openpgp
Internet-Draft
Intended status: Informational
Expires: October 8, 2019

D. Gillmor
ACLU

April 06, 2019

Abuse-Resistant OpenPGP Keystores
draft-dkg-openpgp-abuse-resistant-keystore-01

Abstract

OpenPGP transferable public keys are composite certificates, made up of primary keys, direct key signatures, user IDs, identity certifications ("signature packets"), subkeys, and so on. They are often assembled by merging multiple certificates that all share the same primary key, and are distributed in public keystores.

Unfortunately, since any third-party can add certifications with any content to any OpenPGP certificate, the assembled/merged form of a certificate can become unwieldy or undistributable.

This draft documents techniques that an archive of OpenPGP certificates can use to mitigate the impact of these various forms of flooding attacks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 8, 2019.

Table of Contents

1. Introduction

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

- "OpenPGP certificate" (or just "certificate") is used interchangeably with [RFC4880]'s "Transferable Public Key". The term "certificate" refers unambiguously to the entire composite object, unlike "key", which might also be used to refer to a primary key or subkey.

- An "identity certification" (or just "certification") is an [RFC4880] signature packet that covers OpenPGP identity information - that is, any signature packet of type 0x10, 0x11, 0x12, or 0x13. Certifications are said to (try to) "bind" a primary key to a User ID.

- The primary key that makes the certification is known as the "issuer". The primary key over which the certification is made is known as the "subject".

- A "first-party certification" is issued by the primary key of a certificate, and binds itself to a user ID in the certificate. That is, the issuer is the same as the subject. This is sometimes referred to as a "self-sig".

- A "third-party certification" is a made over a primary key and user ID by some other certification-capable primary key. That is, the issuer is different than the subject. (The elusive "second-party" is presumed to be the verifier who is trying to interpret the certificate)

- A "keystore" is any collection of OpenPGP certificates. Keystores typically receive mergeable updates over the course of their lifetime which might add to the set of OpenPGP certificates they hold, or update the certificates.

- "Certificate discovery" is the process whereby a user retrieves an OpenPGP certificate based on user ID (see Section 10.2). A user attempting to discover a certificate from a keystore will search for a substring of the known user IDs, most typically an e-mail address if the user ID is an [RFC5322] name-addr or addr-spec. Some certificate discovery mechanisms look for an exact match on the known user IDs. [I-D.koch-openpgp-webkey-service] and [I-D.shaw-openpgp-hkp] both offer certificate discovery mechanisms.

- "Certificate validation" is the process whereby a user decides whether a given user ID in an OpenPGP certificate is acceptable. For example, if the certificate has a user ID of "Alice <alice@example.org>" and the user wants to send an e-mail to "alice@example.org", the mail user agent might want to ensure that the certificate is valid for this e-mail address before encrypting to it. This process can take different forms, and can consider many different factors, some of which are not directly contained in the certificate itself. For example, certificate validation might consider whether the certificate was fetched via DANE ([RFC7929]) or WKD ([I-D.koch-openpgp-webkey-service]); or whether it has seen e-mails from that address signed by the certificate in the past; or how long it has known about certificate.

- "Certificate update" is the process whereby a user fetches new information about a certificate, potentially merging those OpnePGP packets to change the status of the certificate. Updates might include adding or revoking user IDs or subkeys, updating

expiration dates, or even revoking the entire certificate by revoking the primary key directly. A user attempting to update a certificate typically queries a keystore based on the certificate's fingerprint.

- A "keyserver" is a particular kind of keystore, typically means of publicly distributing OpenPGP certificates or updates to them. Examples of keyserver software include [SKS] and [MAILVELOPE-KEYSERVER]. One common HTTP interface for keyservers is [I-D.shaw-openpgp-hkp].

- A "synchronizing keyserver" is a keyserver which gossips with other peers, and typically acts as an append-only log. Such a keyserver is typically useful for certificate discovery, certificate updates, and revocation information. They are typically _not_ useful for certificate validation, since they make no assertions about whether the identities in the certificates they server are accurate. As of the writing of this document, [SKS] is the canonical synchronizing keyserver implementation, though other implementations exist.

- An "e-mail-validating keyserver" is a keyserver which attempts to verify the identity in an OpenPGP certificate's user ID by confirming access to the e-mail account, and possibly by confirming access to the secret key. Some implementations permit removal of a certificate by anyone who can prove access to the e-mail address in question. They are useful for certificate discovery based on e-mail address and certificate validation (by users who trust the operator), but some may not be useful for certificate update or revocation, since a certificate could be simply replaced by an adversary who also has access to the e-mail address in question. [MAILVELOPE-KEYSERVER] is an example of such a keyserver.

