Internet Engineering Task Force (IETF)
Internet-Draft
Intended Status: Standards Track
Expires: April 20, 2014
Phillip Hallam-Baker
Comodo Group Inc.
October 17, 2013

PRISM_Proof Email Key Generation and Publication
draft-hallambaker-prismproof-key-00

Abstract

This document describes previous efforts and their deployment legacy and the requirements for a successful email security infrastructure. A gap analysis is performed and the tasks divided into problems that are generally considered solved albeit possibly requiring improved execution and problems that may be regarded as research.

This division of the problem space into 'execution' and 'research' portions allows different groups of developers to address each independently and avoid unnecessary duplication of effort. A testbed for development and early adopter deployment that achieves this separation is described.

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 http://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."

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Problem Statement

Generating a public keypair and registering it for use should be the only occasion on which a user is required to think about their cryptographic security. Nor should the user be required to think too much in this circumstance either.

To enable others to send encrypted email to them, a user must at minimum generate at least one public keypair and make the public key portion available to the intended communit of potential senders. The precise means by which this is achieved may be considered a hard research problem. Accordingly this specification anticipates such processing being performed 'in the cloud' (i.e. by magic) and describes a Web Service interface that may be used to

1.1. Legacy Infrastructure

Twenty years of effort attempting to deploy secure email has left a considerable legacy of deployed code. While this deployed code base is not ideally suited to the task (or the problem would be solved already) it is generally better to support use of such deployed resources where they exist rather than attempt to build everything from scratch.

One significant design consequence that flows from this approach is to adopt ASN.1 encoding for cryptagraphic data objects, including the Key Endorsement object described in this document. While there are many better choices of data encoding and remarkably few that are worse, most cryptographic toolkits provide support for parsing X.509v3 certificates and generating Certificate Signing Requests and many provide comprehensive support for a wide range of ASN.1 encoded objects.

2. Key Generation and Identification

2.1. Strong Key Identifier

A Strong Key Identifier is an identifier that identifies a unique public key formed using a strong Message digest function over the public key parameter values.

This definition of Key Identifiers is considerably more restrictive than the PKIX definition which allows an issuer to use any unique string for the subjectKeyIdentifier and authorityKeyIdentifier extensions.

Compliant certificate issuers SHOULD use Strong Key Identifiers as specified in this document for PKIX Key Identifiers.

A strong Key Identifier takes one of the two following forms:

If the length of the Key Identifier is exactly 20 octets.
: The Key Identifier is an OpenPGP v4 Key fingerprint calculated as specified in [!RFC4880]

Otherwise
: The first byte specifies the digest algorithm and the following bytes the digest value calculated over the DER encoded SubjectPublicKeyInfo.

The following algorithm values are assigned in this document:

0
: SHA-2-512 truncated to 128 bits.

1
: SHA-2-512 truncated to 224 bits.

2
: SHA-2-512 truncated to 256 bits.

3
: SHA-2-512 without truncation

128-255
: Reserved for use in a future multi-byte algorithm identifier scheme.

To prevent a downgrade attack in which an attacker truncates a longer Key Identifier, the input to the message digest function is prepared as follows:

Let V be the algorithm identifier value and D be the DER encoded SubjectPublicKeyInfo and + stand for simple concatenation.

Key Identifier = H (V + D)

If it is necessary to present a Key Identifier to an end user, Base32 encoding is used. Additional dash (-) characters MAY be added to improve readability and MUST be ignored by compliant applications.

2.1.1. Strong Email Addresses

To establish encrypted communications it is necessary to know a public key for the recipient and the recipient's security policy. The fact that a recipient is capable of receiving encrypted email does not mean that they are capable of receiving encrypted email on every device they use or that they are willing to accept encrypted email from every sender.

A similar problem was faced when using Transport Layer Security [RFC5246] with HTTP [RFC2616]. By default, Web requests are sent without use of security. To force use of TLS, the URI method https is used in place of http. The security policy is encoded in the URI.

Strong email addresses allow an email sender to encode the security policy in an RFC822 [RFC2822] compliant email address. RFC822 defines the 'user name' portion of an email address as follows:

```
addr-spec       =       local-part "@" domain
local-part      =       dot-atom / quoted-string / obs-local-part
atext           =       ALPHA / DIGIT /
                        "!" / "#" /
                        "$" / "%" /
                        "&" / "'" /
                        "*" / "+" /
                        "-" / "/" /
                        "=" / "?" /
                        "^" / "_" /
                        "`" / "{" /
                        "|" / "}" /
                        "~"
atom            =       [CFWS] 1*atext [CFWS]
dot-atom        =       [CFWS] dot-atom-text [CFWS]
```

In a Strong Email Address, the character '?' is reserved. Although this is a legitimate account name in some operating systems, use is prohibited in current editions of Windows and most UNIX based operating systems.

