMTP push model. There is no longer a need for any limitation on message size. The same messaging platform can be used to send a short text message, a spreadsheet or raw video files.

Exchange of certain content types naturally leads to security concerns. These concerns are mitigated in the Mesh by performing access control on every message. When accepting Bob as a partner, Alice can choose the types of Mesh Message and the types of content she is willing to accept from him. Thus, Alice might be willing to accept a spreadsheet containing macro code from Bob but not from Carol or Mallet. And she might not want to accept anything at all from Susan because of past abuse.

While there are important technical differences between Mesh Messaging and SMTP, these are not visible to Alice or Bob except insofar as there is no restriction on message size other than the storage capacity of the machine they wish to receive the messages on, there is very little scope for messaging abuse and (unless the Mesh becomes ubiquitous) they can only use Mesh Messaging to communicate with other Mesh users. Thus, while Mesh messaging has been designed to enable replacement of SMTP in the long term, it is not currently a focus for the client implementations. Use of Mesh messaging is thus currently limited to support for applications built on the Mesh platform. One of those applications is the device connection protocol describe earlier. Another is the contact exchange protocol used to acquire contact information from other Mesh users.

4.3.1. Contact exchange

Besides management of private keys across devices, the biggest obstacle to effective use of existing security protocols such as SSH, OpenPGP and S/MIME is the difficulty of obtaining the authentic public keys of the counterparties.

The question of issue and validation of credentials is a complex and difficult one that does not have a single answer that is valid for every use case. For certain applications credentials issued by a Trusted Third Party are appropriate. For others, the Web of Trust proposed in OpenPGP provides a better fit to the requirements and constraints. These issues are discussed in [draft-hallambaker-mesh-trust].

Rather than imposing a single trust model for credential acquisition, the Mesh allows the use of whatever model is best for

validating a credential for a particular use. It is unlikely Alice would have the same security concerns for communication with her employer, her friends, her bank, etc.

For many applications, Trust After First Use provides an adequate basis for credential acquisition.

Alice wants to exchange Mesh messages with Bob. Although Alice knows Bob's Mesh address (bob@example.com), she does not (yet) have permission to send any message to Bob excepting a request to exchange contact information.

Bob sends Alice a contact exchange request:

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

Alice checks his Mesh messages and approves Bob's request:

```
Alice> account sync
Alice> message pending
MessageID: NBKU-OVBZ-YZRN-FEB4-ARMW-VUVI-2JSG
        Contact Request::
        MessageID: NBKU-OVBZ-YZRN-FEB4-ARMW-VUVI-2JSG
        To: alice@example.com From: bob@example.com
        PIN: AD6Q-2HLS-M3HL-MYNB-43SW-OXCM-QFSA
Alice> message accept NBKU-OVBZ-YZRN-FEB4-ARMW-VUVI-2JSG
ERROR - Cannot access a closed file.
```

At this point Alice and Bob can exchange Mesh messages of any type with seamless end to end security. Every Mesh message is signed and encrypted without exception. If Alice and Bob have used the Mesh to configure their email accounts for OpenPGP or S/MIME, they can use these to exchange end-to-end secure SMTP mail.

Alternatively, Bob might have opted to grant Alice only specific messaging access. Bob might choose to restrict synchronous messaging modalities such as instant messaging or voice that interrupt his workflow to specific colleagues. The fact that Alice wants to speak to Bob does not necessarily mean she is interested in what he might say in reply. Thus, messaging access need not be reciprocated.

As with device connection, multiple contact exchange methods are supported including the use of a QR code printed on a business card or presented on a mobile device. These methods are also described in [draft-hallambaker-mesh-protocol].

As a respected figure within the cryptographic community, Alice might employ a curation service for credential requests advising her that Bob's credentials appear to be in order while Mallet's are suspicious. Such services might be offered by her MSP or another provider. Alice might be willing to accept contact requests from members of professional associations she is a member of or who have attended certain conferences in her field. A variety of approaches might be followed for curation of other requests including Machine Learning approaches.

4.3.2. Confirmation service

The Mesh confirmation service is an improvement of traditional second factor authentication techniques offering offers far greater usability and security.

Instead of being asked to present a meaningless numeric code, Alice is presented a request from a named, authenticated source to confirm a specific action. Alice's response will be signed using a signature key that is unique to the particular confirmation device she uses, thus providing a non-repudiable record of her decision.

Alice attempts to log into a secure console in the control room. The secure console recognizes Alice but a second factor is required. The console issues a challenge to Alice at her registered account asking if she would like to log into the secure console:

```
Console> message confirm alice@example.com start
```

Alice checks her pending messages and accepts the request:

```
Alice> message accept NBAC-LLBY-E4EU-7ZRF-I47O-ZYHZ-PBCW
ERROR - The specified message could not be found.
```

The secure console verifies the response and grants access:

```
Alice> $message status {confirmResponseID}
ERROR - The command System.Object[] is not known.
```

In an enterprise environment, tying the confirmation process to a specific source, a specific action and specific device allows for confirmation interactions to be used to implement business processes with attribution and thus accountability.

