Network Working Group P. Hallam-Baker
Internet-Draft Comodo Group Inc.
Intended status: Informational September 18, 2017
Expires: March 22, 2018

# Mathematical Mesh: Architecture
## draft-hallambaker-mesh-architecture-04

## Abstract

The Mathematical Mesh ?The Mesh? is an end-to-end secure infrastructure that facilitates the exchange of configuration and credential data between multiple user devices. The architecture of the Mesh and examples of typical applications are described.

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

## Status of This Memo

This Internet-Draft will expire on March 22, 2018.

## Copyright Notice

Table of Contents

## 1. Introduction

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

The Mesh uses cryptography and an untrusted cloud service to make management of computer configuration data transparent to the end user. Each Mesh user has a personal profile that is unique to them. A user may link devices and applications to their Mesh profile to enable transparent sharing of data between them.

For example, Alice has a laptop computer and a tablet. They are both linked to her Mesh profile which allows either to be used for email or to control any devices in her smart home. Alice has chosen to only make her cloud documents available on her laptop but she could change that to add her tablet should the need arise (Figure XX).

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

Alice's Mesh profile connections.

The Mesh allows devices connected to a profile to be provisioned with all the network configuration settings and credentials to enable the device to be used with the user's applications. In most cases, public key credentials will be provisioned to enable transport layer and end-to-end encryption.

All Mesh profiles are authenticated using digital signatures and all private material protected using industry standard end-to-end encryption.

Mesh profiles are typically published to a Mesh portal, an untrusted cloud service that provides mailbox-like capabilities to enable a seamless user experience for users who do not necessarily have an 'always on' machine to act as a broker for profile management operations. Sophisticated users MAY operate a personal Mesh portal should they choose.

Mesh portals MAY in turn be members of a federation exchanging updates, thus providing users with a guarantee of continued service should the portal they selected become unavailable. Each portal belonging to a federation maintains a local linked hash notary log to which all transactions are recorded. The outputs from each local portal log are in turn periodically (e.g. once an hour) enrolled in a meta-log maintained by the federation as a group. Thus ensuring that

no Portal belonging to a federation can defect un-noticed unless the entire federation defects.

## 1.1. Document Roadmap

The specifications describing the Mesh protocols are divided into three main groups. First set of documents describe the architecture, data structures and protocols that make up the Mesh core. The second set describes the use of the Mesh to secure existing and experimental documents. The third set of documents describe building blocks that were developed to meet requirements arising from the Mesh but are not specific to the Mesh and could be applied to the development of other Web Services that do not involve the Mesh.

Mesh Core  This document provides a high level description of the Mesh architecture and mode of use. Detailed specifications including schemas and examples are specified in the accompanying Mesh Reference document [draft-hallambaker-mesh-reference] .

Mesh Applications  The use of the Mesh to configure mail and SSH clients, and to store catalogs containing passwords, contacts and bookmarks is described in [draft-hallambaker-mesh-app] .

The Mesh is also used as the platform for building two new applications, Mesh/Recrypt [draft-hallambaker-mesh-recrypt] , and Mesh/Confirm [draft-hallambaker-mesh-confirm] .

Mesh Platform  JSON Web Service Binding [draft-hallambaker-json-web-service] , Uniform Data Fingerprint [draft-hallambaker-udf] , Strong Internet Names [draft-hallambaker-sin] , JSON-BCD [draft-hallambaker-jsonbcd]

In addition, two additional documents describe the use of the reference code base [draft-hallambaker-mesh-developer] and recommendations for implementing Mesh enabled applications to take advantage of the cryptographic facilities offered by specific operating system platforms [draft-hallambaker-mesh-platform] .

## 2. Definitions

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

## 2.1. Related Specifications

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

Cryptographic Algorithms Mesh applications use the cryptographic algorithm suites specified by the application. The cryptographic algorithms used in the Mesh itself are limited to SHA-2 [SHA-2] and SHA-3 [SHA-3] digest functions, AES Encryption [FIPS197] and RSA Signature, and Encryption [RFC8017] .