The address syntax is modified as follows:

```
addr-spec       =       local-part "@" domain
local-part      =       dot-atom / quoted-string /
                            obs-local-part / strong-local
atext           =       ALPHA / DIGIT /
                        "!" / "#" /
                        "$" / "%" /
                        "&" / "'" /
                        "*" / "+" /
                        "-" / "/" /
                        "=" /
                        "^" / "_" /
                        "`" / "{" /
                        "|" / "}" /
                        "~"
strong-local    = indirect-key / direct-key / nokey

ktext = ALPHA / DIGIT / "-"
key-identifier = 1*ktext

indirect-key    =   key-identifier "??" dot-atom
direct-key      =   key-identifier "?" dot-atom
nokey           =   "?" dot-atom
```

Addresses of the form indirect-key, direct-key and nokey are interpreted as follows:

nokey
: Messages sent to the address MUST be encrypted under an encryption key that the sender determines to be trustworthy.

direct-key
: If the public key specified by the Key Identifier is an encryption key, messages sent to the address MUST be encrypted under the specified key. Otherwise messages sent to the address MUST be encrypted under an encryption key that has a direct key endorsement under the specified key.

indirect-key
: Messages sent to the address MUST be encrypted under an encryption key that has a key endorsement under the specified key.

2.2. Private Key Backup and Controlled Recovery

A frequently overlooked hazzard of using encryption is the risk of data loss should the private key be lost or otherwise become unavailable. Another practical difficulty that must be faced is the need to enable encrypted email to be read on more than one device.

Once published, a strong email identifier effectively becomes a personal root of trust, the value of which may increase over time.

Each of these use cases requires some form of private key backup and recovery mechanism. While such mechanisms have traditionally been considered to be an implementation choice that is outside the scope of a protocol specification, to do so incurs a substantial risk of a large number of bad implementation choices. In particular the need to enable receipt of email on multiple devices requires a standards based approach or else applications provided by different vendors will not be able to exchange keys.

While a Key Escrow capability provides a Key Backup capability, the reverse is not true. A Key Escrow system is generally understood to support recovery of the private key without notice to the private key holder while a Key Backup system need not meet this requirement.

A publication service MAY support Key Backup and Recovery. A user MAY choose to use the Key Backup and Recovery function supported by a Publication service.

If Key Backup is used, the key management client encrypts the private key under a strong symmetric key and sends the encrypted data to the publication service. The information necessary to recover the private key is presented to the user in a compact form that MAY be written down and stored without risk of hardware failure rendering the key inaccessible.

2.2.1. Encrypted Private Key

Private Keys are encrypted using the PKCS#8 format as specified in [RFC5208].

This specification is prefered to the PKCS#12 [I-D.moriarty-pkcs12v1-1] format as the latter is essentially a wrapper for multiple PKCS#8 keys and associated certificates that can be generated by a publication service if necessary.

Key management tools MUST support the use of AES-256 to encrypt private keys. AES is prefered over AES-128 for the greater number of encryption cycles rather than the increased brute force work factor. Applications MAY use encryption keys with lengths less than 256 bits provided that the keys have a length of at least 128 bits.

If the key size used is shorter than the key size required by the encryption algorithm, the HKDF-Expand function described in [RFC5869] is used to expand the truncated key to provide the necessary number of bits.

Keys are presented in BASE32 encoding [RFC4648] with optional separators '-' to improve readability. Applications MUST ignore separators when decoding the keys.

2.2.2. Key Splitting

Key Management tools MAY support the use of a key splitting scheme to allow greater control over key recovery. For example, the user might split their key into three parts with a requirement that two parts are necessary to reconstruct the key.

At this point the author has a paper by Rober Blakely Snr on an out-of-patent key splitting scheme but insufficient time to read the paper let alone write and implement the specification. If anyone is looking for something to do, that would be useful.