- "Cryptographic validity" refers to mathematical evidence that a signature came from the secret key associated with the public key it claims to come from. Note that a certification may be cryptographically valid without the signed data being true (for example, a given certificate with the user ID "Alice <alice@example.org>" might not belong to the person who controls the e-mail address "alice@example.org" even though the self-sig is cryptographically valid). In particular, cryptographic validity for user ID in a certificate is typically insufficient evidence for certificate validation. Also note that knowledge of the public key of the issuer is necessary to determine whether any given signature is cryptographically valid. Some keyservers perform cryptographic validation in some contexts. Other

keyservers (like [SKS]) perform no cryptographic validation whatsoever.

- OpenPGP revocations can have "Reason for Revocation" (see [RFC4880]), which can be either "soft" or "hard". The set of "soft" reasons is: "Key is superseded" and "Key is retired and no longer used". All other reasons (and revocations that do not state a reason) are "hard" revocations. See Section 10.1 for more detail.

2. Problem Statement

OpenPGP keystores that handle submissions from the public are subject to a range of flooding attacks by malicious submitters.

This section describes three distinct flooding attacks that public keystores should consider.

The rest of the document describes some mitigations that can be used by keystores that are concerned about these problems but want to continue to offer some level of service for certificate discovery, certificate update, or certificate validation.

2.1. Certificate Flooding

Many public keystores (including both the [SKS] keyserver network and [MAILVELOPE-KEYSERVER]) allow anyone to attach arbitrary data (in the form of third-party certifications) to any certificate, bloating that certificate to the point of being impossible to effectively retrieve. For example, some OpenPGP implementations simply refuse to process certificates larger than a certain size.

This kind of Denial-of-Service attack makes it possible to make someone else's certificate unretrievable from the keystore, preventing certificate discovery. It also makes it possible to swamp a certificate that has been revoked, preventing certificate update, potentially leaving the client of the keystore with the compromised certificate in an unrevoked state locally.

Additionally, even without malice, OpenPGP certificates can potentially grow without bound.

2.2. User ID Flooding

Public keystores that are used for certificate discovery may also be vulnerable to attacks that flood the space of known user IDs. In particular, if the keystore accepts arbitrary certificates from the public and does no verification of the user IDs, then any client

searching for a given user ID may need to review and process an effectively unbounded set of maliciously-submitted certificates to find the non-malicious certificates they are looking for.

For example, if an attacker knows that a given system consults a keystore looking for certificates which match the e-mail address "alice@example.org", the attacker may upload hundreds or thousands of certificates containing user IDs that match that address. Even if those certificates would not be accepted by a client (e.g., because they were not certified by a known-good authority), the client typically still has to wade through all of them in order to find the non-malicious certificates.

If the keystore does not offer a discovery interface at all (that is, if clients cannot search it by user ID), then user ID flooding is of less consequence.

2.3. Keystore Flooding

A public keystore that accepts arbitrary OpenPGP material and is append-only is at risk of being overwhelmed by sheer quantity of malicious uploaded packets. This is a risk even if the user ID space is not being deliberately flooded, and if individual certificates are protected from flooding by any of the mechanisms described later in this document.

The keystore itself can become difficult to operate if the total quantity of data is too large, and if it is a synchronizing keyserver, then the quantities of data may impose unsustainable bandwidth costs on the operator as well.

Effectively mitigating against keystore flooding requires either abandoning the append-only property that some keystores prefer, or imposing very strict controls on initial ingestion.

3. Simple Mitigations

These steps can be taken by any keystore that wants to avoid obviously malicious abuse. They can be implemented on receipt of any new packet, and are based strictly on the structure of the packet itself.

3.1. Decline Large Packets

While [RFC4880] permits OpenPGP packet sizes of arbitrary length, OpenPGP certificates rarely need to be so large. An abuse-resistant keystore SHOULD reject any OpenPGP packet larger than 8383 octets. (This cutoff is chosen because it guarantees that the packet size can

be represented as a one- or two-octet [RFC4880] "New Format Packet Length", but it could be reduced further)

This may cause problems for user attribute packets that contain large images, but it's not clear that these images are concretely useful in any context. Some keystores MAY extend this limit for user attribute packets specifically, but SHOULD NOT allow even user attributes packets larger than 65536 octets.

3.2. Enforce Strict User IDs

[RFC4880] indicates that User IDs are expected to be UTF-8 strings. An abuse-resistant keystore MUST reject any user ID that is not valid UTF-8.

Some abuse-resistant keystores MAY only accept User IDs that meet even stricter conventions, such as an [RFC5322] name-addr or addr-spec, or a URL like "ssh://host.example.org".

As simple text strings, User IDs don't need to be nearly as long as any other packets. An abuse-resistant keystore SHOULD reject any user ID packet larger than 1024 octets.

3.3. Scoped User IDs

Some abuse-resistant keystores may restrict themselves to publishing only certificates with User IDs that match a specific pattern. For example, [RFC7929] encourages publication in the DNS of only certificates whose user IDs refer to e-mail addresses within the DNS zone. [I-D.koch-openpgp-webkey-service] similarly aims to restrict publication to certificates relevant to the specific e-mail domain.

