

openpgp  
Internet-Draft  
Updates: 9580 (if approved)  
Intended status: Standards Track  
Expires: 16 April 2026

D. Shaw  
Jabberwocky Tech  
A. Gallagher, Ed.  
PGPKeys.EU  
13 October 2025

OpenPGP Key Replacement  
draft-ietf-openpgp-replacementkey-06

## Abstract

This document specifies a method in OpenPGP to suggest a replacement for an expired, revoked, or deprecated primary key.

## 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://andrewgdotcom.gitlab.io/openpgp-replacementkey>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-openpgp-replacementkey/>.

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/andrewgdotcom/openpgp-replacementkey>.

## 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 16 April 2026.

## 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 . . . . .	3
2. Conventions and Definitions . . . . .	4
2.1. Terminology . . . . .	4
3. The Replacement Key Subpacket . . . . .	5
4. Format of the Replacement Key Subpacket . . . . .	5
4.1. Key Imprints . . . . .	8
4.2. Graph Topology . . . . .	8
5. Interpretation of the Replacement Key Subpacket . . . . .	10
5.1. Identity Equivalence Binding . . . . .	10
5.1.1. Time Evolution of Identity Equivalence . . . . .	11
5.1.2. Consequences of Identity Equivalence . . . . .	12
5.2. Absence of an Identity Equivalence Binding . . . . .	13
5.2.1. Limiting Chaining of Non-Equivalent Replacements . . . . .	13
5.3. Reasons for Revocation . . . . .	14
6. Selection of Encryption Subkeys . . . . .	14
7. Security Considerations . . . . .	15
8. IANA Considerations . . . . .	15
9. References . . . . .	15
9.1. Normative References . . . . .	16
9.2. Informative References . . . . .	16
Appendix A. Example Workflows . . . . .	16
A.1. Alice Generates a New Primary Key . . . . .	16
A.1.1. Alice's Actions . . . . .	16
A.1.2. Bob's Actions . . . . .	17
A.1.3. Alice's Second Device . . . . .	18
A.2. Alice Revokes Her Deprecated Primary Key . . . . .	18
A.2.1. Alice's Actions . . . . .	18
A.2.2. Bob's Actions . . . . .	18
A.2.3. Carol's Actions . . . . .	19
A.2.4. Alice's Second Device . . . . .	19
A.3. Time Evolution of an Identity Equivalence Group . . . . .	20
Appendix B. Acknowledgments . . . . .	21

Appendix C. Document History . . . . .	21
C.1. Changes Between draft-ietf-openpgp-replacementkey-05 and -06 . . . . .	21
C.2. Changes Between draft-ietf-openpgp-replacementkey-04 and -05 . . . . .	22
C.3. Changes Between draft-ietf-openpgp-replacementkey-03 and -04 . . . . .	22
C.4. Changes Between draft-ietf-openpgp-replacementkey-02 and -03 . . . . .	23
C.5. Changes Between draft-ietf-openpgp-replacementkey-01 and -02 . . . . .	23
C.6. Changes Between draft-ietf-openpgp-replacementkey-00 and -01 . . . . .	23
C.7. Changes Between draft-gallagher-openpgp-replacementkey-02 and draft-ietf-openpgp-replacementkey-00 . . . . .	23
C.8. Changes Between draft-gallagher-openpgp-replacementkey-01 and -02 . . . . .	24
C.9. Changes Between draft-gallagher-openpgp-replacementkey-00 and -01 . . . . .	24
C.10. Changes Between draft-shaw-openpgp-replacementkey-00 and draft-gallagher-openpgp-replacementkey-00 . . . . .	24
Authors' Addresses . . . . .	24

## 1. Introduction

The OpenPGP message format [RFC9580] defines two ways to invalidate a primary key. One way is that the primary key may be explicitly revoked via a key revocation signature. OpenPGP also supports the concept of key expiration, a date after which the key should not be used. When a primary key is revoked or expires, very often there is another primary key that is intended to replace it.

A key owner may also create a new primary key that is intended to deprecate and replace their existing primary key, but without revoking or expiring that key. This is useful during the rollout of new key versions and algorithms which may not (yet) enjoy universal support. In such cases, a key owner may prefer that their correspondents use their new primary key, but if this is not possible for technical reasons they may continue to use the non-preferred key, which remains valid.

In the past some key owners have created key transition documents, which are signed, human-readable statements stating that a newer primary key should be preferred by their correspondents. It is desirable that this process be automated through a standardised machine-readable mechanism.