2.3. Private Key Example

Alice uses a key generation tool to generate a public keypair. The public parameters in hexadecimal are:

```
Modulus  :
 c1 66 de 02 62 35 3f af 7a 22 11 66 62 5a 1b 8d
 3b 85 14 65 32 5d 6c e0 b5 db 09 e0 fc e4 16 34
 96 ac 5b 76 01 96 e4 37 d5 8b db 52 a7 71 68 1c
 86 1a 61 58 a7 0a 91 14 f2 d9 cd 4a 6b a5 e2 b3
 94 c9 0b f2 7b ff 3b 6e a8 7b bf ca 27 0e b2 28
 b0 d5 4a 1b 59 9a 8b 40 4e 80 3b dd 79 57 25 52
 7a 70 ba 22 02 45 7b 4c e8 95 69 34 79 77 86 5f
 09 36 30 18 1b 77 be c5 dc d3 ea db 1b 0a a0 8f
Exponent :
 01 00 01
```

2.3.1. Key Identifier

KeyIdentifier: ACAIEA-FONPAC-5AC6LFA-K4ACHC-EAJWAHN-VPAM4A-COYPAO-VAA

alice@example.com
: Send email to Alice using encryption if and only if an encryption key for Alice can be found and Alice has published the email encryption policy 'encryption preferred' or stronger.

?alice@example.com
: Send email to Alice using encryption if and only if an encryption key for Alice can be found, otherwise report an error.

ACAIEA-FONPAC-5AC6LFA-K4ACHC-EAJWAHN-VPAM4A-COYPAO-VAA?alice@example.com
: Send email to Alice using encryption if and only if an encryption key for Alice can be found that is directly endorsed

under the specified key, otherwise report an error.

ACAIEA-FONPAC-5AC6LFA-K4ACHC-EAJWAHN-VPAM4A-COYPAO-
VAA??alice@example.com

Send email to Alice using encryption if and only if an encryption key for Alice can be found that is (directly or indierectly) endorsed under the specified key, otherwise report an error.

2.3.2. Private Key Backup

The private key component of Alice's key is as follows:

```
P  :
 f3 85 24 7b 95 3d a1 77 7c a4 4d a8 b8 00 3e 73
 b2 9d 36 52 dc 64 21 e2 90 56 3c 51 d6 24 0c 20
 77 1e d1 35 b4 c8 77 00 86 96 af 66 b0 5e 31 ff
 15 ef 40 5e 00 21 54 18 fb dd f6 c2 bc 93 c2 1d
Q :
 cb 50 32 f4 eb b5 74 80 b0 d1 f6 41 8c 90 9f 56
 50 19 4e 64 be 93 f0 a2 bc 3c e9 e6 48 56 99 4e
 4e 33 9c 77 31 92 45 a6 aa 35 39 7c f8 aa f3 35
 85 05 09 78 8a 9f 4d 90 e3 36 61 84 ec 39 2d 9b
DP  :
 ca fa 35 58 95 22 d3 cd 66 a5 04 de 16 d0 8d 3d
 9e a9 8f b8 2d 5f 81 26 f9 ac 07 87 26 f8 d0 ea
 d6 9f 67 3e 5e bb a1 05 5d 29 88 76 0d 97 d6 10
 8a d5 eb 4e ee c8 d8 f2 22 2d f7 1a 86 58 9a b9
DQ :
 73 74 37 7b 9d de 8d 2a 07 3f 33 f8 45 3a 5b 41
 48 7b 16 69 5f 4f e3 76 86 2e 91 24 94 2f 99 1f
 3e 89 50 70 df 55 90 f7 f3 f0 05 95 52 20 c1 bb
 c2 ad f9 92 da 25 5c 86 ca 80 37 20 a4 84 53 c1
InverseQ :
 c1 b9 4b bf ee 41 77 b9 dc 0c cd 97 c0 96 77 22
 0d e0 ed b2 3f 02 25 63 c8 0d 86 d8 5c 44 df 4d
 d5 d7 3e 78 4a 5f 3d cb 76 5a 9b a8 1a 68 8e 47
 9c 47 f9 8f 81 8d 6b 99 ab 74 56 88 4f ac c3 88
```