Using traditional second factor approaches, a system administrator presents their credentials to authenticate access to the machine at which point they can perform any action permitted by their current privileges. This typically includes modification of any access logs that might be kept. Using the confirmation approach the individual actions of the system administrator may be authenticated, traced and logged. If a user account is added to the system, it is known which administrator is responsible and the device that was used. This information may then be used if it becomes necessary to unwind the consequences of a breach or an insider threat.

4.4. Encryption Groups

As seen earlier, the Mesh allows encrypted files to be shared with other named users. While this capability is sufficient for simple messaging type use cases, decades of experience prove that it is inadequate to meet the needs of protecting data at rest. In the simple messaging case the list of recipients is known to the sender at the time a message is sent. In the general case the party encrypting the data cannot know the list of intended readers because that will change over time.

Even in the smallest organization, employees join and leave. A new employee must be granted access to all the information they need for their work. The access rights of a terminated employee must also terminate.

Traditional 'Digital Rights Management' product employ key management techniques originating in the field of copyright enforcement to control access to content by controlling disclosure of symmetric decryption keys. This provides the necessary flexibility to control access to the data but leaves the decryption keys vulnerable to a server breach. Such systems do not provide 'end-to-end' security in any useful sense.

Use of threshold techniques allows a threshold service to control decryption of the data without having the ability to decrypt. Sharing data through a Mesh group allows access to be controlled without loss of end-to-end encryption.

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

```
Alice> group create groupw@example.com
ERROR - Cannot access a closed file.
```

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

```
Alice> dare encode grouptext.txt /encrypt groupw@example.com /out ^
    groupsecret.dare
ERROR - The option System.Object[] is not known.
```

Even though she is the group administrator, Alice can't decrypt the file either until she adds herself to the group.

```
Alice> dare decode groupsecret.dare
ERROR - No decryption key is available
```

Alice adds Bob to the group:

Missing example 2

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:

Missing example 3

Removing Bob from the group immediately withdraws his access.

Missing example 4

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

Missing example 5

4.5. Escrow and Recovery

While disclosure of sensitive data might cause serious harm to its owner it is very rarely the case that the consequences of disclosure are greater than the consequences of loss. Thus, whenever static data is to be encrypted, the question of key recovery must be considered.

Alice decides to create a recovery key set. To do this, she specifies the number of key shares to be created and the number required for recovery:

```
Alice> account escrow
Share: SAQN-H7KA-DN7Z-SWGM-NTOL-EBGN-TGJN-UNEA-H4LH-VTYK-P66S-KE3D-JI
FE-NWOY-JRGQ
Share: SAQ5-76L5-SN7F-BMIK-AQRW-DU7F-FDJY-P3C6-DQZE-LOLJ-IGRQ-O2IY-6E
G2-UQ3Z-BCUQ
Share: SARO-X5N3-BN6Q-QCKH-TNVB-DIX4-XAKD-LJB3-7FHB-BI6I-AOEO-TPWO-TA
IQ-3LIZ-YUCQ
```

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

```
Alice2> account recover /verify
ERROR - Expected {
```

4.6. Future Applications

The Mesh is a Threshold Key Infrastructure and as with any infrastructure, it is designed as a platform to support as wide a range of future developments as possible. As shown previously, the Mesh Messaging system provides an improved superset of the functions of SMTP. It is also capable of being extended to support every current communication modality with true end-to-end protection of data confidentiality.

4.6.1. Synchronous Messaging

Addition of a presence service capability to the MSP would allow Mesh Messaging to be used to support the full range of synchronous messaging services from text chat (e.g. xmpp) to video and VOIP.

The main technical issue to be addressed to enable such a service is specifying a means of layering Mesh Messages direct over UDP transport. This is currently at the concept phase. While the precise means of layering audio and video formats onto a network connection is a complex problem, it is one that has already been solved by existing standards.

4.6.2. Social Media

One of the chief distinctions between messaging and 'social media' and is that the former is typically used to describe a synchronous interaction between a closed group of users while most social media

consists of asynchronous interactions which are frequently (but not always) public.

The Data At Rest Envelope technology used in the Mesh was originally designed to support asynchronous social media interactions with full end-to-end confidentiality. The service hosting a forum or discussion board need not have access to the content of the messages to support the complete range of user interactions.

5. 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. Since threshold cryptography is not currently part of the 'canon' from which designers of cryptographic security protocols work, much of the cryptography used in the Mesh has been designed for the Mesh. Despite this fact, it is properly regarded as part of the Internet platform on which the Mesh is built rather than a part of the Mesh itself.

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.

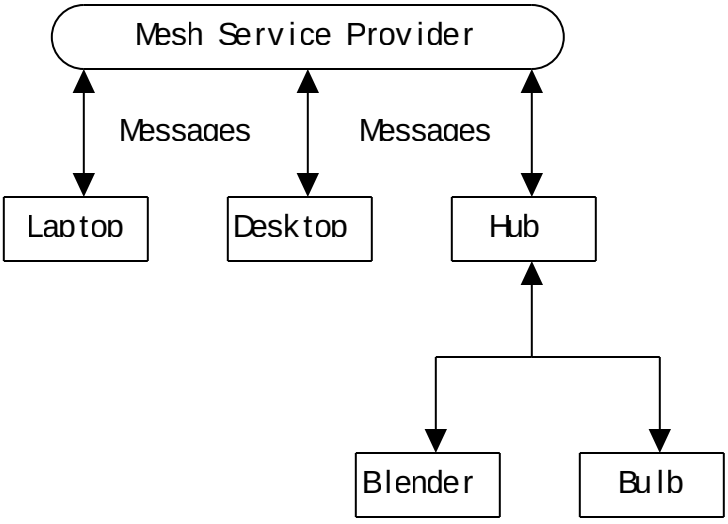

Figure 2

5.1. 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.

