

openpgp  
Internet-Draft  
Updates: 9580 (if approved)  
Intended status: Informational  
Expires: 20 September 2025

D. K. Gillmor  
ACLU  
A. Gallagher  
19 March 2025

Revocation in OpenPGP  
draft-dkg-openpgp-revocation-02

## Abstract

Cryptographic revocation is a hard problem. OpenPGP's revocation mechanisms are imperfect, not fully understood, and not as widely implemented as they could be. Additionally, some historical OpenPGP revocation mechanisms simply do not work in certain contexts. This document provides clarifying guidance on how OpenPGP revocation works, documents outstanding problems, and introduces a new mechanism for delegated revocations that improves on previous mechanism.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://dkg.gitlab.io/openpgp-revocation/>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-dkg-openpgp-revocation/>.

Discussion of this document takes place on the OpenPGP Working Group mailing list (<mailto:openpgp@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/openpgp/>. Subscribe at <https://www.ietf.org/mailman/listinfo/openpgp/>.

Source for this draft and an issue tracker can be found at <https://gitlab.com/dkg/openpgp-revocation>.

## 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 20 September 2025.

## Copyright Notice

Copyright (c) 2025 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 (<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	4
1.1. Terminology . . . . .	4
2. What Can Be Revoked: Keys, Subkeys, Certifications? . . . . .	4
2.1. Revoking a Primary Key . . . . .	5
2.1.1. Key Retirement . . . . .	5
2.1.2. Key Compromise . . . . .	5
2.1.3. Loss of Access . . . . .	5
2.2. Revoking a Subkey . . . . .	6
2.3. Revoking a Certification . . . . .	6
3. Challenges with OpenPGP Revocation . . . . .	7
3.1. Obtaining Revocation Information . . . . .	7
3.1.1. Revocation Stripping . . . . .	8
3.2. Revocations Using Weak Cryptography . . . . .	8
3.3. Revoking Primary Key Binding Signatures . . . . .	8
3.4. Implications for Revoked Key Material . . . . .	8
3.5. No Revocation Expiration . . . . .	9
3.6. Reasons for Revocation Mismatch . . . . .	9
3.7. Revocation Key Subpacket Challenges . . . . .	10
3.7.1. Finding the Revocation Key . . . . .	10
3.7.2. What If the Revocation Key is Itself Revoked or Unusable? . . . . .	10
3.7.3. Social Graph Leakage . . . . .	10
3.7.4. What if the Signature Containing the Revocation Key is Revoked or Superseded? . . . . .	11
3.7.5. What can the Revocation Key Revoke? . . . . .	12

3.8. Revocation Signature Subpacket limitations . . . . .	12
3.9. What About Revocations From the Future? . . . . .	13
4. Dealing With Revoked Certificates . . . . .	13
5. Hard vs. Soft Revocations . . . . .	13
5.1. When is Soft Revocation Useful? . . . . .	14
6. Revocation Certificates . . . . .	14
6.1. Handling a Revocation Certificate . . . . .	15
6.2. Publishing a Revocation Certificate . . . . .	15
7. Escrowed Revocation Certificate . . . . .	15
7.1. Escrowed Hard Revocation Workflow . . . . .	15
7.2. Escrowed Soft Revocation Workflow . . . . .	16
7.3. K-of-N Escrowed Revocation . . . . .	16
8. The Delegated Revoker . . . . .	16
8.1. Delegated Revoker Signature . . . . .	16
8.1.1. Transmitting a Delegated Revoker Signature in OpenPGP Certificate . . . . .	17
8.1.2. "Sensitivity"? . . . . .	17
8.2. Delegated Revoker Subpacket . . . . .	18
8.3. Delegated Revokers Cannot Be Superseded or Revoked . . . . .	18
8.4. Delegated Revokers Are Independent of Any OpenPGP Certificate . . . . .	18
8.5. Delegated Revoker Only Issues Key Revocation Signatures . . . . .	18
8.6. Cryptographic Algorithm Choices . . . . .	19
8.7. Reasonable Workflows . . . . .	19
8.7.1. Delegator Selection . . . . .	19
8.7.2. Delegated Key Selection . . . . .	19
8.7.3. Delegation Publication . . . . .	19
8.8. K-of-N Delegated Revokers . . . . .	19
9. Deprecation of the "Revocable" Signature Subpacket . . . . .	20
9.1. Non-functionality of the "Revocable" Signature Subpacket . . . . .	20
10. IANA Considerations . . . . .	21
10.1. Signature Subpacket Types: Add "Delegated Revoker" and Deprecate "Revocable" . . . . .	21
10.2. Reason for Revocation Code: Add Hard vs. Soft Column . . . . .	21
10.3. Signature Types: Add Delegated Revoker Row . . . . .	21
10.4. Signature Types: Update References . . . . .	21
10.5. Signature Subpacket Types: Update References . . . . .	21
11. Security Considerations . . . . .	22
12. References . . . . .	22
12.1. Normative References . . . . .	22
12.2. Informative References . . . . .	22
Appendix A. Augmenting SOP For Revocation . . . . .	22
Appendix B. Test Vectors . . . . .	22
Appendix C. Acknowledgements . . . . .	23
Appendix D. Substantive changes to this document . . . . .	23