This document is to specify the format of a Signature Subpacket to be optionally included in a revocation signature or direct self-signature over a primary key. This subpacket contains a pointer to a suggested replacement for the (now deprecated) primary key that is signed over, or one or more (deprecated) primary keys for which the current primary key is the suggested replacement. The replacement certificate may then be automatically retrieved by the key owner's correspondents and (if supported and validated) used instead of the deprecated certificate.

## 2. Conventions and Definitions

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.

### 2.1. Terminology

In OpenPGP, the term "key" has often been used broadly to describe different concepts, which can lead to confusion. To avoid ambiguity in this document, we define the following terms:

- \* "Replacement Primary Key" and "Deprecated Primary Key": These refer to a primary key as contained in an OpenPGP Certificate (Section 10.1 of [RFC9580]) or Transferable Secret Key (Section 10.2 of [RFC9580]).
- \* "Target Key": This refers to either a replacement or deprecated primary key that is referenced in a Replacement Key subpacket.
- \* "Current Primary Key": This refers to the primary key that contains the self-signature being discussed.
- \* "Replacement Certificate", "Deprecated Certificate" and "Current Certificate": These refer to the certificate that contains the respective primary key.

The term "OpenPGP Certificate", or just "certificate", is used in this document interchangeably with "OpenPGP Transferable Public Key".

This document also introduces the following terms:

- \* An "identity" is any identifying label such as an email address or "real name"; it is not limited to User IDs, for example it may be a record in a local database.

- \* An "identity claim" is any statement, explicit or implied, that an identity belongs to a particular primary key; it does not need to be a certification signature, and is not necessarily true.
- \* An "identity link" is an identity claim that is considered to be true and accurate by the receiving implementation.
- \* An "Identity Equivalence Binding" is a doubly-linked, directed relationship between two primary keys, one of which is the stated replacement for the other.
- \* An "Identity Equivalence Set" is a collection of two or more primary keys whose Identity Equivalence Bindings form a maximal connected graph.

### 3. The Replacement Key Subpacket

The Replacement Key subpacket is a Signature Subpacket as specified in Section 5.2.3.7 of [RFC9580], and all general Signature Subpacket considerations from there apply here as well. The value of the Signature Subpacket type octet for the Replacement Key subpacket is 100 (TEMPORARY, permanent code point TBC).

The Replacement Key subpacket is a statement by the key owner that either:

- \* The current primary key is replaced by another primary key.
- \* The current primary key is the replacement for one or more deprecated primary key(s).

Use of the Replacement Key subpacket is optional. The absence of a Replacement Key subpacket SHOULD NOT be interpreted as meaning that there is no replacement or deprecated primary key(s) relating to the current primary key.

The Replacement Key subpacket MUST only be used in the hashed subpackets area of a Primary Key Revocation Section 5.2.1.11 of [RFC9580] or Direct Key signature Section 5.2.1.10 of [RFC9580]. A receiving implementation MUST ignore any Replacement Key subpacket found elsewhere.

### 4. Format of the Replacement Key Subpacket

The format of the Replacement Key subpacket is:

Octets	Field	Notes
1	Class	
1	Record length (1)	
1	Target Key Version (1)	
N1	Target Key Fingerprint (1)	
M	Target Key Imprint (1)	
1	Record length (2)	optional
1	Target Key Version (2)	optional
N2	Target Key Fingerprint (2)	optional
M	Target Key Imprint (2)	optional
...	...	...

Table 1: Replacement Key Subpacket Fields

The class octet contains flags that indicate the form and semantics of the subpacket:

Flag bit	Flag name	Form of remainder of packet
0x01	Backward reference(s)	Multiple targets may be given

Table 2: Replacement Key Subpacket Flags

The 0x01 bit of the class octet is the "backward reference(s)" bit. When set, this means that the target key(s) identified by the packet are the primary keys for which the current primary key is the replacement primary key. Otherwise, the subpacket represents a forward reference and the target key is the replacement primary key for the current primary key.

All other flag bits of the class octet are currently undefined. All undefined flags MUST be zeroed when generating the class octet. An implementation that encounters a class octet with nonzero bits that it does not recognise MUST ignore those bits.

Note that if the critical bit on the Replacement Key subpacket is set, a receiving implementation could consider the whole self-signature to be in error (Section 5.2.3.7 of [RFC9580]). The critical bit therefore SHOULD NOT be set on the Replacement Key subpacket.