5.2. 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.

5.3. Threshold 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 $z = (x+y) \bmod g$ (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:

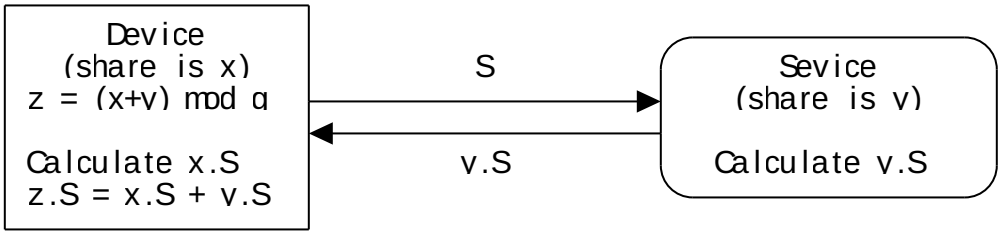

Figure 3

The approach to threshold 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 which is in turn one form of threshold cryptography.

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.

These techniques are described in detail in [draft-hallambaker-threshold].

5.4. Threshold Key Generation

The mathematics that support threshold 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*.

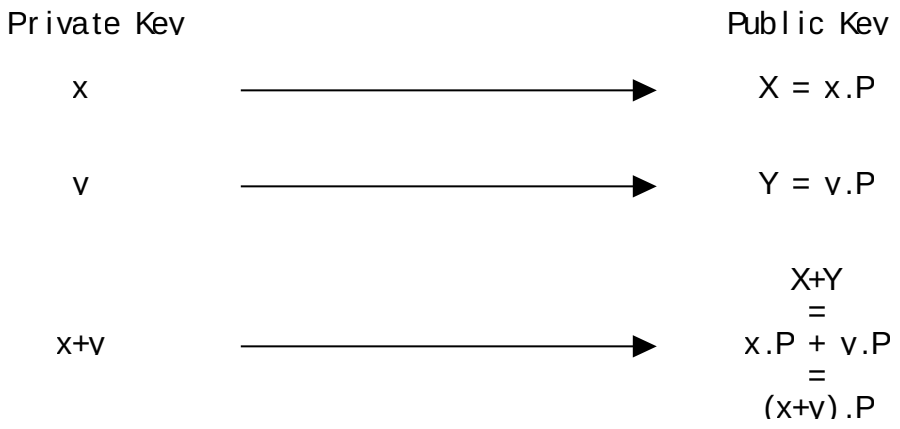

Figure 4

Threshold 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.

These techniques are also described in detail in [draft-hallambaker-threshold].

5.5. Threshold Signature

The techniques that support threshold decryption and key generation are also applicable to signature albeit with some very important constraints. Incorrect implementation of the techniques used to create ECDSA signatures can result in disclosure of the private key. It is therefore essential that a threshold signature algorithm is rigorously reviewed.

This technique is used in the mesh to partition the use of administration keys so that the consequences of losing an administrative device can be mitigated.

These techniques are described in detail in [draft-hallambaker-threshold-sigs].

5.6. 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.

5.6.1. 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.

5.6.2. 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 each time it starts to establish a new shared secret for that session. The shared secret is then used to create fresh symmetric keying material for each entry in the log using a unique nonce per entry.

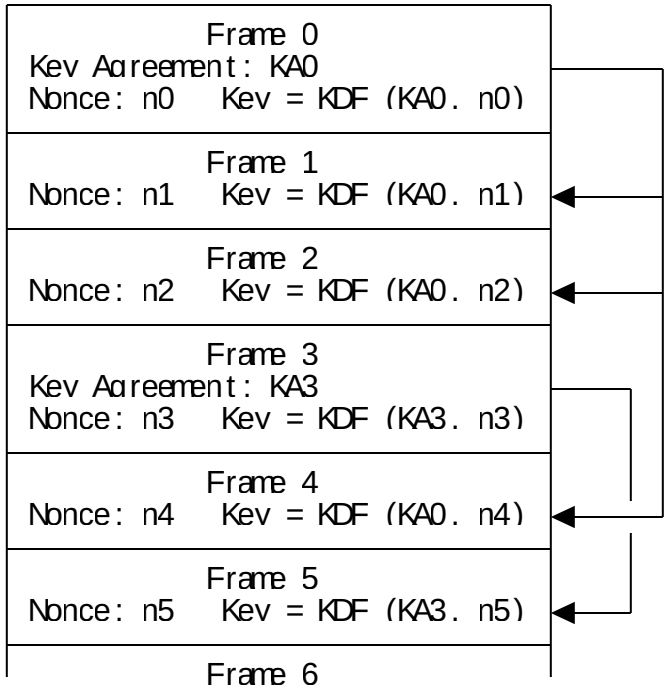

Figure 5

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.