D.1. Substantive Changes From draft-dkg-openpgp-revocation-02 to -03 . . . . .	23
D.2. Substantive Changes From draft-dkg-openpgp-revocation-01 to -02 . . . . .	23
D.3. Substantive Changes From draft-dkg-openpgp-revocation-00 to -01 . . . . .	23
Authors' Addresses . . . . .	23

## 1. Introduction

OpenPGP revocation is complicated. This document attempts to clean it up and build a consensus around syntax, semantics, use cases, and workflows.

The only substantive wire format changes are the introduction of the "Delegated Revoker" mechanism described in Section 8, and the deprecation of the little-used "Revocable" subpacket in Section 9.

### 1.1. Terminology

The term "OpenPGP Certificate" is used in this document interchangeably with "OpenPGP Transferable Public Key", as defined in Section 10.1 of [RFC9580].

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.

## 2. What Can Be Revoked: Keys, Subkeys, Certifications?

There are three kinds of signatures that perform revocation: Key Revocation (0x20), Subkey Revocation (0x28), and Certification Revocation (0x30).

- \* A Key Revocation Signature (0x20) directly invalidates a Primary Key packet, and thereby (indirectly) revokes a full OpenPGP certificate (a.k.a. "Transferable Public Key").
- \* A Subkey Revocation Signature (0x28) directly revokes a Subkey packet, without affecting other key material attached to the same Primary Key.
- \* A Certification Revocation Signature (0x30) revokes:
  - all previous Certification Signatures (0x10..0x13) over the same primary key and User ID or User Attribute, or

- all previous Direct Key Signatures (0x1f) over the same primary key, that were made by the same key that made the revocation.

All revocation types are permanent and cannot be un-revoked. A key may be temporarily invalidated by specifying an expiry date on a new Direct Key, Subkey Binding, or Certification Signature. This expiry date can then be overridden on a later signature of the same type.

## 2.1. Revoking a Primary Key

A Key Revocation signature invalidates the Primary Key packet that it is made over. By implication, this revokes the entire certificate (Transferable Public Key) anchored by the Primary Key.

### 2.1.1. Key Retirement

A key owner may wish to retire a key, for example if it is using an older algorithm or it is no longer required. This can be achieved by making a Key Revocation Signature with a soft revocation reason (see Section 5) and publishing it directly.

### 2.1.2. Key Compromise

If a key owner loses control of their private key material, for example if their storage device is stolen or their computer is infected with malware, they will normally wish to invalidate their key. This can be achieved by making a Key Revocation Signature with a Reason for Revocation of "Key Compromise" (0x02) and publishing it directly.

If the key owner has also lost access to their private key material, for example if all copies of it were stolen, they cannot generate a new revocation and must follow the "Loss of Access" procedure in the next section.

### 2.1.3. Loss of Access

If a key owner loses access to their private key material, for example if they forget an encryption passphrase or a storage medium is destroyed, they will generally wish to invalidate their key. This can be achieved by publishing an escrowed Key Revocation Signature (see Section 7).

If the key owner does not have such a revocation stored safely, there is nothing further that they can do cryptographically. In such circumstances, they will need to inform their correspondents by other means. See Section 2.3 below for possible alternative methods in a controlled environment.

## 2.2. Revoking a Subkey

A Subkey packet may be revoked because its private key material has been compromised. It is possible for a Subkey to be compromised without the Primary Key being affected, for example if the private Subkey and Primary Key material are stored on separate devices. In such a case, it is not necessary for a Subkey Revocation Signature to be generated ahead of time and escrowed, since the Primary Key is still usable and can generate a revocation as required.