3.4. Strip or Standardize Unhashed Subpackets

[RFC4880] signature packets contain an "unhashed" block of subpackets. These subpackets are not covered by any cryptographic signature, so they are ripe for abuse.

An abuse-resistant keysetore SHOULD strip out all unhashed subpackets.

Note that some certifications only identify the issuer of the certification by an unhashed Issuer ID subpacket. If a certification's hashed subpacket section has no Issuer ID or Issuer Fingerprint (see [I-D.ietf-openpgp-rfc4880bis]) subpacket, then an abuse-resistant keystore that has cryptographically validated the certification SHOULD make the unhashed subpackets contain only a

single subpacket. That subpacket should be of type Issuer Fingerprint, and should contain the fingerprint of the issuer.

A special exception may be made for unhashed subpackets in a third-party certification that contain attestations from the certificate's primary key as described in Section 8.

3.5. Decline User Attributes

Due to size concerns, some abuse-resistant keystores MAY choose to ignore user attribute packets entirely, as well as any certifications that cover them.

3.6. Decline Non-exportable Certifications

An abuse-resistant keystore MUST NOT accept any certification that has the "Exportable Certification" subpacket present and set to 0. While most keystore clients will not upload these "local" certifications anyway, a reasonable public keystore that wants to minimize data has no business storing or distributing these certifications.

3.7. Decline Data From the Future

Many OpenPGP packets have time-of-creation timestamps in them. An abuse-resistant keystore with a functional real-time clock MAY decide to only accept packets whose time-of-creation is in the future.

Note that some OpenPGP implementations may pre-generate OpenPGP material intended for use only in some future window (e.g. "Here is the certificate we plan to use to sign our software next year; do not accept signatures from it until then."), and may use modified time-of-creation timestamps to try to acheive that purpose. This material would not be distributable ahead of time by an abuse-resistant keystore that adopts this mitigation.

3.8. Accept Only Profiled Certifications

An aggressively abuse-resistant keystore MAY decide to only accept certifications that meet a specific profile. For example, it MAY reject certifications with unknown subpacket types, unknown notations, or certain combinations of subpackets. This can help to minimize the amount of room for garbage data uploads.

Any abuse-resistant keystore that adopts such a strict posture should clearly document what its expected certificate profile is, and should have a plan for how to extend the profile if new types of certification appear that it wants to be able to distribute.

Note that if the profile is ever restricted (rather than extended), and the restriction is applied to the material already present, such a keystore is no longer append-only (please see Section 5).

3.9. Accept Only Certificates Issued by Designated Authorities

An abuse-resistant keystore capable of cryptographic validation MAY retain a list of designated authorities, typically in the form of a set of known public keys. Upon receipt of a new OpenPGP certificate, the keystore can decide whether to accept or decline each user ID of the certificate based whether that user ID has a certification that was issued by one or more of the designated authorities.

If no user IDs are certified by designated authority, such a keystore SHOULD decline the certificate and its primary key entirely. Such a keystore SHOULD decline to retain or propagate all certifications associated with each accepted user ID except for first-party certifications and certifications by the designated authorities.

The operator of such a keystore SHOULD have a clear policy about its set of designated authorities.

Given the ambiguities about expiration and revocation, such a keyserver SHOULD ignore expiration and revocation of authority certifications, and simply accept and retain as long as the cryptographic signature is valid.

Note that if any key is removed from the set of designated authorities, and that change is applied to the existing keystore, such a keystore may no longer be append-only (please see Section 5).

3.10. Decline Packets by Blocklist

The maintainer of the keystore may keep a specific list of "known-bad" material, and decline to accept or redistribute items matching that blocklist. The material so identified could be anything, but most usefully, specific public keys or User IDs could be blocked.

Note that if a blocklist grows to include an element already present in the keystore, it will no longer be append-only (please see Section 5).

Some keystores may choose to apply a blocklist only at distribution time and not apply it at input time. This allows the keystore to be append-only, and permits synchronization between keystores that don't share a blocklist, and somewhat reduces the attacker's incentive for flooding the keystore.

Note that development and maintenance of a blocklist is not without its own potentials for abuse. For one thing, the blocklist may itself grow without bound. Additionally, a blocklist may be socially or politically contentious. There needs to be a clear policy about how it is managed, whether by delegation to specific decision-makers, or explicit tests. Furthermore, the existence of even a well-intentioned blocklist may be an "attractive nuisance," drawing the interest of would-be censors or other attacker interested in controlling the ecosystem reliant on the keystore in question.

4. Contextual Mitigations

Some mitigations make the acceptance or rejection of packets contingent on data that is already in the keystore or the keystore's developing knowledge about the world. This means that, depending on the order that the keystore encounters the various material, or how it discovers the material, the final set of material retained and distributed by the keystore might be different.

While this isn't necessarily bad, it may be a surprising property for some users of keystores.