5.7. 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:

```
NAB6-GXIR-GXA5-AQJO-4CNH-LWOL-EOVQ
7UHC-4YPF-LDY4-2F65-7DYN-UUPC-ZY
SAQF-HCGQ-JTQ5-5YGF-PT4S-QPXL-KOPE-U
MB5S-R4AJ-3FBT-7NHO-T26Z-2E6Y-WFH4
KCM5-7VB6-IJXJ-WKHX-NZQF-OKGZ-EWVN
AABD-4PN3-QGA2-IJU5-RP7H-Z5AV-I3GK
```

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.

5.7.1. 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.

5.7.2. 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:

```
mcu://alice@example.com/ADJ5-4NLV-O7YC-SKR6-EI
```

The EARL may be expressed as a QR code:

Figure 6

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.

5.7.3. 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.

5.8. 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/Lagrange secret sharing and recovery 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.

To facilitate escrow and recovery, all the public key pairs and key shares associated with a Mesh profile are generated from a seed value using a deterministic algorithm. Thus, escrow of the seed value is sufficient to permit recovery of the private key data.

For example, Alice escrows her Mesh Profile creating three recovery shares, two of which are required to recover the master secret:

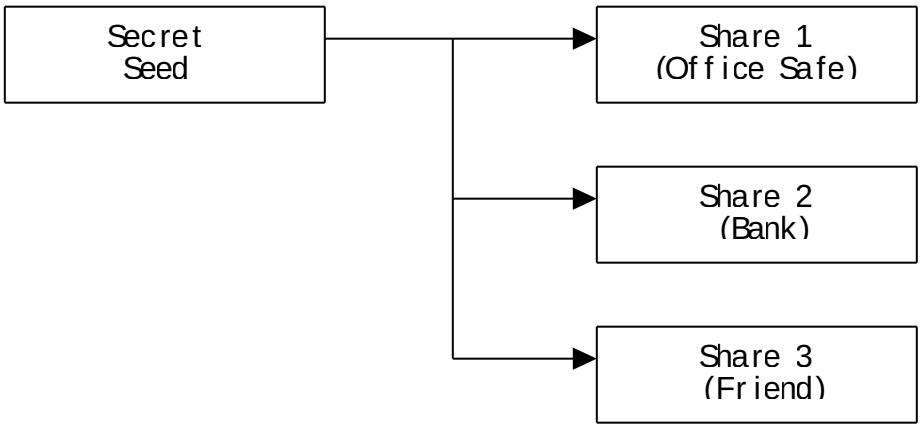

Figure 7

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:

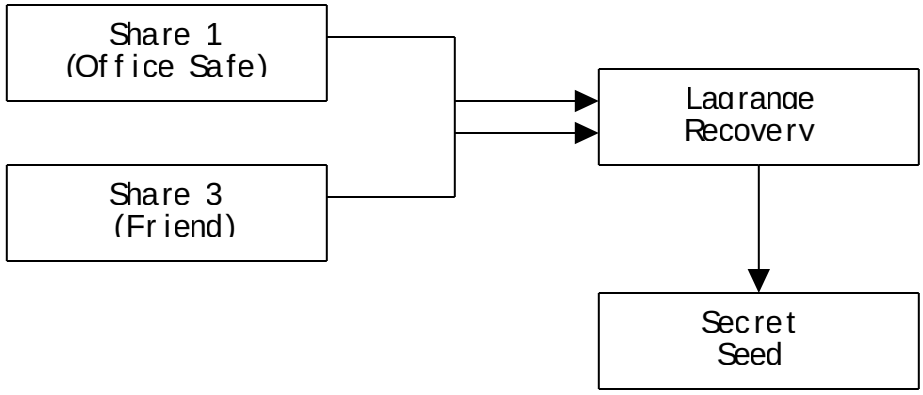

Figure 8

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.

6. Mesh Architecture

The Mesh infrastructure is supported by a compact set of structures and protocols. These are discussed in detail in the Schema Reference [draft-hallambaker-mesh-schema] and Protocol Reference [draft-hallambaker-mesh-protocol] documents.

The JSON object model and JSON or JSON-B serialization [draft-hallambaker-jsonbcd] are used for all Mesh data structures. These include:

Assertions A DARE envelope containing a signed data object. Assertions include Profiles and Connections.

Profile A self-signed assertion describing a Mesh principal. Principals include Accounts, Devices and Services.

Every profile specifies a profile signature key under which it is signed. The UDF fingerprint of the profile signature key is used as a unique identifier for the profile. Thus, all attributes declared in the profile such as authentication keys and encryption keys are bound to the unique identifier for the profile.

Connection An assertion signed by one principal delegating rights to another. Connection assertions are used to bind devices to accounts and hosts to a service.

Activation A DARE envelope encrypted under the encryption key of a principal that grant rights or capabilities to that principal.

This is typically but not always achieved through use of threshold key techniques.

Message A DARE envelope signed by the sender that is encrypted under the encryption key of the intended recipient(s) whose content is a Mesh messaging object.

Entry An object stored in a catalog that carries an identifier that is unique for that catalog.

6.1. Actors

Three Mesh actors are defined: Accounts, Devices and Services. Each of these is described by a specific type of profile.

6.1.1. Account

Two types of Mesh account are currently specified: personal accounts and group accounts. For concise exposition, this document is limited to the description of personal accounts. Group accounts are specified in the Schema Reference [draft-hallambaker-mesh-schema].