FIXME: are other kinds of subkey revocation meaningful, see #1 ABG

## 2.3. Revoking a Certification

User ID and User Attribute self-certifications and Direct Key self-signatures can be explicitly expired or replaced by the keyholder by issuing a superseding signature, so the only reason for a certification revocation is for third-party certifications.

[RFC1991] provided guidance for the interpretation of Certification Revocations as follows:

```
<30> - public key packet and user ID packet, revocation ("I
retract all my previous statements that this key is related to
this user") (*)
```

While this language was not carried over into later specifications of the OpenPGP standard, neither was it explicitly contradicted.

FIXME: are first-party certification revocations meaningful, see #3 ABG

When Alice revokes her third-party certification over Bob's Primary Key and User ID, she is saying one of the following:

- \* Key is Compromised (0x02): "I believe that Bob's key has been compromised"
- \* User ID No Longer Valid (0x32): "I believe that Bob no longer controls this identity"

Hard third-party certification revocations are useful in an environment where Alice is treated as an authority (say as a member of a corporate IT department) but does not have control over Bob's key material or access to an escrowed revocation of Bob's key.

FIXME: we need to specify what a receiving application should do when seeing an 0x02 certification revocation made by a trusted authority;  
ABG

Alice's Certification Revocation signature packet could get attached to Bob's certificate by several methods:

- \* By submitting it to a keystore that performs an unauthenticated merge; this is however vulnerable to abuse
- \* By submitting it to a keystore whose administrators can override Bob's published certificate, for example a corporate directory
- \* By attaching it to her own key as an Embedded Signature subpacket (FIXME: using a method TBC, see #16 ABG)

Alice could issue a superseding certification of her own over Bob's User ID or User Attribute instead of using a soft revocation type, however she may wish to be explicit about the finality of her decision.

FIXME: Given an initial certification at time T, if the superseding certification has a timestamp of T+1, then will a verifier that cares about the certification still accept signatures from the key based on the User ID that were made exactly at time T? Alternately, if the superseding certification has a timestamp of time T exactly, will verifiers prefer its expiration date or the initial certification's expiration date (or lack thereof)?

FIXME: specify how third-party certification revocations interact with third-party direct signatures; ABG

### 3. Challenges with OpenPGP Revocation

This section describes a number of outstanding challenges with implementing OpenPGP revocation in the state of the field before this document. Some of these problems are fixed by this document.

#### 3.1. Obtaining Revocation Information

How does the user know that they have the correct revocation status? Where do they look for revocations from? With what frequency?

When the keyholder changes to a new key, how do they distribute revocations for older keys?

### 3.1.1. Revocation Stripping

Given the chance to tamper with an OpenPGP certificate, the simplest thing that an adversary can do is to strip signature subpackets. Stripping a revocation signature subpacket is trivial, and the resulting certificate looks valid.

An OpenPGP implementation needs a reliable channel to fetch revocation signatures from, and a reliable and well-indexed storage mechanism to retain them safely to avoid using revoked certificates.

### 3.2. Revocations Using Weak Cryptography

What if we find a Key Revocation signature made using SHA1 or MD5?

Should we consider the indicated key revoked?

### 3.3. Revoking Primary Key Binding Signatures

Primary keys sign Subkey Binding Signatures. Signing-capable subkeys sign Primary Key Binding Signatures.

A Primary Key Binding Signature is only valid in an Embedded Signature subpacket of another signature. If the enclosing signature is revoked, the embedded signature is no longer meaningful. It is therefore not necessary to define a method to explicitly revoke a Primary Key Binding Signature.

### 3.4. Implications for Revoked Key Material

If a primary key is revoked with Reason for Revocation 2 (key has been compromised), then an implementation MAY infer that any other certificate containing the same key material has also been compromised. Note that testing key material for equality is nontrivial due to flexibility in representation, and is therefore outside the scope of this document.

If a primary key is revoked for any reason other than key compromise, an implementation MUST NOT infer anything about any other certificate containing the same key material.

If a subkey is revoked for any reason, an implementation MUST NOT infer anything about any other certificate containing the same key material. This is because a key owner can create a valid subkey revocation signature over a subkey containing arbitrary key material:

- \* embedded Primary Key Binding Signatures are not required in Subkey Revocation Signatures



- \* an earlier valid Subkey Binding Signature is not required to validate a later Subkey Revocation Signature

Encryption subkeys cannot create embedded Primary Key Binding Signatures, but a malicious subkey binding over an arbitrary encryption subkey has no security implications, since the only person adversely affected would be the attacker themselves, whose correspondents would encrypt to the wrong key. By contrast, if a malicious revocation over such a subkey was interpreted as a valid revocation over the original key material, the key's actual owner might no longer be able to receive encrypted messages at all.