4.1. Accept Only Cryptographically-verifiable Certifications

An abuse-resistant keystore that is capable of doing cryptographic validation MAY decide to reject certifications that it cannot cryptographically validate.

This may mean that the keystore rejects some packets while it is unaware of the public key of the issuer of the packet.

4.2. Accept Only Certificates Issued by Known Certificates

This is an extension of Section 3.9, but where the set of authorities is just the set of certificates already known to the keystore. An abuse-resistant keystore that adopts this strategy is effectively only crawling the reachable graph of OpenPGP certificates from some starting core.

A keystore adopting the mitigation SHOULD have a clear documentation of the core of initial certificates it starts with, as this is effectively a policy decision.

This mitigation measure may fail due to a compromise of any secret key that is associated with a primary key of a certificate already present in the keystore. Such a compromise permits an attacker to flood the rest of the network. In the event that such a compromised key is identified, it might be placed on a blocklist (see

Section 3.10). In particular, if a public key is added to a blocklist for a keystore implementing this mitigation, and it is removed from the keystore, then all certificates that were only "reachable" from the blocklisted certificate should also be simultaneously removed.

4.3. Rate-limit Submissions by IP Address

Some OpenPGP keystores accept material from the general public over the Internet. If an abuse-resistant keystore observes a flood of material submitted to the keystore from a given Internet address, it MAY choose to throttle submissions from that address. When receiving submissions over IPv6, such a keystore MAY choose to throttle entire nearby subnets, as a malicious IPv6 host is more likely to have multiple addresses.

This requires that the keystore maintain state about recent submissions over time and address. It may also be problematic for users who appear to share an IP address from the vantage of the keystore, including those beind a NAT, using a VPN, or accessing the keystore via Tor.

4.4. Accept Certiifcates Based on Exterior Process

Some public keystores resist abuse by explicitly filtering OpenPGP material based on a set of external processes. For example, [DEBIAN-KEYRING] adjudicates the contents of the "Debian keyring" keystore based on organizational procedure and manual inspection.

4.5. Accept Certificates by E-mail Validation

Some keystores resist abuse by declining any certificate until the user IDs have been verified by e-mail. When these "e-mail-validating" keystores review a new certificate that has a user ID with an e-mail address in it, they send an e-mail to the associated address with a confirmation mechanism (e.g., a high-entropy HTTPS URL link) in it. In some cases, the e-mail itself is encrypted to an encryption-capable key found in the proposed certificate. If the keyholder triggers the confirmation mechanism, then the keystore accepts the certificate.

[PGP-GLOBAL-DIRECTORY] describes some concerns held by a keystore operator using this approach. [MAILVELOPE-KEYSERVER] is another example.

5. Non-append-only mitigations

The following mitigations may cause some previously-retained packets to be dropped after the keystore receives new information, or as time passes. This is entirely reasonable for some keystores, but it may be surprising for any keystore that expects to be append-only (for example, some keyserver synchronization techniques may expect this property to hold).

Furthermore, keystores that drop old data, or certifications that are superseded may make it difficult or impossible for their users to reason about the validity of signatures that were made in the past. See Section 9.3 for more considerations.

Note also that many of these mitigations depend on cryptographic validation, so they're typically contextual as well.

A keystore that needs to be append-only, or which cannot perform cryptographic validation MAY omit these mitigations.

Note that [GnuPG] anticipates some of these suggestions with its "clean" subcommand, which is documented as:

> Compact (by removing all signatures except the selfsig) any user ID that is no longer usable (e.g. revoked, or expired). Then, remove any signatures that are not usable by the trust calculations. Specifically, this removes any signature that does not validate, any signature that is superseded by a later signature, revoked signatures, and signatures issued by keys that are not present on the keyring.

5.1. Drop Superseded Signatures

An abuse-resistant keystore SHOULD drop all signature packets that are explicitly superseded. For example, there's no reason to retain or distribute a self-sig by key K over User ID U from 2017 if the keystore have a cryptographically-valid self-sig over <K,U> from 2019.

Note that this covers both certifications and signatures over subkeys, as both of these kinds of signature packets may be superseded.

Getting this right requires a nuanced understanding of subtleties in [RFC4880] related to timing and revocation.

5.2. Drop Expired Signatures

If a signature packet is known to only be valid in the past, there is no reason to distribute it further. An abuse-resistant keystore with access to a functionally real-time clock SHOULD drop all certifications and subkey signature packets with an expiration date in the past.

Note that this assumes that the keystore and its clients all have roughly-synchronized clocks. If that is not the case, then there will be many other problems!

5.3. Drop Dangling User IDs, User Attributes, and Subkeys

If enough signature packets are dropped, it's possible that some of the things that those signature packets cover are no longer valid.

An abuse-resistant keystore which has dropped all certifications that cover a User ID SHOULD also drop the User ID packet.