The private key is encrypted under a randomly assigned symmetric key using PKCS8 encoding.

```
30 82 02 88 30 02 05 00 04 82 02 80 f3 fa 63 b9
1c d3 60 54 2f 75 ca 99 fe 42 1f 21 0d 2c f9 bd
4a 72 e4 ba da 09 91 f3 96 b7 b8 4a b6 78 da bc
92 55 ce c0 77 7c 75 96 86 05 cf 21 1b 23 a6 c6
12 fc e6 2c a5 36 7f 14 b3 bc 53 70 8a 8e fe 7f
99 d6 1d da 00 5f 5b 43 b2 cf 2d eb 0f 23 9c ce
0b bd 9c 81 29 b9 b8 7d 78 35 55 f7 45 5e 7b e0
d6 ef 9b bd 79 51 be 6d 88 f7 63 bb ef c8 b9 5b
90 c2 e9 a1 b5 d2 7b dd 69 95 3b 55 3a 79 8f 70
f4 26 38 4e 40 50 43 14 8c 57 65 7c cc 37 6a e2
4d 2f 51 fe 06 05 3b 7c 60 47 58 01 ef c6 f1 ae
4c 3c 28 8c c5 f0 0b f6 dd 8f ff 5d 22 a8 b6 5e
1b 94 29 ad f5 63 2e b8 60 ec 96 c8 63 df 2b 50
8b 27 a9 da ff 4a f8 b1 7d 6f 30 4c 9b 3d f6 65
a0 3e 24 6d 6f 2e d5 37 a4 52 ef 5a ef 11 51 84
e3 7e d2 19 7e 86 34 22 c5 78 5e 9e 6f 40 10 76
b3 cd 34 dd ea 3b 0e fc f3 38 1e e6 a3 32 54 a1
1a 7e 51 8d 0c 99 2e e7 20 06 21 5e b9 f8 87 19
f1 cd 82 00 6f 72 fb b9 a3 85 21 fb ac 80 2b aa
3a 47 b0 5d 03 74 77 08 70 de 64 25 a3 f5 bb 97
a3 08 ff 29 db 17 7b fa f8 80 c0 4e 90 5d 9a 15
04 60 73 f6 47 ff 82 6b 16 ce 19 a2 a0 1e a1 a3
a0 b0 2a fa 5b 51 0b 0c ab 92 53 fe 1e 1d af d9
78 9f 70 26 a6 32 80 d1 ef 6d 67 2a 48 b1 4b 3d
76 cf bc 8e 48 0c b0 9d da cf 4c b2 be aa d3 5f
ed 91 36 76 ef 5e ef 95 9c fd 4f 72 46 1d cf d2
15 4c 1a 93 9f 52 84 ce a5 c8 17 ff bc 0e 70 83
fc d3 c9 98 b6 0d f0 a8 72 8d 2d d4 49 6d 9d 79
35 c7 48 36 b1 49 95 f0 02 77 52 7a 50 13 74 80
7b 9d 1b bf cf f2 1e 99 6e 92 8f 8b b2 d8 35 6c
c9 2f 5d 3d da 4c 53 49 03 d6 63 56 7b d6 80 5d
69 b6 8d 9b fd 50 f4 c8 5e 9a 3c 91 26 b7 0a 62
d3 a7 2d 99 49 b2 1b 0f 74 71 62 ce 41 8a 2c dd
8a b4 97 38 dc be b8 6f b6 ba 29 e4 73 2d c0 ee
23 d1 2b 97 eb 7c 8d 42 16 33 3d 93 84 31 59 2f
f7 26 78 f2 3f 9a af ff 26 81 da 34 a6 74 bf 35
a9 c5 4d 6a 9d 48 d8 4b 00 a5 56 c2 46 e2 ce 65
f8 86 22 75 07 32 df 69 2c e2 74 09 54 4a 1e 38
62 56 6e 8e be c4 23 78 c1 f4 ea 20 96 a7 ac 89
54 6f 2d 0f 73 53 3b 66 7a 61 69 e7 6a d0 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```


The cipher (specified in the PKCS8 object) is AES-256. The password value in Base-32 encoding is:

Passcode: H2AL6A-ESJYAE-JABNDMA-HAAANDG;

1/3: CFAHVA-FMUWAN-PAHXIZA-BAAH2FE
2/3: PCAP7A-BEBNAH-LACDFJA-OAAHZXE
3/3: DBABPA-EHTBAD-CAGVAZA-MIAGPXE

3. Public Key Infrastructure

The precise means by which a public key is validated by a relying party is outside the scope of this specification. Keys MAY be validated by a traditional Certificate Authority or through peer to peer endorsement or any combination of the two.

In order to maximize the flexibility for the trust infrastructre designers, two syntaxes for presenting public keys for use are supported. Key Management tools SHOULD support both:

A Certificate Signing Request
: May be presented to a CA or other signer.

A self signed certificate
: Presents the public key in a form that many Internet applications accept directly.

3.1. Certificate Signing Request.

Certificate Signing Requests SHOULD conform to the following profile:

* The Key Identifier MUST be specified and MUST be a strong key identifier

* [[Prohibit various PKIX lunacies]

3.2. Self-Signed Certificate.

Self Signed Certificates SHOULD conform to the following profile:

* The Key Identifier MUST be specified and MUST be a strong key identifier

* [[Prohibit various PKIX lunacies]

3.3. Peer Endorsement

Traditionally PKIX only permits use of Certification Authority provided trust assertions while OpenPGP only permits use of peer endorsement through key signing. PPE supports the use of a combination of both approaches for reasons described in [I-

To perform peer endorsement, the following data structure is used:

```
Class Endorsement
     TBSEndorsement                TBSEndorsement
     SignatureAlgorithm            AlgorithmIdentifier
     Signature                             Bits