The remainder of the subpacket contains one or more target records of the form:

( Record Length || Target Key Version || Target Key Fingerprint || Target Key Imprint )

The Record Length field contains a one-octet number which indicates the length of the next three fields in octets. If a receiving implementation does not understand the target key version, it SHOULD ignore the target record and skip to the next. The Record Length field is authoritative, and a receiving implementation MUST NOT infer the length of a target record by any other means. If the Record Length field indicates that a target record contains more octets than expected, a receiving implementation MUST ignore any trailing extra octets. If a target record contains fewer octets than expected, it is malformed and MUST be ignored.

- \* If the class octet does not have the 0x01 bit set, the subpacket MUST contain exactly one target record to identify the replacement primary key.
- \* If the class octet has the 0x01 bit set, the subpacket contains one or more target records, to identify the deprecated primary key(s) that the current primary key is a replacement for.

If a subpacket contains an unexpected number of target records, it is malformed and MUST be ignored.

The length of the Target Key Fingerprint field (N) MUST equal the fingerprint length corresponding to the immediately preceding Target Key Version field, e.g. 20 octets for version 4, or 32 octets for version 6. The length of the Target Key Imprint field (M) MUST equal the length of the output of the digest algorithm used by the enclosing signature, e.g. 32 octets for SHA2-256.

If a replacement or deprecated primary key is unknown, then a Replacement Key subpacket SHOULD NOT be included in the signature.

#### 4.1. Key Imprints

An imprint of a public key packet is a generalisation of a fingerprint. It is calculated in the same way as the fingerprint, except that it may use a digest algorithm other than the one specified for the fingerprint. Conversely, the fingerprint of a public key packet can be considered a special case of an imprint. A public key packet has only one fingerprint, but may have any number of imprints, each using a different digest algorithm.

When used in a Replacement Key subpacket, an imprint **MUST** use the same digest algorithm as the enclosing signature. This guards against key-substitution attacks when referring to keys that use weaker digest algorithms in their fingerprints. If the signature's digest algorithm is the same as that used by the fingerprint, then the imprint and the fingerprint will be identical. In such a case, the imprint **MUST** still be included for parsing reasons.

A receiving implementation **MAY** use the fingerprint to locate a target key, but **MUST** verify that the imprint matches.

#### 4.2. Graph Topology

A given signature **MUST** contain at most one Replacement Key subpacket in its hashed subpacket area. If a signature contains more than one such subpacket, even if malformed, a receiving implementation **MUST** disregard them all.