Therefore, the meaning of a Subkey Revocation Signature MUST be limited to the context of the primary key that made the revocation signature.

### 3.5. No Revocation Expiration

Key material can be marked with an expiration date (e.g. in a self-signature). Signatures themselves can also be marked with an expiration date.

While Revocation Signatures are signatures, the act of revocation is permanent, so expiration is not applicable to revocations.

An implementation generating a Revocation Signature MUST NOT include an Signature Expiration Time subpacket or a Key Expiration Time subpacket in either the hashed subpackets area or the unhashed subpackets area of the signature packet. An implementation encountering a Revocation Signature packet that contains either expiration subpacket MUST ignore the subpacket.

### 3.6. Reasons for Revocation Mismatch

How should an implementation interpret a Key Revocation signature or Subkey Revocation signature with Reason for Revocation subpacket with ID 32 ("User ID information is no longer valid")?

How should an implementation interpret a Certification Revocation with a Reason for Revocation with, say, ID 1 ("Key is superseded")?

Do we just say these Revocation signatures are invalid? Do we ignore the Reasons for Revocation subpacket?

### 3.7. Revocation Key Subpacket Challenges

The Revocation Key Subpacket is deprecated because it suffers from several significant challenges in use. The Delegated Revoker mechanism (described in Section 8) is intended to avoid all of these problems.

#### 3.7.1. Finding the Revocation Key

The "Revocation Key" subpacket contains only a fingerprint. If a verifier observes such a packet, and a Key Revocation Signature that claims to be issued by the identified key, how does the verifier obtain the revoking key itself if they do not already have a copy of it?

#### 3.7.2. What If the Revocation Key is Itself Revoked or Unusable?

What happens if the revocation key's public key packet is known, but it is not part of a certificate at all?

What if it is enclosed in a certificate, but that certificate indicates that it is expired, revoked, or otherwise invalid?

The following questions are based on the assumption that key A has pointed to key B in a "Revocation Key" subpacket.

What if B revokes both itself and key A: is A valid?

What if, instead, B indicates (via the deprecated "Revocation Key" subpacket) that key A is permitted to revoke key B? In this scenario, what happens when both A and B revoke each other?

What if A designates that B can revoke A, and B designates that C can revoke B? In that case, if C revokes B and then B revokes A, is A still valid?

#### 3.7.3. Social Graph Leakage

If it's possible to find a certificate containing the matching fingerprint in a deprecated "Revocation Key" subpacket, then an observer can learn who the keyholder trusts even when no revocation is needed.

An attacker that wants to target Alice, for example, can observe that Alice has indicated Bob seems trustworthy if Alice has pointed to Bob's key's fingerprint with a deprecated "Revocation Key" subpacket. The attacker might then go after Bob as a way to get to Alice.

### 3.7.4. What if the Signature Containing the Revocation Key is Revoked or Superseded?

Section 5.2.3.3 of [RFC4880] states:

Revoking a direct-key signature cancels that signature.

and

An implementation that encounters multiple self-signatures on the same object may resolve the ambiguity in any way it sees fit, but it is RECOMMENDED that priority be given to the most recent self-signature.

The revised version, Section 5.2.3.10 of [RFC9580] makes this last sentence even stronger:

An implementation that encounters multiple self-signatures on the same object MUST select the most recent valid self-signature, and ignore all other self-signatures.

Consider the following sequence of events:

- \* At time t0, key A makes a self-signed Direct Key Signature X on itself that contains a Revocation Key subpacket pointing to key B.
- \* At time t1, key A decides to update preferences or expiration date on itself and issues a new Direct Key Signature Y (which lacks a Revocation Key subpacket).
- \* At time t2, key B produces a Key Revocation signature Z to revoke key A.

The verifier examines this sequence of packets: A, X, Y, Z.

X appears to have been superseded by Y. Should A be considered revoked?

But what if signature packet Y was a revocation signature instead:

- \* At time t1, key A creates a Certification Revocation signature Y over Direct Key signature X.