A Mesh account is an abstraction which may be loosely regarded as the thing to which a collection of devices (in the case of a user account) or a collection of members (in the case of a group account) belong.

Each personal account profile specifies:

Profile Signature Key Used to authenticate the profile. Updates to the profile require use of the Profile Signature Key. The Profile

Signature Key cannot be changed but a profile may be replaced by a new profile.

Uniform Data Fingerprint The UDF fingerprint of the Profile Signature Key. This is used as a unique identifier for the account.

Account Name The account name through which the profile is serviced.

Administration Keys UDF fingerprint of keys that are authorized to sign device connection assertions and update the Device Catalog.

Signature Key Public parameters of the account signature key. This is the key that counterparties will use to verify messages sent by the account holder.

Encryption Key Public parameters of the account encryption key. This is the key that counterparties will use to encrypt messages sent to the account holder.

Authentication Key Public parameters of the account authentication key. This is the key that counterparties will use to establish authenticated exchanges with the account holder.

The public keys for encryption, authentication and signature specified in the account profile are the only keys that will (in normal circumstances) be visible to other Mesh accounts.

6.1.2. Device

A Mesh Device is any device that is connected to a Mesh Account through a Device profile. A given physical device may have multiple device profiles associated with it but for the purposes of the Mesh, these are considered to be separate devices. A given device profile may be connected to more than one account

The device profile specifies:

Profile Signature Key Used to authenticate the profile. Updates to the profile require use of the Profile Signature Key. The Profile Signature Key cannot be changed but a profile may be replaced by a new profile.

Uniform Data Fingerprint The UDF fingerprint of the Profile Signature Key. This is used as a unique identifier for the device.

Signature Key Public parameters of the device signature key share. This key share is used as a contribution to the signature key the

device will use in the context of the account and to authenticate device connection requests.

Encryption Key Public parameters of the account encryption key share. This key share is used as a contribution to the encryption key the device will use in the context of the account and to decrypt activation records sent in response to device connection requests.

Authentication Key Public parameters of the account authentication key share. This key share is used as a contribution to the authentication key the device will use in the context of the account.

Description Optional information describing the device provided by the manufacturer. E.g. model, serial number, date of manufacture etc.

A Mesh Device is connected to an account through the creation of an activation record and a connection record.

The activation record contains key shares that are overlaid on the corresponding shares specified in the device profile to create the set of encryption, authentication and signature keys the device will use in the context of the account. Since the private keys corresponding to the device profile keys are only used to enable the connection of the device to an account, these keys are only trusted to a minimal degree.

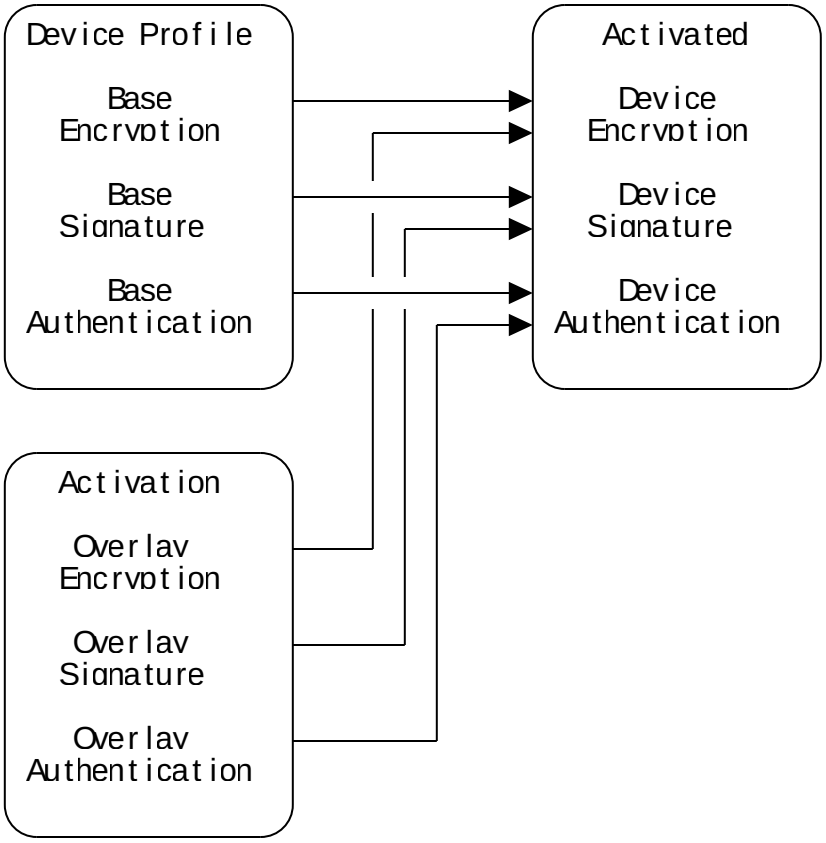

Figure 9

In the ideal case, the device profile keys are fixed to the device such that they may be used to perform private key operations without the ability to extract the private key data from the device. Since the device profile is only trusted for the limited purpose of connecting the device to an account, the device profile may be created during manufacture without undue concern for either disclosure of the private key on the part of the account holder or a reputation attack alleging disclosure of the private key on the part of the manufacturer.