Class TBSEndorsement
     Version                               Integer
     Issued                                Time
     IssuerKeyIdentifier           Octets
     SubjectKeyIdentifier    Octets
     Subject                               List Name
     SubjectAltName                List SubjectAltName
     Extensions                            List Extension

Class AlgorithmIdentifier
     Algorithm                             OIDRef
     Parameters                            Any

Class Name
     Member                                Set AttributeTypeValue

Class AttributeTypeValue
     Type                                  OIDRef
     Value                                 AnyString

Object SubjectAltName id_ce_subjectAltName
     Names                                 List GeneralName

Class GeneralName
     Value                                         Choice
             RFC822Name                                    IA5String
                     Code 1
                     Implicit
             DNSName                               IA5String
                     Code 2
                     Implicit

Class Extension
     ObjectIdentifier              OIDRef
     Critical                              Boolean
             Default "false"
             Optional
     Data                                  Octets
```

[[Note that although my tool generates ASN.1 encoding this is for purely pragmatic reasons of providing consistency. It is not meant to in any shape or fashion stand for an endorsement of this crackpot technology.]

A new structure is introduced to support Key Endorsement rather than attempting to re-use the X.509v3 Certificate format in recognition of key endorsement having distinctly different semantics from issue of PKIX certificates. PKIX certificates are either end entity certificates or certificate signing certificates. A PKIX certificate is expressly prohibited from being used for both purposes. In the PKIX model, finding a certificate chain to a trusted anchor is necessary and sufficient to establish the trustworthiness of an end entity certificate. In the Key Endorsement model the reliance on a single key endorsement MAY be qualified by the age of the endorsement, the circumstances of issue, the number of independent trust paths from the relying party to the subject and the lengths of each path.

Most of the fields in the TBSEndorsement structure have the same semantics as in PKIX with the exception of the Validity interval which is replaced by the time of issue.

The precise mechanism by which endorsement is used requires further development. At minimum, the endorsement mechanism should allow the following forms of endorsement to be differentiated:

Direct Endorsement
: A endorsement of a user's key identifier by another key held by the same user. This form of endorsement allows a user to establish a personal master key that is only used for the purpose of endorsing keys for specific uses (email encryption, email signature, endorsement, etc.)

Peer Endorsement
: A user endorses the key identifier of another user (the subject) and possibly other aspects of the subject's identity such as their name, likeness etc. Such an endorsement SHOULD specify the basis for the endorsement (in person, remote, recent acquaintance, verification of government documents, childhood friend, etc.)

Group Endorsement
: One of the use practices that has emerged from attempts to employ PGP is the 'key party' in which groups of users perform mutual keysigning.

Withdrawing an Endorsement
: In certain circumstances, it MAY be necessary to withdraw an endorsement. The reason for withdrawing the endorsement SHOULD be specified in the UnEndorsement notice and MAY include, notification of the loss of the private key, the subject is deceased, etc.)

4. Publication Service

The Publication Service is a JSON/REST Web Service layered over HTTP transport. Although the publication service performs an important service, it is not a service trusted by the user since the publication service has no access to the user's private key (except in encrypted form) and does not sign any data that is read by the user.

The Publication Service is one of the two interfaces between the part of the email message security problem that is well understood and the part that is widely regarded to be 'research'.

Selection of the publication service MAY be left to individual user choice or a domain name holder MAY specify that publication requests be directed to a specific publication service. Users of a public email service are likely to want to insist on their own choice of publication service while a bank or government enterprise that has deployed its own security infrastructure is likely to want to insist that only credentials they approve are accepted for their site.

To allow researchers the widest possible lattitude in developing new trust infrastructures, publication of three trust assertion formats are supported together with support for key backup and recovery. These assertion formats are:

Self Signed Certificate
: A PKIX self signed certificate which MAY be used in conjunction with an existing application that accepts public key information in self signed certificate form.

Certificate Signing Request
: A PKCS#10 Certificate Signing Request conforming to [!RFC2986]. A publication interface MAY forward the Certificate Signing request to a Certificate Authority for issue of a PKIX end entity certificate.

Key Endorsement
: A Key Endorsement in the format described in this document.

4.1. Initial Key Publication

The first time that the Publication Service is used is after the user generates a new keypair.

For example, Alice registers the keypair generated in the previous example with her chosen Publication Service. Her key management tool makes an Assert request to the service with the following information:

* The Strong Key Identifier

* The Encrypted Private Key

* A Self-Signed Certificate

* A Signed Certificate Signing Request

* Service information describing the email service parameters to be used when sending messages using the corresponding email account. [[Which really should be encrypted, shouldn't they?]

5. Registration Example

Request