Or, what if signature packet Y pointed to a different Revocation Key:

- \* At time  $t_1$ , key A creates a Direct Key Signature Y that looks exactly like X, except that its Revocation Key subpacket points to key C.

If, in any of these situations, the verifier does not consider A to be revoked by Z due to the presence of signature Y, then the mechanism does not work to protect the keyholder of A. An adversary who has taken control of A can create a signature packet Y to evade the third-party revocation capabilities that B is supposed to wield.

If delegating revocation power to a third party is intended to defend against an adversary, this implies that the guidance about superseding signatures cannot apply to signature packets that contain a Revocation Key. But then, if signature X is not revoked or superseded by signature Y (in whatever form Y takes), how should implementations deal with the other subpackets in signature X?

### 3.7.5. What can the Revocation Key Revoke?

Surely it can issue a Key Revocation signature that covers the primary key itself.

But can it issue a Subkey Revocation signature on behalf of the primary key? Can it issue a Certification Revocation signature on behalf of the primary key?

### 3.8. Revocation Signature Subpacket limitations

When generating a revocation signature, an implementation:

- \* SHOULD include a Signature Creation Time subpacket
- \* SHOULD NOT set the critical bit for any subpacket
- \* MUST NOT set the critical bit for any subpacket other than Signature Creation Time
- \* MUST NOT place Signature Creation Time or Reason for Revocation packets in the unhashed area

When consuming a revocation signature, an implementation:

- \* MUST ignore the critical bit for every subpacket
- \* MUST ignore any Signature Creation Time or Reason for Revocation subpacket in the unhashed area

An implementation MUST support the Signature Creation Time subpacket. If a revocation signature does not contain a valid Signature Creation Time subpacket, a receiving implementation MAY treat it as if it was created in the infinite past.

### 3.9. What About Revocations From the Future?

If a Revocation signature appears to have been made in the future, its interpretation will depend on whether it is hard or soft:

- \* If a hard revocation is from the future, then its creation date is irrelevant, since hard revocations are retrospective. Hard revocations MUST be treated as if their creation date was in the infinite past, regardless of the value of the creation date subpacket.
- \* If a soft revocation is from the future, then the revocation SHOULD NOT take effect until that date.

## 4. Dealing With Revoked Certificates

Implementations MUST NOT encrypt to a revoked certificate. Implementations MUST NOT accept a signature made by a revoked certificate as valid unless the revocation is "soft" (see Section 5) and the timestamp of the signature predates the timestamp of the revocation. Implementations MUST NOT use secret key material corresponding to a revoked certificate for signing, unless the secret key material also corresponds to a non-revoked certificate.

Implementations MAY use the secret key material corresponding to a revoked certificate.

## 5. Hard vs. Soft Revocations

Reasons for Revocation subpacket allows different values.

Some of them suggest that a verifier can still accept signatures from before the timestamp of the Revocation. These are "soft" revocations.

All the rest require that a verifier MUST treat the certificate as "hard" revoked, meaning that even signatures that have creation timestamps before the creation timestamp of the revocation signature should themselves be rejected.

### 5.1. When is Soft Revocation Useful?

Expiration makes just as much sense as a soft revocation in many circumstances, and is typically better supported. Soft revocation can however be useful if the signer wishes to explicitly indicate that their decision is final. Since revocations are permanent, a correspondent who sees a soft revocation does not need to poll for further updates to see whether an expiration date has been extended. The only reason to poll for an update to a soft-revoked key would be to check whether the soft revocation had been upgraded to a hard revocation.

Since an encryption subkey is not useful for historical purposes, only for creating new encrypted data, there is no practical distinction between soft and hard revocation reasons, and all soft encryption subkey revocations SHOULD be treated as hard revocations with reason "none" (0x00).

FIXME: what about encryption-only primary keys; ABG

## 6. Revocation Certificates

A revocation certificate indicates that a given primary key is revoked.

This can take two common forms. Each form is a sequence of OpenPGP packets:

- \* A standalone Key Revocation signature packet by key X over X (this form is valid only for primary keys earlier than version 6)
- \* Primary Key X + Key Revocation signature by X over X

Additionally, there is a deprecated form:

- \* Primary Key X + Direct Key Signature with Revocation Key subpacket pointing to Y + Key Revocation signature by Y over X (this form is valid only for primary keys earlier than version 6)

This document introduces a new form in Section 8:

- \* Primary Key X + Delegated Revoker Signature with Delegated Revoker Subpacket containing Y + Key Revocation signature by Y over X

### 6.1. Handling a Revocation Certificate

When an implementation observes any of the above forms of revocation certificate for a certificate with primary key X, it should record it and indicate that X has been revoked and is no longer to be used, along with all of its User IDs and Subkeys.