The device connection record is functionally a certificate that the device may use to interact with the Mesh Service or to other devices connected to the same account. Note however that use of threshold cryptography means that Mesh devices would not normally present their device connection record to any other party since all communication with external parties takes place through the keys published in the account profile.

6.1.3. Service

A Mesh Service is an abstract network service that is provided by one or more hosts. The properties of the service are described by the service profile.

The service profile specifies:

Profile Signature Key Used to authenticate the profile. Updates to the profile require use of the Profile Signature Key. The Profile Signature Key cannot be changed but a profile may be replaced by a new profile.

Uniform Data Fingerprint The UDF fingerprint of the Profile Signature Key. This is used as a unique identifier for the device.

Signature Key Public parameters of the service signature key. This is the key that counterparties will use to verify messages sent by the service.

Encryption Key Public parameters of the service encryption key. This is the key that counterparties will use to encrypt messages sent to the service.

Authentication Key Public parameters of the account authentication key. This is the key that counterparties will use to establish authenticated exchanges with the service.

Hosts are Mesh Devices that have been granted a Host Activation and Host Connection by a service administrator. These are used in the same fashion as the device activation and connection records.

6.2. Stores

Mesh Stores are append-only sequences that are used to represent collections of objects, messages and data. All Mesh stores are implemented as DARE Sequences authenticated by means of a Merkle tree. The payload of each envelope in the sequence is usually encrypted.

Three types of Mesh store are currently defined:

Catalog A set of Mesh objects, each of which has an identifier that is unique in the scope of the catalog. Objects may be added, updated, and deleted.

Spool A sequence of Mesh Messages.

All the state represented within a Mesh account is contained in Mesh stores bound to the account. Thus, to synchronize a device to the state of the account, it is sufficient to synchronize the collection of stores the device is permitted to read. Since every store is an append-only sequence, it is sufficient for the Mesh service to return the envelopes added to each of the stores since the device was last synchronized.

Rapid synchronization of catalogs and spools is ensured by limiting the size of entries to each. Implementations may further improve performance by redacting stores to remove obsolete entries that have been updated or deleted. Alternatively, a device may maintain a complete record of the state of the store to allow erroneous changes to the store to be unwound.

6.2.1. Catalogs

Mesh Catalogs track a collection of entries. Every Mesh account contains a Threshold Catalog that is used by Mesh services as the source of access control policy. The Threshold Catalog is unique in that it is the only catalog whose contents can be read by the Mesh Service. Every other Mesh Catalog connected to a Mesh account is end-to-end encrypted so that it can only be read by devices connected to the account.

The Mesh specifies various catalogs that are used to track information relevant to a Mesh Account:

Device The devices connected to the corresponding Mesh profile.

Contact Logical and physical contact information for people and organizations.

Bookmark Web bookmarks and citations.

Credential Username and password information for network resources.

Calendar Appointments and tasks.

Network Network access configuration information allowing access to wireless networks and VPNs.

Application Configuration information for applications including mail (SMTP, IMAP, OpenPGP, S/MIME, etc) and SSH.

Each catalog connected to an account has a unique identifier of the form mmm_<<name>. Applications may specify additional catalogs without risk of collision with future Mesh catalogs by using an appropriate IANA assigned protocol label.

6.2.2. Spools

Spools are used to track inbound and outbound messages. Three spools are currently defined:

Inbound Messages that have been received by the service and accepted for delivery to the account.

Outbound Messages that have been sent from the account through the service.

Local A spool used to exchange messages with devices connecting to the device.

6.3. Mesh Service Protocol

Mesh services communicate with Mesh devices and other Mesh Services through the Mesh Service Protocol. Despite the wide range of Mesh functionality, the Mesh protocol is remarkably compact. The bulk of the semantics associated with the Mesh are expressed in the schemas describing Mesh Messages and Catalogs. The objective of reducing the degree of trust in the Mesh service to the absolute minimum by necessity requires that the Mesh Service be extremely simple.

Mesh Service Protocol transactions are divided into the following groups:

Service Description The Hello transaction returns a description of the service including information used to authenticate future interactions with the service.

Account Management The Create and Delete transactions are used to bind an account to a service.

Device Connection The Connect and Complete transactions are used to connect devices to an account

Synchronization The Status, Download and Transact transactions are used to update stores connected to an account.

Messaging The Post transaction is used by one Mesh Service to transfer a message from one of its users to a different Mesh Service serving one of the recipients.

Publication The Publish, Claim and PollClaim transactions are used to publish and retrieve data objects through an account.

Cryptographic The Operate transaction requests that the service performs a cryptographic operation on behalf of the account. This

is used to provide execution of threshold operations on behalf of the account holder for both internal and external users.

Future versions of the Mesh Service Protocol may support additional transactions to support features such as providing DNS resolution.

6.3.1. Protocol Interactions

Every Mesh Service Protocol transaction consists of a single request from a Mesh client followed by a single response. Requests and responses are authenticated and encrypted under a key established between the client and the service. This application layer enhancement is in addition to any transport layer enhancement that may be employed (e.g. TLS).

Mesh Service Protocol messages may be exchanged through any binding advertised by the service by means of the Hello transaction. Currently only one binding is defined, mapping Mesh requests and responses to the content data of HTTP POST requests and responses layers over a TLS transport.