```
{
  "AssertRequest": {
    "KeyIdentifier": "
AJqCYq5r2i7DXllBrhJHESWbLe2rw84cTsPr6qo",
    "EncryptedKey": {
      "EncryptedPrivateKey": "
MIICiDACBQAEggKA8_pjuRzTYFQvdcqZ_kIfIQ0s-b1KcuS62gmR85a3uEq2eNq8
klXOwHd8dZaGBc8hGyOmxhL85iylNn8Us7xTcIqO_n-Z1h3aAF9bQ7LPLesPI5zO
C72cgSm5uH14NVX3RV574Nbvm715Ub5tiPdju-_IuVuQwumhtdJ73WmVO1U6eY9w
9CY4TkBQQxSMV2V8zDdq4k0vUf4GBTt8YEdYAe_G8a5MPCiMxfAL9t2P_10iqLZe
G5QprfVjLrhg7JbIY98rUIsnqdr_SvixfW8wTJs99mWgPiRtby7VN6RS71rvEVGE
437SGX6GNCLFeF6eb0AQdrPNNN3qOw788zge5qMyVKEaflGNDJku5yAGIV65-IcZ
8c2CAG9y-7mjhSH7rIArqjpHsF0DdHcIcN5kJaP1u5ejCP8p2xd7-viAwE6QXZoV
BGBz9kf_gmsWzhmioB6ho6CwKvpbUQsMq5JT_h4dr9l4n3AmpjKA0e9tZypIsUs9
ds-8jkgMsJ3az0yyvqrTX-2RNnbvXu-VnP1PckYdz9IVTBqTn1KEzqXIF_-8DnCD
_NPJmLYN8KhyjS3USW2deTXHSDaxSZXwAndSelATdIB7nRu_z_IemW6Sj4uy2DVs
yS9dPdpMU0kD1mNWe9aAXWm2jZv9UPTIXpo8kSa3CmLTpy2ZSbIbD3RxYs5Biizd
irSXONy-uG-2uinkcy3A7iPRK5frfI1CFjM9k4QxWS_3JnjyP5qv_yaB2jSmdL81
qcVNap1I2EsApVbCRuLOZfiGInUHMt9pLOJ0CVRKHjhiVm6OvsQjeMH06iCWp6yJ
VG8tD3NTO2Z6YWnnatAAAAAAAAAAAAAAAAAAA"},
    "Certificate": ["
MIICKjCCAZ4CAQICEQDPmUDPAObF9gNRAiPqkCswMAIFADAEMAIxADAeFw0xMzEw
MTYxMjAwMDFaFw0zMzEwMzEwNjA2MDJaMAQwAjEAMIGUMAIFAAOBjQAwgYkCgYEA
wWbeAmI1P696IhFmYlobjTuFFGUyXWzgtdsJ4PzkFjSWrFt2AZbkN9WL21KncWgc
hhphWKcKkRTy2c1Ka6Xis5TJC_J7_ztuqHu_yicOsiiw1UobWZqLQE6AO915VyVS
enC6IgJFe0zolWk0eXeGXwk2MBgbd77F3NPq2xsKoI8CAwEAAQUABQAwgbowKAMO
HVUCAQAEHwQdAJqCYq5r2i7DXllBrhJHESWbLe2rw84cTsPr6qowLwMjHVUCAQAE
JjAkBB0AmoJirmvaLsNeWUGuEkcRJZst7avDzhxOw-vqqgUAAgEAMCQDER1VAgEA
BBswGTAXMBUFABYRYWxpY2VAZXhhbXBsZS5jb20wDwMPHVUCAf8EBgMEAAcAgDAW
AyUdVQIBAAQNMAswCQgEAwcFBQEGKzAOAxMdVQIBAAQFMAMCAQAwAgUAA4GBAGMo
0Ky-ccYSHWqRLbd4JFns3UVgEbcbGUzm-H29DEJq1WUuihR03dfzeXQv9BY271o_
Q_RsuFIbOYpEhpP2OG_5v6DdLOrvE6GsjydN7isLo0E6F-rxkVP6GfyMiDI5cr9z
1IR9b--DZUx_C8QK1c4JcASVANMc_Yt7_yn7kDkD"],
    "CertificateRequest": ["
MIIBLTCBogIBADAEMAIxADCBlDACBQADgY0AMIGJAoGBAMFm3gJiNT-veiIRZmJa
G407hRRlMl1s4LXbCeD85BY0lqxbdgGW5DfVi9tSp3FoHIYaYVinCpEU8tnNSmul
4rOUyQvye_87bqh7v8onDrIosNVKG1mai0BOgDvdeVclUnpwuiICRXtM6JVpNHl3
hl8JNjAYG3e-xdzT6tsbCqCPAgMBAAEFADACBQADgYEABExEqqopLQXfVWZr5MJ0
digUmdcugrfykTnNMkLx3En8fVLMbrgBEu0Ndax_TqOk36_gjnyyjg2XGCI5BaTd
jHp13C8dsIdcfdePc3droSGLuPzMosZqzyN1qLhf5dEfXwp32gBwteXPV-YE9Nf3
rDEZ_vc32sK-09766Fbitz0"],
    "Service": [{
        "Email": "alice@example.com",
        "Name": "smtp.example.com",
```

"Protocol": "_smtp._tls",
"Port": 587,