Note that a User ID that becomes invalid due to revocation MUST NOT be dropped, because the User ID's revocation signature itself remains valid, and needs to be distributed.

A primary key with no User IDs and no subkeys and no revocations MAY itself also be removed from distribution, though note that the removal of a primary key may make it impossible to cryptographically validate other certifications held by the keystore.

5.4. Drop All Other Elements of a Directly-Revoked Certificate

If the primary key of a certiifcate is revoked via a direct key signature, an abuse-resistant keystore SHOULD drop all the rest of the associated data (user IDs, user attributes, and subkeys, and all attendant certifications and subkey signatures). This defends against an adversary who compromises a primary key and tries to flood the certificate to hide the revocation.

Note that the direct key revocation signature MUST NOT be dropped.

In the event that an abuse-resistant keystore is flooded with direct key revocation signatures, it should retain the hardest, earliest revocation (see also Section 10.1).

In particular, if any of the direct key revocation signatures is a "hard" revocation, the abuse-resistant keystore SHOULD retain the earliest such revocation signature (by signature creation date).

Otherwise, the abuse-resistant keystore SHOULD retain the earliest "soft" direct key revocation signature it has seen.

If either of the above date comparisons results in a tie between two revocation signatures of the same "hardness", an abuse-resistant keystore SHOULD retain the signature that sorts earliest based on a binary string comparison of the direct key revocation signature packet itself.

5.5. Implicit Expiration Date

In combination with some of the dropping mitigations above, a particularly aggressive abuse-resistant keystore MAY choose an implicit expiration date for all signature packets. For example, a signature packet that claims no expiration could be treated by the keystore as expiring 3 years after issuance. This would permit the keystore to eject old packets on a rolling basis.

FIXME: it's not clear what should happen with signature packets marked with an explicit expiration that is longer than implicit maximum. Should it be capped to the implicit date, or accepted?

Warning: This idea is pretty radical, and it's not clear what it would do to an ecosystem that depends on such a keystore. It probably needs more thinking.

6. Updates-only Keystores

In addition to the mitigations above, some keystores may resist abuse by declining to accept any user IDs or certifications whatsoever.

Such a keystore MUST be capable of cryptographic validation. It accepts primary key packets, cryptographically-valid direct-key signatures from a primary key over itself, subkeys and their cryptographically-validated binding signatures (and cross signatures, where necessary).

Clients of an updates-only keystore cannot possibly use the keystore for certificate discovery, because there are no user IDs to match. However, they can use it for certificate update, as it's possible to ship revocations (which are direct key signatures), new subkeys, updates to subkey expiration, subkey revocation, and direct key signature-based certificate expiration updates.

Note that many popular OpenPGP implemenations do not implement direct primary key expiration mechanisms, relying instead on user ID expirations. These user ID expiration dates or other metadata

associated with a self-certification will not be distributed by an updates-only keystore.

Certificates shipped by an updates-only keystore are technically invalid [RFC4880] "transferable public keys," because they lack a user ID packet. However many OpenPGP implementations will accept such a certificate if they already know of a user ID for the certificate, because the composite certificate resulting from a merge will be a standards-compliant transferable public key.

7. First-party-only Keystores

Slightly more permissive than the updates-only keystore described in Section 6 is a keystore that also permits user IDs and their self-sigs.

A first-party-only keystore only accepts and distributes cryptographically-valid first-party certifications. Given a primary key that the keystore understands, it will only attach user IDs that have a valid self-sig, and will only accept and re-distribute subkeys that are also cryptographically valid (including requiring cross-sigs for signing-capable subkeys as recommended in [RFC4880]).

This effectively solves the problem of abusive bloating attacks on any certificate, because the only party who can make a certificate overly large is the holder of the secret corresponding to the primary key itself.

However, a first-party-only keystore is still problematic for those people who rely on the keystore for discovery of third-party certifications. Section 8 attempts to address this lack.

8. First-party-attested Third-party Certifications

We can augment a first-party-only keystore to allow it to distribute third-party certifications as long as the first-party has signed off on the specific third-party certification.

An abuse-resistant keystore SHOULD only accept a third-party certification if it meets the following criteria:

- The third-party certification MUST be cryptographically valid. Note that this means that the keystore needs to know the primary key for the issuer of the third-party certification.

- The third-party certification MUST have an unhashed subpacket of type Embedded Signature, the contents of which we'll call the "attestation". This attestation is from the certificate's primary

key over the third-party certification itself, as detailed in the steps below:

- The attestation MUST be an OpenPGP signature packet of type 0x50 (Third-Party Confirmation signature)
- The attestation MUST contain a hashed "Issuer Fingerprint" subpacket with the fingerprint of the primary key of the certificate in question.
- The attestation MUST NOT be marked as non-exportable.
- The attestation MUST contain a hashed Notation subpacket with the name "ksok", and an empty (0-octet) value.
- The attestation MUST contain a hashed "Signature Target" subpacket with "public-key algorithm" that matches the public-key algorithm of the third-party certification.
- The attestation's hashed "Signature Target" subpacket MUST use a reasonably strong hash algorithm (as of this writing, any [RFC4880] hash algorithm except MD5, SHA1, or RIPEMD160), and MUST have a hash value equal to the hash over the third-party certification with all unhashed subpackets removed.
- The attestation MUST be cryptographically valid, verifiable by the primary key of the certificate in question.