### 6.2. Publishing a Revocation Certificate

FIXME: talk about interactions with HKP, VKS, WKD, OPENPGPKEY (DANE), or other key discovery methods?

## 7. Escrowed Revocation Certificate

An escrowed revocation certificate is just a valid revocation certificate that is not published. The parties who can retrieve or reassemble the escrowed revocation certificate can publish it to inform the rest of the world that the certificate has been revoked. It is described in Section 13.9 of [RFC9580].

Since the reason for publishing an escrowed revocation cannot be known in advance, escrowed revocations SHOULD NOT include a Reason for Revocation subpacket. If such a subpacket is included, it SHOULD explicitly state a reason of "none" (0x00).

Since the reason for publishing an escrowed revocation cannot be known in advance, escrowed revocations SHOULD NOT include a Reason for Revocation subpacket. If such a subpacket is included, it SHOULD explicitly state a reason of "none" (0x00).

In what circumstances does escrowed revocation work? When is it inappropriate?

### 7.1. Escrowed Hard Revocation Workflow

An escrowed hard revocation certificate covers the use case where where the keyholder has lost control of the secret key material, and someone besides the keyholder may have gotten access to the secret key material.

At key creation time, keyholder creates a hard revocation certificate. Optionally, they encrypt it to a set of trusted participants. The keyholder stores the revocation certificate somewhere they or one of the trusted participants will be able to access it.

If the keyholder sends it to any trusted participant immediately, that participant can trigger a revocation any time they like. In this case, the keyholder and the trusted participants should clarify between themselves what an appropriate signal should be for when the trusted participant should act

If physical access is retained by the keyholder, then the keyholder has to be capable of consenting for the revocation to be published.

## 7.2. Escrowed Soft Revocation Workflow

Do regular updates of the escrowed revocation (e.g. after each signing). Store them somewhere safe?

## 7.3. K-of-N Escrowed Revocation

FIXME: how to split an escrowed revocation certificate among N parties so that any K of them can reconstruct it.

## 8. The Delegated Revoker

[ See also [https://gitlab.com/openpgp-wg/rfc4880bis/-/merge\\_requests/18](https://gitlab.com/openpgp-wg/rfc4880bis/-/merge_requests/18) where this was originally specified, in a slightly different form ]

This document introduces a new mechanism for permitting a distinct key to revoke an OpenPGP certificate. It should effectively replace the deprecated "Revocation Key subpacket", and should resolve the concerns described in {#revocation-key-subpacket-challenges}.

It uses a novel signature type and a novel subpacket type, is self-contained, irrevocable, and can only be used to revoke the entire OpenPGP certificate rooted in the primary key it corresponds to.

### 8.1. Delegated Revoker Signature

This introduces a new Signature Type, "Delegated Revoker Signature" with type ID TBD.

This signature type is made over data structured in the same way as a Direct Key Signature, but it does not supersede or replace any other signature, and it cannot be revoked or superseded itself. It is a permanent delegation.

A Delegated Revoker Signature MUST be made over the primary key of a given certificate, and its hashed subpackets area MUST contain exactly one each of the following three subpackets:



- \* Signature Creation Time (see Section 5.2.3.11 of [RFC9580])
- \* Issuer Fingerprint (see Section 5.2.3.35 of [RFC9580])
- \* Delegated Revoker (defined in Section 8.2 of this document)

The Issuer Fingerprint subpacket contains the fingerprint of the primary key that is asserting this delegation. All subpackets MUST be marked as Critical.

No corresponding revocation signature type is defined for the Delegated Revoker Signature. This is intentional.

FIXME: What should a verifier do if the set of hashed subpackets does not match this list?

#### 8.1.1. Transmitting a Delegated Revoker Signature in OpenPGP Certificate

The Delegated Revoker Signature packet can be placed in an OpenPGP certificate immediately after the Primary Key packet, before any Key Revocation Signature or Direct Key Signature packet.

FIXME: test implementations with this placement strategy to see whether they choke on the certificate. If this does not work, consider recommending its inclusion in an unhashed Embedded Signature subpacket the relevant associated Key Revocation Signature packet (if it is travelling with the revocation), or in such a subpacket of a Direct Key Signature packet directly.