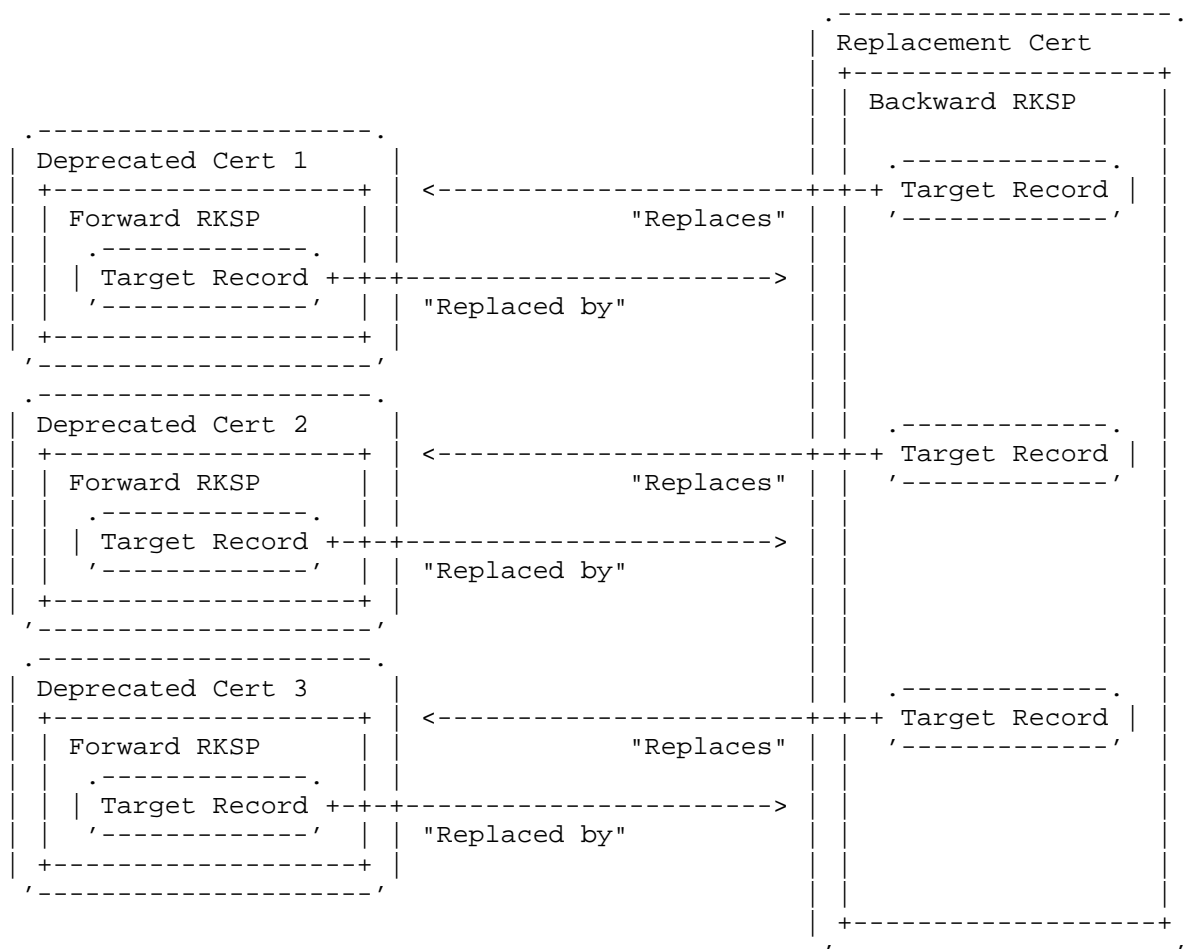
A certificate **MAY** have multiple signatures that contain Replacement Key subpackets, however at most one Replacement Key subpacket in a certificate is current (and therefore meaningful) at any given time. If the primary key is revoked with a Reason for Revocation of "Key is superseded" Section 5.2.3.31 of [RFC9580], the current Replacement Key subpacket (if any) is found in the most recent valid Key Revocation signature. If it is not revoked, the current Replacement Key subpacket (if any) is found in the most recent valid Direct Key signature. If the most recent valid Key Revocation or Direct Key signature does not contain a Replacement Key subpacket, or the primary key is revoked with any other Reason for Revocation, then there is no current Replacement Key subpacket.

This imposes a simple graph topology:

- \* A deprecated certificate **MUST NOT** claim to have more than one replacement.
- \* A deprecated certificate that claims to have a replacement **MUST NOT** claim to be the replacement for any other(s).



In addition, the order of the deprecated primary keys specified in a backward-reference Replacement Key subpacket is meaningful. If a replacement primary key is supported by a receiving implementation, but is not usable for the desired purpose (for example, it may not have an encryption-capable subkey), the implementation MAY use the ordering of the deprecated primary keys in its backward Replacement Key subpacket (if one exists) to indicate which deprecated primary key is preferred as a fallback. The deprecated primary keys SHOULD therefore be listed in order of decreasing preference.



"RKSP" = Replacement Key Subpacket

Figure 1: Example Topology of Replacement Key Subpackets

## 5. Interpretation of the Replacement Key Subpacket

The relationships expressed by Replacement Key subpackets may be either singly- or doubly-linked. Doubly-linked Replacement Key subpackets have a more consequential interpretation. If a Replacement Key subpacket is found in a Key Revocation signature, then the Reason for Revocation subpacket provides crucial additional context.

### 5.1. Identity Equivalence Binding

The existence of a matching pair of forward- and backward-reference Replacement Key subpackets on the most recent direct self-signatures or key revocations over two primary keys, with each referring to the other primary key, forms an Identity Equivalence Binding. A collection of certificates whose Identity Equivalence Bindings form a maximal connected graph (see Section 4.2) is an Identity Equivalence Set, and all members of that set are said to be Identity Equivalent to each other.

If an implementation links a particular identity (such as a User ID or a database record) to one primary key by a means other than Identity Equivalence, then it SHOULD treat that identity as being linked, in the same manner and to the same extent, with each other primary key in the same Identity Equivalence Set. Each such "derived" identity link is a subsidiary relationship of the original "direct" identity link. If the same identity is directly linked into the Identity Equivalence Set by multiple identity claims, the derived link(s) are subsidiary to the strongest or most reliable direct link. A derived identity link is otherwise independent of any identity claims over the other primary keys, or lack thereof. Each distinct identity SHOULD be modelled separately; a direct or derived link to one identity makes no statement about any other identities found in the Identity Equivalence Set.