While the use of up to three layers of encryption may be regarded as excessive, each layer provides separate protections:

Transport Layer Provides confidentiality for metadata and limited traffic analysis protections.

Application Layer Encryption and authentication of requests and responses using keys bound to the specific device and service performing the interaction provides the basis for access control.

Data Layer Encryption of stored data (catalog data, device activations, etc.) provides end to end security between the devices connected to the account.

6.4. The Threshold Catalog

The Threshold Catalog of a Mesh account provides the basis through which the service implements the access control policy specified by the account.

Each entry in the catalog specifies an operation that the service will perform when it receives a request that is authenticated and authorized by the access control policy specified in the entry. Operations include:

Inbound Message Filtering Messages received by the service that match the specified criteria will be appended to the inbound message spool.

Threshold Key Generation
: Performs a threshold key splitting operation of a private key held by the service and encrypts one part under a key known only to the service and encrypts the other under a public key specified by the party making the request.

Key Agreement
: Performs a key agreement operation on a private key held by the service. This may be used as a component in a threshold key agreement scheme.

Signature
: Performs a signature operation on a private key held by the service. This may be used as a component in a threshold signature scheme.

These operations provide the vocabulary from which a Threshold Key Infrastructure is built. Keys that are bound to a service using threshold techniques can only be applied with the co-operation of that service.

6.5. Mesh Messaging Protocol

Mesh devices connected to an account interact with the Mesh Service through the Mesh Service protocol. Mesh devices interact with other Mesh devices through the Mesh Messaging Protocols, each of which provides a distinct application functionality:

* Connection Protocol
* Confirmation Protocol
* Contact Exchange Protocol

Each of these protocols is described in depth in the Mesh Protocol Reference [draft-hallambaker-mesh-protocol].

Mesh Messages provide a means of communication between Mesh Service Accounts with capabilities that are not possible or poorly supported in traditional SMTP mail messaging:

* End-to-end confidentiality and authentication by default.
* Abuse mitigation by applying access control to every inbound and outbound message.
* End-to-end secure group messaging.
* Transfer of exceptionally large data sets (Terabytes).

Note that although Mesh Messaging is designed to facilitate the transfer of very large data sets, the size of Mesh Messages

themselves is severely restricted. The current default maximum size being 64 KB. This approach allows Mesh

In addition, the platform anticipates but does not currently support additional cryptographic security capabilities:

*Traffic analysis resistance using mix networks (Chaum).

*Simultaneous contract binding using fair contract signing (Micali).

While these capabilities might in time cause Mesh Messaging to replace SMTP, this is not a near term goal. The short-term goal of Mesh Messaging is to support the Contact Exchange and Confirmation applications.

Two important classes of application that are not currently supported directly are payments and presence. While prototypes of these applications have been considered, it is not clear if these are best implemented as special cases of the Confirmation and Contact Exchange applications or as separate applications in their own right.

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.

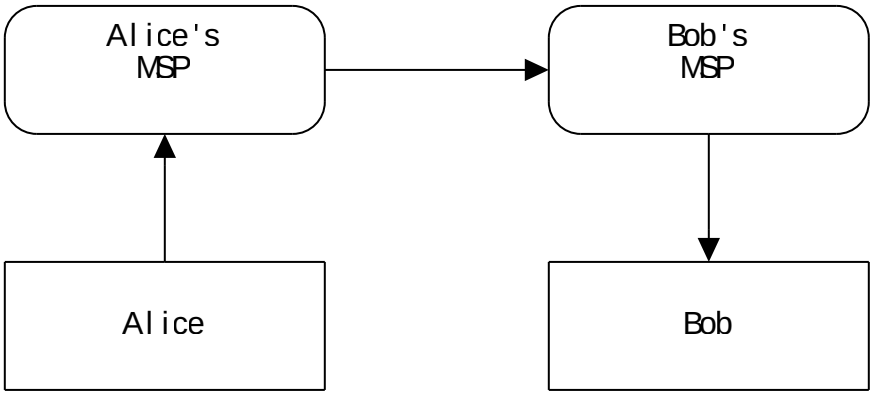

Figure 10

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.

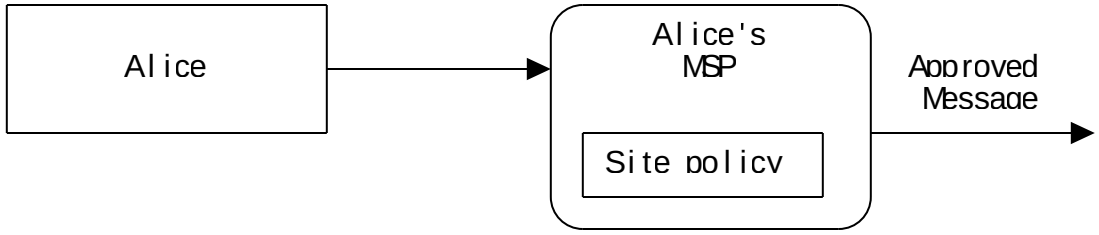

Figure 11

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.