#### 8.1.2. "Sensitivity"?

FIXME: how do we deal with avoiding a leak of the existence of this relationship (e.g., the "Sensitive" bit in the deprecated "Revocation Key" subpacket)? Do we need to?

Someone concerned about the risk of social graph leakage (see Section 3.7.3) can simply mint a new secret key and encrypt its corresponding Secret Key packet to their preferred revoker.

Or should we add an "Exportable" subpacket to the list above and describe its syntax more explicitly?

Alternately: what if we said that this is simply *\_always\_* treated as sensitive, in that without explicitly being part of the described workflow, it must not be transmitted except in the presence of a valid revocation? This puts the burden on the holder of the secret key to keep a copy of the delegation lying around, which is a novel use case.

## 8.2. Delegated Revoker Subpacket

The Delegated Revoker Subpacket has type ID 36.

It is only valid in the hashed subpackets section of a Delegated Revoker Signature (see Section 8.1) over a primary key. It **MUST** be marked as Critical.

The contents of this subpacket are a full Public Key packet, see Section 5.5.1.1 of [RFC9580].

The embedded Public Key packet **MUST** be signing-capable.

## 8.3. Delegated Revokers Cannot Be Superseded or Revoked

Unlike other OpenPGP Signature packets, a Delegated Revoker Signature cannot be revoked, and is not superseded by any other Signature packet, including other Delegated Revoker Signature packets. If multiple valid Delegated Revoker Signatures are issued by a primary key X, they are all capable of issuing Key Revocation signatures over X on behalf of X.

## 8.4. Delegated Revokers Are Independent of Any OpenPGP Certificate

This Public Key **MAY** be the same Public Key packet that is the root of a larger OpenPGP certificate, but it need not be. In the Delegated Revoker context, this Public Key packet is used on its own, regardless of the status of any matching OpenPGP certificate.

## 8.5. Delegated Revoker Only Issues Key Revocation Signatures

If an OpenPGP certificate with primary key X has an associated Delegated Revoker containing Public Key Y, that only indicates that the Y **MAY** make a valid Key Revocation signature on behalf of X, revoking X itself.

The Delegated Revoker Public Key (Y in the example above) **MUST NOT** be used to validate any other type of signature associated with that OpenPGP certificate.

FIXME: should we constrain the types of Key Revocations it can issue (i.e., the contents of any Reason for Revocation subpackets, or "hard" or "soft" choices)?

## 8.6. Cryptographic Algorithm Choices

FIXME: What if the Delegated Revoker Signature is made over a digest algorithm that is deemed unsafe in the future? FIXME: What if the embedded Public Key Packet is known to be weak or compromised?

## 8.7. Reasonable Workflows

The Delegated Revoker mechanism is only useful for a potential scenario where the keyholder has lost control of the primary secret key. Otherwise, the keyholder could simply issue the desired Key Revocation signature (type 0x20) directly.

If the keyholder needs a hard revocation, and they have access to an escrowed Key Revocation signature, they can use that.

So the circumstances where a Delegated Revoker is relevant

### 8.7.1. Delegator Selection

Keyholder needs to choose who they think will be responsible for handling the delegated responsibility of revoking when the time is needed. This could be another individual, or it could be a machine that has distinct security and operational characteristics from the machine that holds the primary key.

### 8.7.2. Delegated Key Selection

- \* Keyholder generates revocation secret key
- \* Keyholder selects signing-capable key or subkey already belonging to delegate

### 8.7.3. Delegation Publication

- \* private communication
- \* public (keyservers)

## 8.8. K-of-N Delegated Revokers

FIXME: should this document outline how a group of trusted parties could effectively revoke a given certificate that authorized them to do so?

## 9. Deprecation of the "Revocable" Signature Subpacket

The "Revocable" subpacket is not commonly supported, and when used as described is effectively non-functional. It is hereby deprecated.

### 9.1. Non-functionality of the "Revocable" Signature Subpacket

Section 5.2.3.20 of [RFC9580] states:

Signatures that are not revocable have any later revocation signatures ignored. They represent a commitment by the signer that he cannot revoke his signature for the life of his key. If this packet is not present, the signature is revocable.