The equivalence binding is invalidated under the following circumstances:

- \* if either primary key is revoked with a Reason for Revocation other than "Key is superseded".
- \* if either primary key overrides the equivalence binding with a new direct self-signature that a) does not contain a Replacement Key subpacket, or b) contains a Replacement Key subpacket that does not refer to the other key, or c) contains a Replacement Key subpacket that inverts the direction of the binding, and there is no corresponding update to the other certificate.

- \* if either signature that forms the equivalence binding has expired.

Note however:

- \* If either primary key is expired or revoked with a Reason for Revocation of "Key is superseded", the equivalence binding is unaffected.
- \* If either primary key is revoked with any other Reason for Revocation, then the equivalence binding is invalidated but the other key is not revoked.
- \* Other properties (such as expiry dates, usage preferences, custom notations) SHOULD NOT be applied to other members of the Identity Equivalence Set.

If a backward Replacement Key subpacket refers to multiple deprecated keys, it is possible that only some of those have a valid Identity Equivalence Binding. If a particular Identity Equivalence Binding is invalid, it does not invalidate the other bindings associated with the same backward Replacement Key subpacket.

Identity Equivalence is transitive; if A and B are both Identity Equivalent to C (in other words, if C replaces both A and B), then A is also Identity Equivalent to B. This transitivity is the basis for an Identity Equivalence Set, in which an identity linked to any of the primary keys is linked to them all. This equality of identity, and the distinction between direct and derived identity links, is independent of the order of preference of fallback primary keys (Section 4.2).

Members of an Identity Equivalence Set MUST be treated as a single key for the purposes of the Web of Trust, particularly when relying on multiple independent certification pathways.

#### 5.1.1. Time Evolution of Identity Equivalence

An implementation MUST NOT assume that Identity Equivalence Bindings have any permanent significance. For example, if an MUA relies solely upon an Identity Equivalence Binding between A and B to validate B, the validity of B at a future date depends on the continuing validity of the Identity Equivalence Binding. If the binding is no longer valid, and there are no other certification pathways to B, then B is no longer valid.

It is therefore RECOMMENDED that applications attempt to find alternative certification pathways for replacement certificates. The optimal method of obtaining alternative certification pathways is application-dependent, and therefore beyond the scope of this document. It should be noted however that similar time evolution concerns also apply to other methods of validation, such as the Web of Trust.

#### 5.1.2. Consequences of Identity Equivalence

Identity Equivalence Binding operates at the same conceptual level as subkey binding. A subkey is linked to a particular identity if its primary key is linked to that identity. It is not possible to limit the use of particular subkeys of the same primary to particular identities.

Similarly, a primary key is linked to an identity if any of the primary keys in its Identity Equivalence Group is so linked. This means that it is not possible to limit the use of particular certificates in the group to particular identities. Any identities that the key owner's correspondents link to any member of an Identity Equivalence Group will be equally linked to them all.

For example, if a key owner has a particular User ID in one certificate, but not in an Identity Equivalent certificate, her correspondents will still link that User ID with both certificates. If so, they may (in some circumstances) send messages encrypted to one of the certificates but addressed to an identity which that particular certificate does not claim.

An implementation SHOULD warn the user if they try to create an Identity Equivalence Group using certificates with mismatched current User IDs. It SHOULD similarly warn the user if they try to add a new User ID to only one member of an Identity Equivalence Group, and (if possible) offer to add it to all members instead.

This is a design limitation of the Key Replacement mechanism. Since Section 10.1 of [RFC9580] does not require a certificate to contain a User ID, it MUST still be possible to link an identity to it by other means. If we were to require matching User IDs for Identity Equivalence, the Replacement Key mechanism would not be fully functional for identity links that did not rely on User IDs.

## 5.2. Absence of an Identity Equivalence Binding

The Replacement Key subpacket MUST NOT be treated as a Web of Trust certification over either the current or replacement primary key. In the absence of an Identity Equivalence Binding, a receiving implementation SHOULD validate the replacement certificate as they would any other.