What this means is that a third-party certificate will only be accepted/distributed by the keystore if:

- the keystore knows about both the first- and third-parties.
- the third-party has made the identity assertion
- the first-party has confirmed that they're OK with the third-party certification being distributed by any keystore.

FIXME: it's not clear whether the "ksok" notification is necessary - it's in place to avoid some accidental confusion with any other use of the Third-Party Confirmation signature packet type, but the author does not know of any such use that might collide.

8.1. Key Server Preferences "No-modify"

[RFC4880] defines "Key Server Preferences" with a "No-modify" bit. That bit has never been respected by any keyserver implementation that the author is aware of. This section effectively asks an abuse-

resistant keystore to treat that bit as always set, whether it is present in the certificate or not.

8.2. Client Interactions

The multi-stage layer of creating such an attestation (certificate creation by the first-party, certification by the third-party, attestation by the first-party, then handoff to the keystore) may represent a usability obstacle to a user who needs a third-party-certified OpenPGP certificate.

No current OpenPGP client can easily create the attestions described in this section. More implementation work needs to be done to make it easy (and understandable) for a user to perform this kind of attestation.

9. Side Effects and Ecosystem Impacts

9.1. Designated Revoker

A first-party-only keystore as described in Section 7 might decline to distribute revocations made by the designated revoker. This is a risk to certificate-holder who depend on this mechanism, because an important revocation might be missed by clients depending on the keystore.

FIXME: adjust this document to point out where revocations from a designated revoker SHOULD be propagated, maybe even in first-party-only keystores.

9.2. Certification-capable Subkeys

Much of this discussion assumes that primary keys are the only certification-capable keys in the OpenPGP ecosystem. Some proposals have been put forward that assume that subkeys can be marked as certification-capable. If subkeys are certification-capable, then much of the reasoning in this draft becomes much more complex, as subkeys themselves can be revoked by their primary key without invalidating the key material itself. That is, a subkey can be both valid (in one context) and invalid (in another context) at the same time. So questions about what data can be dropped (e.g. in Section 5) are much fuzzier, and the underlying assumptions may need to be reviewed.

If some OpenPGP implementations accept certification-capable subkeys, but an abuse-resistant keystore does not accept certifications from subkeys in general, then interactions between that keystore and those implementations may be surprising.

9.3. Assessing Certificates in the Past

Online protocols like TLS perform signature and certificate evaluation based entirely on the present time. If a certificate that signs a TLS handshake message is invalid now, it doesn't matter whether it was valid a week ago, because the present TLS session is the context of the evaluation.

But OpenPGP signatures are often evaluated at some temporal remove from when the signature was made. For example, software packages are signed at release time, but those signatures are validated at download time.

Further complicating matters, the composable nature of an OpenPGP certificate means that the certificate associated with any particular signing key (primary key or subkey) can transform over time. So when evaluating a signature that appears to have been made by a given certificate, it may be better to try to evaluate the certificate at the time the signature was made, rather than the present time.

When evaluating a certificate at a time T in the past, one approach is to discard all packets with a creation time later than T, and then evaluate the resulting certificate from the remaining packets in the context of time T.

However, any such evaluator SHOULD NOT ignore "hard" OpenPGP key revocations, regardless of their creation date. (see Section 10.1).

If a non-append-only keystore (Section 5) has dropped superseded (Section 5.1) or expired (Section 5.2) certifications, it's possible for the certificate composed of the remaining packets to have no valid first-party certification at the time that a given signature was made. Such a certificate would be invalid according to [RFC4880].

10. OpenPGP details

This section collects details about common OpenPGP implementation behavior that are useful in evaluating and reasoning about OpenPGP certificates.

10.1. Revocations

It's useful to classify OpenPGP revocations of key material into two categories: "soft" and "hard".

If the "Reason for Revocation" of an OpenPGP key is either "Key is superseded" or "Key is retired and no longer used", it is a "soft" revocation.

An implementation that interprets a "soft" revocation will typically not invalidate signatures made by the associated key with a creation date that predates the date of the soft revocation. A "soft" revocation in some ways behaves like a non-overridable expiration date.

All other revocations of OpenPGP keys (with any other Reason for Revocation, or with no Reason for Revocation at all) should be considered "hard".

The presence of a "hard" revocation of an OpenPGP key indicates that the user should reject all signatures and certifications made by that key, regardless of the creation date of the signature.

Note that some OpenPGP implementations do not distinguish between these two categories.