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

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

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

## 2.2. Defined Terms

TBS

## 2.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] .

## 2.4. Implementation Status

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

## 3. Requirements

Before the Web, most trade was performed in person. Mail order existed but was limited in scope. Most banking transactions other than withdrawing cash from an ATM had to be performed in-line at a branch rather than online through the Web. Trade in stocks and shares barely existed in its modern form. The TLS protocol and the WebPKI that supported it enabled the e-commerce economy that we live in today.

The WebPKI is a powerful infrastructure but it does have one major drawback: It Authenticates the bank to the customer but it does not authenticate the customer to the bank. The use of passwords instead of strong cryptographic credentials makes users vulnerable to phishing.

Design of a public key authentication protocol is straightforward. Making the use of such a protocol practical is a much greater challenge because it requires the user to manage a private key. Users cannot perform even the simplest public key cryptography algorithm in their head and so some sort of device must perform the calculations and this in turn must be provisioned with the private key.

Management of the server private keys for the WebPKI is challenging enough and they tend to stay in one place (at least until recently). Managing private keys for users is much more challenging than managing server keys because:

- Users typically own multiple devices and expect them all to work in the same way.
- User devices are considerably more complex than servers. They have many functions.
- Users are more likely to lose devices or lend them to other people.
- Users cannot be expected to be experts.

What has made matters worse is the notion that because users are less sophisticated than system administrators, they cannot use sophisticated technology. In practice, the exact opposite is the case. To make the user experience completely frictionless, we must embrace technologies that are more sophisticated, not avoid them. Systems administrators are usually willing to accept technology that is less than perfect and shows many 'rough edges' because they are

experts and using those tools is their job. Users are much less tolerant of technology that meets their needs.

Modern cryptography provides us with the tools to secure practically any form of Internet interaction. In almost every case, the main constraint that holds us back is the impracticality of using client-side private keys:

- OpenPGP [RFC4880]
- S/MIME [RFC5751]
- Jabber [RFC6120]
- IPSEC [RFC4301]
- SSH [RFC4251]

The SSH protocol supports the use of public key cryptography for authentication, of course. But this is the exception that proves the rule. The use of client side keys in SSH is highly effective for the developer or the systems administrator who only makes use of one machine as their client. Attempting to manage SSH credentials across multiple client and server machines under strict operational controls is not for the fait hearted. Not only is it not uncommon for users to use a single private key for SSH on all the machines they use, there are Web sites that 'explain' how to do this by emailing their private key to themselves over SMTP.

From the user's perspective, a security protocol 'works' when it lets them do their job in peace. The VPN access token works when they gain access to the corporate network, SSH works when they gain access to the server, passwords work when they can log into their bank Web site and pay their bills. But when evaluating a security protocol, it is equally important that the attacker is defeated and that no new failure modes are introduced. The acronym C.I.A. provides a useful mnemonic:

Confidentiality  Protect data from disclosure to unauthorized parties. Prevent unauthorized parties inferring confidential information from traffic analysis.

Integrity  Protect data from unauthorized modification. Establish and verify data authenticity.

Availability  Ensure that data and data services are available when needed. Restrict access to authorized parties. Establish accountability.

While Confidentiality is usually the paramount concern of users, Integrity attacks almost invariably inflict more serious damage and Availability attacks include some that inflict the greatest damage of all.  The typical consumer gets irritated if their bank carelessly reveals details of their account but is considerably angrier if it has been depleted by fraud.  But as the recent spate of ransomware attacks prove, while the consumer becomes angry when they are a victim of fraud, many will actually pay money to the criminals if the alternative is to lose the pictures of their grandchildren when they were five.

Best practices for managing cryptographic keys present multiple operational challenges:

Separation of Cryptographic Purpose  Each private key SHOULD be used for exactly one cryptographic function.  Keys used for encryption SHOULD NOT be used for authentication.  Keys used to secure one application SHOULD NOT be used to secure another.