```
          "TLS": true},
        {
          "Email": "alice@example.com",
          "Name": "imap.example.com",
          "Protocol": "_smtp._tls",
          "Port": 993,
          "TLS": true}]}}
```


Response


```
{
  "AssertResponse": {}}
```


5.1. Enabling a new Device

Alice uses several different devices to read her email and she would like to be able to read encrypted emails on all of them. This requires that the private key be installed on each of the devices that she might want to use.

Alice provides either the Key Recovery Passcode or a sufficient number of Key Shares to reconstruct the passcode to the key management tool running on each device. The device then requests recovery of the private key and associated service information:

6. Recovery Example

Request


```
{
  "RecoverRequest": {
    "KeyIdentifier": "
AJqCYq5r2i7DXllBrhJHESWbLe2rw84cTsPr6qo"}}
```


Response


```
{
  "RecoverResponse": {}}
```


Providing the service information with the private key allows the key recovery tool to automate configuration of the user's email account on the device if this has not been done already.

Using the key recovery mechanism to support key transport between devices simplifies the initial coding task at the cost of a sub-optimal user experience for the user with a large number of devices in use and/or frequent key updates.

Future versions of the specification may adopt a different approach to key recovery in which each device in which keys are to be installed establishes a device specific keypair which is in turn used to automate the key transport. A key concern in the design of such a scheme being to prevent a weak random number generator on one device causing the private key to be compromised.

6.1. Revocation

Should the private key be lost, the subject be deceased or some other event occur that renders the key no longer servicable, a revocation statement is generated and issued. Such revocation statements use the Revoke request and the key endorsement message format:

7. Revocation Example

Request


```
{
  "RevokeRequest": {}}
```


Response


```
{
  "RevokeResponse": {}}
```


7.1. Key Endorsement

From time to time, Alice meets other PPE users and they endorse each other's keys. The AssertRequest is used to submit one or more signed key endorsements:

8. Endorsement Example

Request


```
{
  "AssertRequest": {
    "Endorsement": ["
MIH5MG8CAQAXDTEzMTAxNjA2MDYwM1oEHQCagmKua9ouw15ZQa4SRxElmy3tq8PO
HE7D6-qqBB0AGXoKrrHJ0-qMHfed6IOFC9Y_-V8bWp6Zmi_g3wUAMBkwFzAVMBMF
ABYPYm9iQGV4YW1wbGUuY29tBQAwAgUAA4GBAEPmx2IAjnlpR0z1V3K51HmjpYY3
dpJvsE0M41uAxPvhnnz-yCX5XYfa9MJzILag0eiVrVgTbE7CVH-ccRDgsr73sEri
LOc3vre32JWU2Cg1Y0s1sh1GMWJTj8DGPFLR-uOHCsFAxWK8XD6Y7hSlwZrh3EFu
SQfWoqzMtYkCY9wd"]}}
```


Response