A defensive OpenPGP implementation that does not distinguish between these two categories SHOULD treat all revocations as "hard".

An implementation aware of a "soft" revocation or of key or certificate expiry at time T SHOULD accept and process a "hard" revocation even if it appears to have been issued at a time later than T.

10.2. User ID Conventions

[RFC4880] requires a user ID to be a UTF-8 string, but does not constrain it beyond that. In practice, a handful of conventions predominate in how User IDs are formed.

The most widespread convention is a name-addr as defined in [RFC5322]. For example:

Alice Jones <alice@example.org>

But a growing number of OpenPGP certificates contain user IDs that are instead a raw [RFC5322] addr-spec, omitting the display-name and the angle brackets entirely, like so:

alice@example.org

Some certificates have user IDs that are simply "normal" human names (perhaps display-name in [RFC5322] jargon, though not necessarily conforming to a specific ABNF). For example:

Alice Jones

Still other certificates identify a particular network service by scheme and hostname. For example, the administrator of an ssh host participating in the [MONKEYSPHERE] might choose a user ID for the OpenPGP representing the host like so:

ssh://foo.example.net

11. Security Considerations

This document offers guidance on mitigating a range of denial-of-service attacks on public keystores, so the entire document is in effect about security considerations.

Many of the mitigations described here defend individual OpenPGP certificates against flooding attacks (see Section 2.1). But only some of these mitigations defend against flooding attacks against the keystore itself (see Section 2.3), or against flooding attacks on the space of possible user IDs (see Section 2.2). Thoughtful threat modeling and monitoring of the keystore and its defenses are probably necessary to maintain the long-term health of the keystore.

Section 9.1 describes a potentially scary security problem for designated revokers.

Note that there is an inherent tension between accepting arbitrary certificate uploads and permitting effective certificate discovery. If a keystore accepts arbitrary certificate uploads for redistribution, it appears to be vulnerable to user ID flooding (Section 2.2), which makes it difficult or impossible to rely on for certificate discovery.

TODO (more security considerations)

12. Privacy Considerations

Keystores themselves raise a host of potential privacy concerns. Additional privacy concerns are raised by traffic to and from the keystores. This section tries to outline some of the risks to the privacy of people whose certificates are stored and redistributed in public keystores, as well as risks to the privacy of people who make use of the key stores for certificate discovery or certificate update.

TODO (more privacy considerations)

12.1. Publishing Identity Information

Public OpenPGP keystores often distribute names or e-mail addresses of people. Some people do not want their names or e-mail addresses distributed in a public keystore, or may change their minds about it at some point. Append-only keystores are particularly problematic in that regard. The mitigation in Section 5.4 can help such users strip their details from keys that they control. However, if an OpenPGP certificate with their details is uploaded to a keystore, but is not under their control, it's unclear what mechanisms can be used to remove the certificate that couldn't also be exploited to take down an otherwise valid certificate.

An updates-only keyserver (Section 6) avoids this particular privacy concern because it distributes no user IDs at all.

12.2. Social Graph

Third-party certifications effectively map out some sort of social graph. A certification asserts a statement of belief by the issuer that the real-world party identified by the user ID is in control of the subject cryptographic key material. But those connections may be potentially sensitive, and some people may not want these maps built.

A first-party-only keyserver (Section 7) avoids this privacy concern because it distribues no third-party privacy concern.

First-party attested third-party certifications described in Section 8 are even more relevant edges in the social graph, because their bidirectional nature suggests that both parties are aware of each other, and see some value in mutual association.

12.3. Tracking Clients by Queries

Even without third-party certifications, the acts of certificate discovery and certificate update represent a potential privacy risk, because the keystore queried gets to learn which user IDs (in the case of discovery) or which certificates (in the case of update) the client is interested in. In the case of certificate update, if a client attempts to update all of its known certificates from the same keystore, that set is likely to be a unique set, and therefore identifies the client. A keystore that monitors the set of queries it receives might be able to profile or track those clients who use it repeatedly.

Clients which want to to avoid such a tracking attack MAY try to perform certificate update from multiple different keystores. To hide network location, a client making a network query to a keystore SHOULD use an anonymity network like [TOR]. Tools like [PARCIMONIE] are designed to facilitate this type of certificate update.

Keystores which permit public access and want to protect the privacy of their clients SHOULD NOT reject access from clients using [TOR] or comparable anonymity networks. Additionally, they SHOULD minimize access logs they retain.

Alternately, some keystores may distribute their entire contents to any interested client, in what can be seen as the most trivial form of private information retrieval. [DEBIAN-KEYRING] is one such example; its contents are distributed as an operating system package. Clients can interrogate their local copy of such a keystore without exposing their queries to a third-party.

12.4. Cleartext Queries

If access to the keystore happens over observable channels (e.g., cleartext connections over the Internet), then a passive network monitor could perform the same type profiling or tracking attack against clients of the keystore described in Section 12.3. Keystores which offer network access SHOULD provide encrypted transport.