Key Compromise  Revocation of compromised keys MUST be supported.

Key Rotation  It SHOULD be possible to periodically refresh keys used to secure applications, thus ensuring that any compromise of the keying material is time bounded.

Device Isolation  Each device SHOULD have separate keys to protect applications on that device.  This limits the consequences should the device be compromised, lost or stolen and permits revocation to be limited to the keys used in that device.

Key Escrow  Whenever a key is used to encrypt static data (i.e. data stored on a disk or other storage device), provision MUST be made to permit (but not require) the decryption key to be escrowed to permit recovery should the need arise.

Provision of Key Escrow is controversial since any mechanism that enables the user to recover a cryptographic key voluntarily enables the user to disclose the key in case of coercion.  But this state of affairs, while unsatisfactory, is a lot more satisfactory than the current state of affairs which is to rarely encrypt static data at all.

## 3.1.  Use Case

Alice works with Bob, Carol, and Doug.  As part of her work she exchanges email messages with them which may contain confidential information.  She also connects to the machines Server1, Server2 and

Server3 to perform system administration tasks. She has a personal desktop, a laptop, and a tablet computer. She requires:

- The ability to send and receive S/MIME end-to-end encrypted email with Bob and Carol as per corporate policy.

- The ability to send and receive OpenPGP end-to-end encrypted email with Doug who does not use S/MIME.

- The ability to connect to Server1, Server2 and Server3 from any of her personal devices.

- The ability to quickly configure a new device for her personal use.

- The ability to quickly disable further use of a device should it be stolen.

This use case is deliberately limited to configuring Alice's devices to enable her to use current security protocols. A large number of use cases and applications were considered during the design including configuring IoT devices in Alice's home and configuring a borrowed or rented device. It was discovered that considering the end-to-end email case was sufficient because the requirements it exposes are a superset of the requirements of the others.

The only use case that introduced requirements beyond Alice configuring end-to-end email and SSH for herself was configuring end-to-end email and other secure applications for a remote co-worker. Meeting these requirements is deferred as a future work item.

### 3.2. What Makes PKI Hard

Every day, billions of people access Web sites using PKI encryption without even being aware that they are doing so. A smaller but still large number of people use PKI for secure payments, the only difference they are aware of being that they insert their card rather than swiping it through the reader.

Many of the issues that have made PKI hard in the past are due to design choices taken by technologists rather than an intrinsic restriction of PKI. In particular:

- Expiry of private keys and credentials.

- Poorly designed enrollment protocols.

- No provision for use of multiple devices.

- Inappropriate trust models.

- Intrusive user experience.

S/MIME and other client centered security technologies were added to many Internet applications in the 1990s in response to enterprise requirements.  The people who make the purchasing decisions for enterprise software and those who use it on a daily basis are often if not invariably different, the enterprise software market has prioritized functionality over usability.  An email client that requires the user to remember to click on the right button to encrypt an email is considered acceptable in the enterprise space, it is not acceptable in the consumer space.

While working on this project, the author attempted to configure a very popular email client to make use of the built in S/MIME capabilities.  Even with 25 years of experience, this took over half an hour and required the user to follow a procedure with 17 different steps involving three different applications.  Even though the certificate was being issued for use with email, the user had to use a Web browser to enroll for the certificate, validate the request using the email client, download the certificate using the Web browser and then install it using a key management tool.

The bar for security usability is much higher than most security specialists, even those who focus on security admit.  Experience should teach us that the iron law of security usability is that a security application that requires the user to think about security will fail.

It is noted in passing that security usability is not achieved by preventing the user seeing the information they need to make their own security decisions.