```
{
  "AssertResponse": {}}
```


A key endorsement MAY be submitted to the Publication Interface by any party including the signer or the subject.

9. OmniAssertBroker

9.1. Assert

Register an assertion set.

The Assert transaction is used when a keypair is first created to register the new Key Identifier, Self Signed Certificate and Certificate Signing Request and to request revision of embedded attributes such as the email security policy.

The Assert transaction is also used to request registration of Key Endorsements.

9.1.1. Structure: Service

Email :
: String [0..1] Principal Email address associated with the account

OtherEmail :
: String [0..Many] Additional Email addresses associated with the account.

Name :
: String [0..1] DNS Address of Service

Protocol :
String [0..1] SRV format protocol identification prefix.

Port :
Integer [0..1] IP Port number

TLS :
Boolean [0..1] If true, use of TLS is required

Security :
String [0..1] Security policy description

9.1.2. Structure: EncryptedKey

EncryptedPrivateKey :
Binary [1..1] PKCS#8 Encrypted Private Key as specified in [!RFC5208].

ReleaseCode :
Binary [0..1] Release Code value for authorizing private key recovery requests. If specified the service MUST NOT release the encrypted private key unless the requestor satisfies a challenge-response request that establishes knowledge of the Release Code.

9.1.3. Message: AssertRequest

Register an assertion set

At present only a single Key Identifier may be registered per request and no provision is made to link related requests. This is likely to become necessary when different keys are being used for key endorsement, signature, encryption and master purposes.

KeyIdentifier :
Binary [1..1] Strong Key Identifier formed using a message digest function over the DER encoded Public Key Info block.

EncryptedKey :
EncryptedKey [0..1] Encrypted Private Key and associated attributes.

Certificate :
Binary [0..Many] PKIX Certificates to be registered, comply with [!RFC5280] and additional profile constraints specified here.

CertificateRequest :
Binary [0..Many] Certificate Request in [!RFC2986] format.

Endorsement :
Binary [0..Many] Key Endorsements as specified in this document.

Service :
Service [0..Many] Service connection information for associated services. For example, email IMAP [!RFC3501], POP3 [!RFC5034] and SUBMIT [!RFC4409] accounts.

9.1.4. Message: AssertResponse

Response to an assertion registration request.

It may be useful to expand the response to allow the gateway to provide information such as certificates issued in response to the certification request but these will typically require some form of validation and thus be returned asynchronously.

9.2. Recover

Recover a previously registered encrypted private key file from the service

If the Key Identifier cannot be found or there is no release code associated with the encrypted private key, the transaction is complete after the first response. Otherwise the service returns the status code 'ChallengeResponse' in response to the initial request and the client MUST make a second request in which it establishes proof of knowledge of the release code to complete the transaction.

9.2.1. Message: RecoverRequest

Request recovery of a previously registered encrypted private key.

KeyIdentifier :
Binary [1..1] Key Identifier of key pair for which recovery of the private key is being requested.

Challenge :
Binary [0..1] Client challenge value for proof of knowledge of the release code.

Answer :
Binary [0..1] Answer value for proof of knowledge of the release code.

9.2.2. Message: RecoverResponse

Respond to a recovery request.

If the encrypted private key associated with the specified Key Identifier has an associated

EncryptedPrivateKey :
: Binary [0..1] PKCS#8 Encrypted Private Key as specified in [!RFC5208].

Challenge :
: Binary [0..1] Server challenge value for proof of knowledge of the release code.

Algorithm :
: String [0..1] Digest algorithm for proof of knowledge of the release code.

9.3. Revoke

Publish a revocation meta-assertion

9.3.1. Message: RevokeRequest

Regquest revocation of a previously registered key and all related certificates and endorsements.

Note that whil key revocation necessarily entails revocation of all the certificates and endorsement associated with the key, the reverse is not the case. A user may revoke a certificate granting use of a key for encrypted email without wishing to revoke a certificate for the same key granting use for signed email.

KeyIdentifier :
: Binary [1..1] Key Identifier of Key to be revoked.

Notice :
: Binary [0..1] Signed Key Endorsement object with the 'revoke' attribute specified.

9.3.2. Message: RevokeResponse

Response to revocation request.

10. Security Considerations

I am sure there are some.

11. Acknowledgments

Thanks to the many people who have encouraged me in this work and in particular the members of the IETF PERPASS list and the Cryptography mailing list. Future versions of the draft will have a more complete list.

12. References

12.1. Normative References

[RFC5280] Cooper, D.,Santesson, S.,Farrell, S.,Boeyen, S.,Housley, R.,Polk, W., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC2986] Nystrom, M.,Kaliski, B., "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[RFC4409] Gellens, R.,Klensin, J., "Message Submission for Mail", RFC 4409, April 2006.

[RFC5034] Siemborski, R.,Menon-Sen, A., "The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism", RFC 5034, July 2007.

[I-D.hallambaker-prismproof-trust] Hallam-Baker, P, "PRISM Proof Trust Model", Internet-Draft draft-hallambaker-prismproof-trust-00, 16 October 2013.

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

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

[RFC5208] Kaliski, B., "Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2", RFC 5208, May 2008.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

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

12.2. Informative References

[I-D.moriarty-pkcs12v1-1] Moriarty, K,Nystrom, M,Parkinson, S,Rusch, A,Scott, M, "PKCS 12 v1: Personal Information Exchange Syntax", Internet-Draft draft-moriarty-pkcs12v1-1-01, 25 March 2013.

[RFC2616] Fielding, R.,Gettys, J.,Mogul, J.,Frystyk, H.,Masinter, L.,Leach, P.,Berners-Lee, T., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

philliph@comodo.com