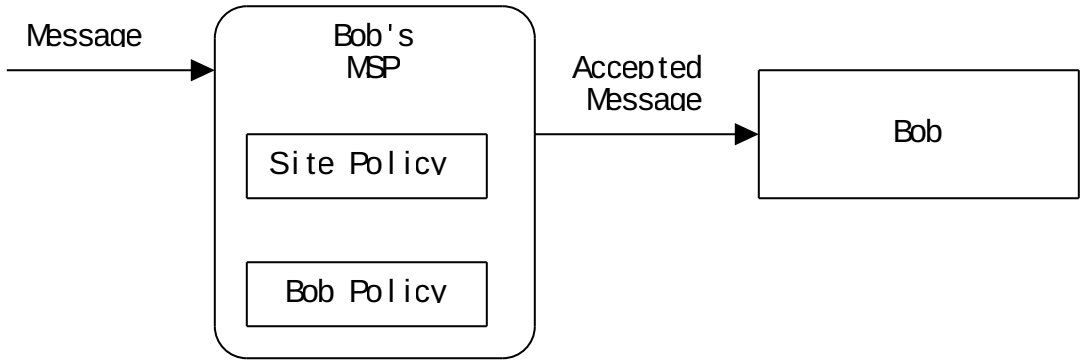

Figure 12

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 exceptionally 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.

6.6. 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:

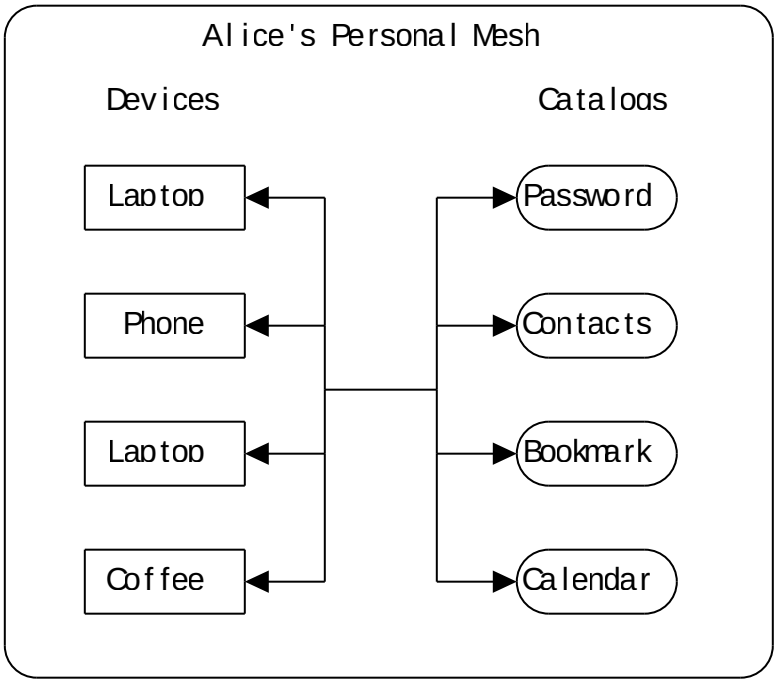

Figure 13

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

6.6.1. 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.

7. 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].

8. IANA Considerations

This document does not contain actions for IANA

9. Acknowledgements

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

10. 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-18, 23 October 2020, <https://tools.ietf.org/html/draft-hallambaker-jsonbcd-18>.

[draft-hallambaker-mesh-cryptography]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VIII: Cryptographic Algorithms", Work in Progress, Internet-Draft, draft-hallambaker-mesh-cryptography-06, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-cryptography-06>.

[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-08, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-dare-08>.

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-hallambaker-mesh-developer-10, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-developer-10>.

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

[draft-hallambaker-mesh-platform]
Hallam-Baker, P., "Mathematical Mesh: Platform Configuration", Work in Progress, Internet-Draft, draft-hallambaker-mesh-platform-06, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-platform-06>.

[draft-hallambaker-mesh-protocol]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part V: Protocol Reference", Work in Progress, Internet-Draft, draft-hallambaker-mesh-protocol-06, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-protocol-06>.

[draft-hallambaker-mesh-schema]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part IV: Schema Reference", Work in Progress, Internet-Draft, draft-hallambaker-mesh-schema-05, 16 January 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-schema-05>.

[draft-hallambaker-mesh-security]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VII: Security Considerations", Work in Progress, Internet-Draft, draft-hallambaker-mesh-security-05, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-security-05>.

[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-10, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-udf-10>.

[draft-hallambaker-threshold]
Hallam-Baker, P., "Threshold Modes in Elliptic Curves", Work in Progress, Internet-Draft, draft-hallambaker-threshold-03, 3 September 2020, <https://tools.ietf.org/html/draft-hallambaker-threshold-03>.

[draft-hallambaker-threshold-sigs]
Hallam-Baker, P., "Threshold Signatures in Elliptic Curves", Work in Progress, Internet-Draft, draft-hallambaker-threshold-sigs-04, 3 September 2020, <https://tools.ietf.org/html/draft-hallambaker-threshold-sigs-04>.

[draft-hallambaker-web-service-discovery]
Hallam-Baker, P., "DNS Web Service Discovery", Work in Progress, Internet-Draft, draft-hallambaker-web-service-discovery-04, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-web-service-discovery-04>.

[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>.

11. Informative References

[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-06, 27 July 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-trust-06>.