### 3.3.  The Devil is in the Deployment

One of the most important reasons for the failure of PKI applications has been the failure of PKI applications.  As with any communication tool, the value of end-to-end secure email is a function of the size of the network that can be reached.  The community of S/MIME users and the community of OpenPGP users have both stalled in the low millions, a significant number but falling far short of ubiquity. End to end secure email can only realize its full potential when its use is the norm and not the vanishingly rare exception.

After a time, failure becomes a self-reinforcing vicious circle. Very few people use end-to-end secure email applications because they are difficult to use.  Application providers refuse to invest in

developing end-to-end secure email applications because 'there is no demand'.

The Mesh is designed for deployment by providing a stand-alone value proposition to early adopters. The ability to automate the use of end-to-end secure email is not a highly attractive proposition for most when less than 0.1% of the Internet user population have ever registered an S/MIME or OpenPGP key. But an end-to-end secure password manager for Web browsers, or an SSH credential management tool do provide a stand-alone value proposition.

### 3.4. The Untrusted Cloud

The Mesh supports multiple mechanisms for connecting devices to an account. Each of these mechanisms provides for strong mutual authentication of the device profile to the personal profile it is being connected to and vice versa. In each case, the connection request MUST be authorized by the user from a device that has already been connected to the profile and granted administration privileges.

Figure XX shows Alice connecting a new desktop computer to her profile, the connection request is initiated from the new device (desktop) and approved from the administration device (tablet).

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html [3].]]

Alice connects a new device.

Implementing a pure peer-to-peer protocol in which the desktop and tablet communicate directly would require both machines to be turned on at the same time and able to communicate. Experience has shown that this process is considerably more reliable when mediated by some form of broker.

Unlike traditional cloud services that encourage users to rely on and trust them to the greatest possible extent, the Mesh Portal is an untrusted cloud service. The Mesh cloud cannot be breached because Mesh cloud does not guarantee confidentiality, or integrity or availability.

There are three main functions that a Mesh Portal MAY perform:

- Acts as a mailbox to which connection requests and responses are posted.

- Acts as an intermediary to third party trust providers.

- Acts as an intermediary to credential publication services.

Only the first of these functions is required. But the protocol requirements for meeting the requirements of the first makes the other two a natural fit.

### 3.5. Why change is possible

All four of the open standards based PKIs that have been developed in the IETF are based on designs that emerged in the mid-1990s. Performing the computations necessary for public key cryptography without noticeable impact on the speed of user interaction was a constraint for even the fastest machines of the day. Consequently, PKI designs attempted to limit the number of cryptographic operations required to the bare minimum necessary. There were long debates over the question of whether certificate chains of more than 3 certificates were acceptable.

Today a 32-bit computer with two processing cores running at 1.2GHz can be bought for $5 and public key algorithms are available that provide a higher level of security than was achievable in the 1990s for less computation time. In 1995, the idea that a single user might need a hundred public key pairs and a personal PKI to manage them as an extreme scenario. Today when the typical user has a phone, a tablet and a laptop and their home is about to fill up dozens if not hundreds of network connected devices, the need to manage large numbers of keys for individual users is clear.

Use of public key cryptography on the scale used in the Mesh would have been impractical even for financial applications as recently as 15 years ago. Today, the performance and memory overhead is negligible.

## 4. Architectural Principles

Over the course of the first quarter century of commercial use of PKI, a consensus has emerged as to the principles of robust protocol design; All aspects of the design should be public, all cryptographic algorithms should be open standards that have been widely reviewed by domain experts, etc.

The Mathematical Mesh is appropriately aligned with this established consensus but departs from existing practice in certain areas as set out in the following.

### 4.1. User Centered Design

Traditional enterprise centered PKI keeps the enterprise (and to an extent the users) secure but only as long as the users follow a long list of often complex instructions. Even the US National Security Agency, an institution whose core competence is cryptography and whose principle purpose is to protect US government information failed to protect some of its greatest secrets because it was just too hard to use the right cryptography.