12.5. Traffic Analysis

Even if a keystore offers encrypted transport, the size of queries and responses may provide effective identification of the specific certificates fetched during discovery or update, leaving open the types of tracking attacks described in Section 12.3. Clients of keystores SHOULD pad their queries to increase the size of the anonymity set. And keystores SHOULD pad their responses.

The appropriate size of padding to effectively anonymize traffic to and from keystores is likely to be mechanism- and cohort-specific. For example, padding for keystores accessed via the DNS ([RFC7929] may use different padding strategies that padding for keystores accessed over WKD ([I-D.koch-openpgp-webkey-service]), which may in turn be different from keystores accessed over HKPS ([I-D.shaw-openpgp-hkp]). A keystore which only accepts user IDs within a specific domain (e.g., Section 3.3) or which uses custom process (Section 4.4) for verification might have different padding criteria than a keystore that serves the general public.

Specific padding policies or mechanisms are out of scope for this document.

13. User Considerations

Section 8.2 describes some outstanding work that needs to be done to help users understand how to produce and distribute a third-party-certified OpenPGP certificate to an abuse-resistant keystore.

14. IANA Considerations

This document asks IANA to register the "ksok" notation name in the OpenPGP Notation IETF namespace, with a reference to this document, as defined in Section 8.

15. Document Considerations

[RFC Editor: please remove this section before publication]

This document is currently edited as markdown. Minor editorial changes can be suggested via merge requests at https://gitlab.com/dkg/draft-openpgp-abuse-resistant-keystore or by e-mail to the author. Please direct all significant commentary to the public IETF OpenPGP mailing list: openpgp@ietf.org

15.1. Document History

substantive changes between -00 and -01:

- split out Contextual and Non-Append-Only mitigations
- documented several other mitigations, including:
  - Decline Data From the Future
  - Blocklist
  - Exterior Process
  - Designated Authorities
  - Known Certificates
  - Rate-Limiting
  - Scoped User IDs
- documented Updates-Only Keystores
- consider three different kinds of flooding

- deeper discussion of privacy considerations
- better documentation of Reason for Revocation
- document user ID conventions

16. Acknowledgements

This document is the result of years of operational experience and observation, as well as conversations with many different people - users, implementors, keystore operators, etc. A non-exhaustive list of people who have contriubuted ideas or nuance to this document specifically includes:

- Antoine Beaupre
- Jamie McClelland
- Jonathan McDowell
- Justus Winter
- Neal Walfield
- vedaal
- Vincent Breitmoser
- Wiktor Kwapisiewicz

17. References

17.1. Normative References

[I-D.ietf-openpgp-rfc4880bis]
Koch, W., carlson, b., Tse, R., and D. Atkins, "OpenPGP Message Format", draft-ietf-openpgp-rfc4880bis-06 (work in progress), November 2018.

[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/info/rfc2119>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

17.2. Informative References

[DEBIAN-KEYRING]
McDowell, J., "Debian Keyring", n.d., <https://keyring.debian.org/>.

[GnuPG] Koch, W., "Using the GNU Privacy Guard", n.d., <https://www.gnupg.org/documentation/manuals/gnupg.pdf>.

[I-D.koch-openpgp-webkey-service]
Koch, W., "OpenPGP Web Key Directory", draft-koch-openpgp-webkey-service-07 (work in progress), November 2018.

[I-D.shaw-openpgp-hkp]
Shaw, D., "The OpenPGP HTTP Keyserver Protocol (HKP)", draft-shaw-openpgp-hkp-00 (work in progress), March 2003.

[MAILVELOPE-KEYSERVER]
Oberndoerfer, T., "Mailvelope Keyserver", n.d., <https://github.com/mailvelope/keyserver/>.

[MONKEYSPHERE]
Gillmor, D. and J. Rollins, "Monkeysphere", n.d., <https://web.monkeysphere.info/>.

[PARCIMONIE]
Intrigeri, ., "Parcimonie", n.d., <https://gaffer.ptitcanardnoir.org/intrigeri/code/parcimonie/>.

[PGP-GLOBAL-DIRECTORY]
Symantec Corporation, "PGP Global Directory Key Verification Policy", 2011, <https://keyserver.pgp.com/vkd/VKDVerificationPGPCom.html>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC7929] Wouters, P., "DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP", RFC 7929, DOI 10.17487/RFC7929, August 2016, <https://www.rfc-editor.org/info/rfc7929>.

[SKS] Pennock, P., "SKS Keyserver Documentation", March 2018, <https://bitbucket.org/skskeyserver/sks-keyserver/wiki/Home>.

[TOR] "The Tor Project", n.d., <https://www.torproject.org/>.

Author's Address

Daniel Kahn Gillmor
American Civil Liberties Union
125 Broad St.
New York, NY 10004
USA

Email: dkg@fifthhorseman.net