But this is not an effective constraint on the key owner's future behaviour:

- \* Since there is no such thing as a document revocation signature, this is only applicable to self-signatures and third party certifications.
- \* If a key is compromised, then the timestamp on any revocation can be trivially backdated. So this must only apply to revocations by valid or soft-revoked keys.
- \* A soft revocation of a self-signature or third-party certification is functionally equivalent to a later signature with an expiry date, which is not covered by the "Revocable" semantics.
- \* A hard revocation has the same semantics regardless of its creation date. In particular, an escrowed revocation signature (such as the revocation signatures commonly made at key generation time) will have a creation time significantly in the past compared to when it is typically published. A "non-revocable" certification created after the escrowed revocation sig cannot prevent the escrowed revocation taking effect.

Therefore any "non-revocable" signature can still be effectively "revoked" by one of the following unremarkable events:

- \* by a later signature with an explicit expiry date, which has the same practical effect as a soft revocation,
- \* or by an escrowed hard revocation, which has the same practical effect as a later hard revocation.

The "revocable" subpacket is therefore non-functional.

## 10. IANA Considerations

This draft asks IANA to do several things, all within the OpenPGP protocol group.

### 10.1. Signature Subpacket Types: Add "Delegated Revoker" and Deprecate "Revocable"

The "OpenPGP Signature Subpacket Types" registry should be updated as follows:

- \* Add an entry "Delegated Revoker" to OpenPGP subpackets, type 36, referencing this document, Section 8.2.
- \* Mark the "Revocable" subpacket entry as "deprecated", referencing this document, Section 9.

### 10.2. Reason for Revocation Code: Add Hard vs. Soft Column

The "OpenPGP Reason for Revocation Code" registry should add a column to indicate "Hard/Soft". Only "Key is Superseded" and "Key is retired and no longer used" are marked "Soft". All other values should be treated as "Hard".

### 10.3. Signature Types: Add Delegated Revoker Row

Add an entry "Delegated Revoker Signature" to the "OpenPGP Signature Types" registry, type ID TBD, referencing this document, Section 8.1.

### 10.4. Signature Types: Update References

The "OpenPGP Signature Types" registry entries should have References updated:

- \* 0x20 references should additionally point to this document, Section 2.1
- \* 0x28 references should additionally point to this document, Section 2.2
- \* 0x30 references should additionally point to this document, Section 2.3

### 10.5. Signature Subpacket Types: Update References

The "Reason for Revocation Code" entry in the "OpenPGP Signature Subpacket Types" registry should have its References column updated to point to this document.

## 11. Security Considerations

This document describes ways that the OpenPGP ecosystem deals with revoked certificates. Revocation is a security measure because it is a method of last resort for dealing with cryptographic credentials that are known to have failed for one reason or another.

The entire document is therefore focused on security. Implementers who ignore this guidance may either allow known-bad key material to be used (by ignoring a valid revocation signature) or may issue revocation signatures that other implementations are likely to ignore.

## 12. References

### 12.1. Normative References

- [RFC9580] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <<https://www.rfc-editor.org/info/rfc9580>>.
- [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>>.
- [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>>.

### 12.2. Informative References

- [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>>.

## Appendix A. Augmenting SOP For Revocation

FIXME: Can all of the plausible workflows described in this document be done with the Stateless OpenPGP Interface? If not, what is missing?

## Appendix B. Test Vectors

FIXME: This document should include several different valid OpenPGP Revocation Certificates.

## Appendix C. Acknowledgements

Phil Zimmermann motivated the Delegated Revoker work. Thanks to the people who have given useful reviews, feedback, and suggestions on this document, including: Daniel Huigens, and Heiko Schaefer.

## Appendix D. Substantive changes to this document

RFC Editor Note: Please delete this section before publication.

### D.1. Substantive Changes From draft-dkg-openpgp-revocation-02 to -03

- \* Updated references to RFC9580

### D.2. Substantive Changes From draft-dkg-openpgp-revocation-01 to -02

- \* Deprecated the Revocable subpacket
- \* Un-deprecated Subkey and Certification Revocations

### D.3. Substantive Changes From draft-dkg-openpgp-revocation-00 to -01

- \* Enumerate problems with Revocation Key subpacket, including superseded signatures
- \* Offer doubt about deprecating Subkey Revocation and Certification Revocation (maybe a future version will un-deprecate with clearer guidance?)
- \* Change mechanism from Direct Key Signature to dedicated Delegated Revoker Signature Type

## Authors' Addresses

Daniel Kahn Gillmor  
ACLU  
Email: dkg@fifthhorseman.net

Andrew Gallagher  
Email: andrewg@andrewg.com