A key principle that guides the design of the Mesh is that any set of instructions that can be written down and given to a user can be written down as code and executed by the computer. Public key cryptography is used to automate the process of managing public keys. Instead of telling the user how to register for a certificate and install it in their mail client, we tell the computer how to do the task for them.

### 4.2. User Centered Trust.

One of the principal ideological battles that has been fought in the development of end-to-end email security has been the manner in which trust is provided. In the S/MIME protocol, trust is established through a hierarchy of trust providers. In the OpenPGP protocol, trust is in theory established through a 'Web of Trust' but is in practice almost invariably established through either a direct trust model (exchange of key fingerprints) or on a trust after first use basis.

Perhaps, what we should be willing to learn from the experience of attempting to apply these models to real world applications is that none is sufficient by itself. Rather than attempting to impose a single model of trust on every circumstance, multiple trust models should be supported.

The trust model appropriate for validating a key depends on the context in which it is to be used. If Alice is sending Bob a personal email, it is likely that the best key to use will be the one that matches the key fingerprint from the business card he gave her when they met in person. But when Alice is sending Bob an email as her stockbroker, the best key is going to be the one that is issued to Bob as an employee of the brokerage company.

Any trust model must be built around the needs of the user. The Mesh does not impose a model for mapping human to machine interaction identifiers but it does allow the user to put that mapping under their personal control. Devices connected to a Mesh Personal Profile share the same view of the world; the same set of bookmarks and

contacts for defining personal names and the same set of trust roots for Certification Authorities trusted to provide brokered trust.

In the traditional model, PKI is used to validate network hosts after discovery. The credential issued by the CA is verified each time the user visits the site.

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

Traditional PKI role.

In the Mesh trust model, the primary role of the CA is to provide introductions. When the user first visits the site, to buy goods, the site (and often the vendor it belongs to) is unknown. At this point, the user is looking to the Authority to help decide if they wish to purchase from.

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

Trusted Authority as Introducer

Once users can easily maintain a personal directory of trusted vendors and share it across all the devices they use, their personal trust directory becomes their primary trust provider. Thus, the role of the authority changes once a trust relationship has been established from trust provider to trust revoker. The user does not need the authority to tell them to trust a vendor they are already doing business with but they may need the authority to warn them if the vendor has defaulted on purchases made by other customers or has suffered a major breach, etc.

## 4.3. A Credential Designed for Persistence

One of the main difficulties in using S/MIME for email security is that many users stop using the system when their certificate expires. It is likely that one of the main reasons that OpenPGP is more popular amongst system administrators is that a maintenance-free user experience is available to anyone who decides to neglect key rollover.

A Mesh Master Profile fingerprint is designed to provide a user with a credential that can be used for a lifetime. Using the same

offline/online key management approaches that have been applied to root key management in the WebPKI since it began provides users with a credential with a cryptographic lifetime of 20-30 years. The addition of linked notary log technology in the Mesh Portals allows such credentials to be securely renewed should the need arise and thus enabling indefinite use.

## 4.4. Eliminate unnecessary options

Traditionally cryptographic applications give the user a bewildering choice of algorithms and options. They can choose to have one RSA keypair used for encryption and signature or they can have separate keys for both, they can encrypt their messages using 3DES or AES at 128, 192 or 256 bit security. And so on.

The Mesh eliminates such choices as unnecessary. Except where required by an application, the Mesh always uses separate keys for encryption and signature operations and only uses the highest strength on offer. Currently, Mesh profiles are always encrypted using RSAES-PKCS1-v1_5 with a 2048 bit key [RFC8017] , AES with a 256 bit key [FIPS197] and SHA-2-512 [SHA-2] . The use of RSA 2048 will be replaced with Ed448 [RFC8032] when sufficiently mature implementations become available.

For similar reasons, every Mesh master profile has an escrow key. The use of key escrow by applications is optional, but every profile has the capability of using it should circumstances require.

## 4.5. Strong Public Key Identifiers

A key departure from traditional PKI approaches is that all cryptographic keys are identified in every circumstance by either the UDF fingerprint of the public key in the case of a public key pair or the UDF fingerprint of the symmetric key in the case of a symmetric key pair. No other form of key identifier is used.

This approach greatly simplifies the processes of key discovery, management and signature verification.

Since a UDF fingerprint of sufficient length, uniquely identifies a public key pair, it follows that if the attempt to verify the signature under a public key whose fingerprint matches that specified returns the result false, that the signature is invalid. Contrawise, if the result returned is true, the data is validly signed for a given purpose if and only if the key identifier is authorized for that purpose.

## 5. Architecture

All information exchanged through the Mesh is described in a profile. Profiles have the following properties.

- Profiles are first class objects with a unique, immutable identifier that either is or contains a UDF fingerprint of a signature key

- All profiles are digitally signed.

- Profiles are encoded in JSON [RFC7159] .

At present, four types of Mesh Profile are defined:

Master Profile  Describes the criteria for validating a user's personal profile.

Personal Profile  Describes the user's personal Trust Mesh, the set of device and application profiles connected to it.

Application Profile  Describes the use of a particular application.

Device Profile  Describes a device and cryptographic keys specific to that device.

A profile is Valid if and only if it is Verified, Current and Correct:

Verified  A signature is verified if the key identifier specified maps to a

Current  A profile is current if and only if it has not been superseded by a new profile published to the Mesh Portal.

Correct  A profile is correct if the signing key identifier specified in the signature data is a legitimate signing key for that type of profile as specified in Section YY below.

A profile is connected to a personal profile if and only if:

- The Personal Profile contains an entry for its profile identifier that is consistent with the profile type.

- The Personal Profile is Valid.

- The Application Profile is Valid.

This trust model allows Application Profiles and Device Profiles to be connected to a Personal Profile by enumerating the identifiers of the connected profile in the personal profile.

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

A Master/Personal Profile connected to Application and Device profiles.

## 5.1. Master and Personal Profiles

Each Mesh user has a Master Profile and a Personal Profile.

Personal Profile Contains a list of all the device profiles and application profiles that are currently connected to the user's personal Mesh. The personal profile is signed by an administrative key.

Master Profile Contains a list of administrative keys used to sign personal profiles and the master signature key used to sign the Master Profile.

The use of master signature keys and administration keys to authenticate the Master and Personal profiles needs further explanation.

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

Master and Personal Profile Signature

This separation allows the user to add devices and/or change application settings frequently without the need for changes to their master profile or to access their master signature key. This allows the master signature key to be stored in a safe place, preferably using a Hardware Security Module or other precautions without the inconvenience this would entail if regular use of the key was required.

A typical user might modify their personal profile hundreds of times over their lifetime, conceivably even more in a world where homes are filled with hundreds of IoT devices. Adding or removing administrative devices is likely to occur much less frequently. The

master signature key used to authenticate the Master Profile never changes. If it becomes necessary to replace the master signature key, it will be necessary to create a new Master profile and perform a secure transition to the new key.

Since the master signature key does not change by definition, the fingerprint of the master signature key is a persistent identifier that will remain constant and only ever refer to exactly one Master Profile. Furthermore, since at any given time a Master Profile has exactly one Personal Profile attached to it (and vice versa), the fingerprint of the master signature key is a persistent identifier for the Personal Profile as well.

## 5.2. Device Profiles

A device profile contains a description of the device and the public keys to be used as the basis for encryption, authentication and signature on that device (Figure XXX). Ideally, the private keys associated with the device are generated using a secure procedure on the device itself and are bound to the device so that they cannot be exported from it.

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html [8].]]

Device Profile

Application profiles MAY specify additional profiles to be used with a particular device for an application specific purpose.

While it is usually desirable for such keys to be generated on the device itself, this is not always possible. If, for example, the application profile is connected to a personal profile with multiple existing devices. In this circumstance, the use of a co-operative key generation approach [PHB-CoKey] is preferred but not always possible. If no other options are available, additional application private keys may be provisioned to the device by encrypting them under the device private key.

Device profiles are connected to a personal profile by enumerating them in the Personal Profile.

### 5.3. Application Profiles

An application profile describes the configuration of an application. An Application Profile MAY contain:

Public data  Information that is intended for public use that applies to all devices.  For example, the user's public encryption key for end to end secure email.

Private Data  Information that applies to all devices that is only intended for use by connected devices.  For example, the network configuration information describing how to access the inbound and outbound mail servers.

Public Per Device Data  Information that is intended for public use that applies to a specific device.  For example, the user's public signature key might be different for each device.

Private Per Device Data  Information that applies to a specific device and is only for the use of that device.  For example, a per-device signature key.

Application Profiles are connected to a Personal Profile by an Application Device Entry.  The Application Device Entry specifies the identifier of the device and the set of privileges delegated to it with respect to that application.  These privileges MAY include the privilege of signing Application Profile updates.  This allows a device that is not an administration device for the personal profile to be permitted to update the application profile.  Thus, Alice might not want her laptop to be an administration device but would likely want to be able to add or update Web Site usernames and passwords.

### 5.4. Verifying Profiles

As described in section XXX, the key identifier in a Mesh Profile signature is the UDF fingerprint of the public key to be used for verification.  Therefore, a Profile is invalid if either:

Attempting to validate the profile under a public key whose UDF fingerprint matches that specified in the key identifier returns the result false.

The Key Identifier specified in the signature is not explicitly authorized for the purpose as described below.

The Key Identifiers authorized to sign a profile depend on the type of profile as follows:

Master Profile The Key Identifier of the key that signs the profile MUST be the UDF fingerprint of the Master Signature Key specified in that Master Profile.

Personal Profile The Key Identifier of the key that signs the profile MUST be listed as the identifier of an Administrator Key specified in the corresponding Master Profile.

Device Profile The Key Identifier of the key that signs the profile MUST be the UDF fingerprint of the Device Signature Key specified in that Device Profile.

Application Profile The Key Identifier of the key that signs the profile MUST be listed as the identifier of an Administrator key for that Application Profile in the corresponding Personal Profile.

### 5.5. Key Escrow and Recovery

Key Escrow and Recovery capabilities are built into the core of the Mesh. Use of these capabilities is RECOMMENDED but not required.

Encryption of stored data such as email messages and personal photographs protects confidentiality but introduces a major availability risk: The data will be lost if the user loses access to the decryption key. While the risk of being coerced into disclosure of material is a risk for some users, availability is a concern for all users.

The Mesh supports two types of key escrow, Offline and Online.

Offline Key Escrow Is used to escrow Master Signature Keys, Master Escrow Keys and other private keys that are particularly important to a user and are to be preserved.

Online Key Escrow Is used to protect all other keys by escrowing the key under a Master Escrow Key.

#### 5.5.1. Offline Escrow

The use of Shamir Secret Sharing [ShamirXX] to escrow private keys is supported as follows:

- A 256 bit random session key is generated.
- The private key and related data is encrypted under the session key to form the key escrow data blob.

- The unique identifier for the key escrow data blob is the UDF fingerprint of the session key.

- The key escrow data blob is stored in some form of persistent storage that is believed to provide a very high degree of availability.

- The session key is split into n of m shares using Shamir secret sharing.

Since the purpose of the Mesh Portal is in part to provide a high availability data storage facility, the use of the Mesh to store the key recovery blob is preferred.

Data recovery is the reverse of the escrow process:

- The shared secret is recovered from at least n of the key shares.

- The unique identifier for the key escrow data blob is calculated from the session key

- The unique identifier is used to retrieve the key recovery blob.

- The key recovery blob is decrypted using the session key.

It will be noted that this process is anonymous. The key recovery blob does not identify the profile or user to which it refers in any way.

### 5.5.2. Online Escrow

Support for Online escrow requires only that decryption keys for application use be encrypted under the master escrow key as if it was another device connected to a Personal Profile.

## 6. Mesh Portals and the CryptoMesh

As described earlier, a Mesh Portal is an untrusted cloud services that facilitates the management of Mesh profiles by providing persistent storage. A personal profile MAY be registered to one, many or no Mesh Portals at a time.

For the sake of convenience and familiarity, the Mesh Portal Protocol makes use of account identifiers in the traditional [RFC5322] format <user>@<domain>. Transactions supported by the Portal Protocol include:

- Create Account

- Add profile
- Update profile
- Request device connection*
- View pending requests*
- Accept or reject a connection request*

Transactions marked with an asterisk* require administration privileges for the personal profile.

Although the Mesh Architecture regards the Mesh Portal to be an untrusted cloud service with respect to Confidentiality and Integrity, the information transferred between devices and the portal could be susceptible to traffic analysis. For this reason, all exchanges between devices and the portal MUST be protected using a transport layer security enhancement such as TLS [RFC5246] .

## 6.1. Federated Portals

A Mesh Portal MAY belong to a Federation. Portals belonging to a Federation periodically synchronize their transaction logs so that all the members of the federation have access to the complete set of transaction logs for all the portals belonging to the federation. This allows the federation to collectively guarantee the availability of the user's profile data should one or more portals become unavailable either temporarily or permanently.

The InterMesh protocol is supports Portal Federation.

The CryptoMesh is proposed open federation of Mesh Portals that permits any portal willing to accept its terms of service to join. Due to the nature of the network effect it is expected that there would be only one such open federation baring major disagreements as to terms of service. Figure XXX

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

The Cryptomesh.

Mesh Portals that are a member of a Federation have a mutual responsibility to protect availability by acting to mitigate abuse by

users attempting to create excessive numbers of profiles or perform excessive numbers of updates.

## 7. Security Considerations

NYI

## 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. References

### 10.1. Normative References

[draft-hallambaker-json-web-service]
Hallam-Baker, P., "JSON Web Service Binding Version 1.0", draft-hallambaker-json-web-service-07 (work in progress), September 2017.

[draft-hallambaker-jsonbcd]
Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", draft-hallambaker-jsonbcd-08 (work in progress), September 2017.

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

[draft-hallambaker-mesh-confirm]
Hallam-Baker, P., "Mesh Confirmation Protocol (Mesh/Confirm)", draft-hallambaker-mesh-confirm-01 (work in progress), August 2017.

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", draft-hallambaker-mesh-developer-04 (work in progress), September 2017.

[draft-hallambaker-mesh-platform]
Hallam-Baker, P., "Mathematical Mesh: Platform Configuration", draft-hallambaker-mesh-platform-00 (work in progress), September 2016.

[draft-hallambaker-mesh-recrypt]
Hallam-Baker, P., "Mesh/Recrypt: Usable Confidentiality", draft-hallambaker-mesh-recrypt-02 (work in progress), August 2017.

[draft-hallambaker-mesh-reference]
Hallam-Baker, P., "Mathematical Mesh: Reference", draft-hallambaker-mesh-reference-06 (work in progress), September 2017.

[draft-hallambaker-sin]
Hallam-Baker, P., "Strong Internet Names (SIN)", draft-hallambaker-sin-00 (work in progress), August 2017.

[draft-hallambaker-udf]
Hallam-Baker, P., "Uniform Data Fingerprint (UDF)", draft-hallambaker-udf-06 (work in progress), August 2017.

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

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015.

[RFC8017] Moriarty, K., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016.

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

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

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

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

## 10.2. Informative References

[PHB-CoKey] "[Reference Not Found!]".

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007.

[RFC5322] Resnick, P., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011.

## 10.3. URIs

[1] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[2] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[3] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[4] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[5] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[6] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[7] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[8] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

[9] http://prismproof.org/Documents/draft-hallambaker-mesh-architecture.html

## Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com