If an implementation supports the creation of unpaired Replacement Key subpackets, it SHOULD warn the key owner that correspondents may not be able to use the preferred certificate in the absence of other means of identity verification. For example, it could prompt the key owner to ask the third parties who certified the User ID(s) on the deprecated certificate to certify the corresponding User ID(s) on the replacement.

### 5.2.1. Limiting Chaining of Non-Equivalent Replacements

It is possible for a singly-linked chain of certificates to exist, where each (non-final) certificate contains a valid forward Replacement Key subpacket with no equivalence binding. Such a chain may terminate, or may form a closed loop. A generating implementation SHOULD NOT intentionally create singly-linked chains or loops, however they may be encountered in practice as a result of a stale cache, user or implementation error, or malice.

It may be possible for the certificates in a singly-linked chain to be validated without Identity Equivalence Bindings, such as by provenance or Web of Trust. In such a scenario, a receiving implementation might be induced to process the chain of references until it found the end, or a certificate that did not validate. An implementation SHOULD limit the maximum number of references it follows (per invocation), and/or implement a loop detection mechanism to prevent following a closed loop of references indefinitely.

For example, a receiving implementation MAY choose to process only one unpaired forward Key Replacement subpacket per invocation. Such an implementation MAY however process a subsequent unpaired subpacket on its next invocation, if the current certificate in the chain successfully validated. While this could result in unstable behaviour, where the apparent preferred certificate of the correspondent changes continually, each invocation will consume a finite amount of resources.

### 5.3. Reasons for Revocation

If a Key Revocation signature contains a Replacement Key subpacket, a Reason for Revocation subpacket MUST also be included, to prevent it from being interpreted as "No reason specified".

- \* if a Replacement Key subpacket is included in a Key Revocation signature, then the Reason For Revocation subpacket MUST indicate "Key is superseded".
- \* if no Replacement Key subpacket is included in a Key Revocation signature, but a Replacement Key might be specified at a later date, then the Reason For Revocation subpacket MAY indicate "Key is superseded".
- \* otherwise, the Reason for Revocation subpacket SHOULD indicate "Key is retired and no longer used", to explicitly state that the revoked primary key has no replacement.

A receiving implementation MUST ignore any Reason for Revocation subpacket present in a Key Revocation signature with a Reason for Revocation that is not "Key is superseded".

### 6. Selection of Encryption Subkeys

If a replacement certificate has been validated, whether as a member of an Identity Equivalence Set or otherwise, correspondents SHOULD assign it preference over the current certificate. When a correspondent of the key owner selects subkeys for encryption, the subkeys in the replacement certificate SHOULD therefore be considered first. If the subkeys in the replacement certificate are not usable, then:

- \* If there is an equivalence binding, the subkeys in the first listed deprecated certificate SHOULD be considered next. If the subkeys in the first deprecated certificate are not usable, then the subkeys in the next deprecated certificate (if any) SHOULD be considered, and so forth.
- \* If there is no equivalence binding, the subkeys in the current certificate SHOULD be used.

When encrypting to herself, the key owner MAY use a different encryption subkey selection algorithm from the one used for her correspondents.

## 7. Security Considerations

If the Replacement Key subpacket was allowed in the unhashed subpackets area, an attacker could add a bogus Replacement Key subpacket to an existing signature.

An Identity Equivalence Binding requires the active consent of both primary key owners. This is to prevent one key owner from unilaterally claiming signatures made by the other key owner, using the same argument that motivates the embedded Primary Key Binding signature in a signing-capable subkey's binding signature (Section 5.2.1.9 of [RFC9580]).

The Target Key Imprint is included to mitigate against weaknesses in the fingerprint digest algorithm used by older key versions. By including a digest over the key material in the target primary public key packet, using the same digest algorithm as the enclosing signature, we ensure that the indirect cryptographic binding between the equivalent keys is of the same overall strength as a signature made directly over the target primary public key (as in a certification signature or subkey binding signature). We intentionally chose not to use embedded back-signatures or third-party certifications, both to keep the design simple and to limit the size of the subpacket(s) required.

In the absence of a complete Identity Equivalence Binding, the Replacement Key subpacket is merely advisory. In this scenario, it provides information for the purposes of key discovery only, without any actionable statement about the User IDs on the replacement.

In addition, as this document is an update of [RFC9580], the security considerations there should be carefully reviewed.

## 8. IANA Considerations

This document requests that the following entry be added to the OpenPGP Signature Subpacket registry:

Type	Name	Specification
100 (TEMPORARY)	Replacement Key	This document

Table 3: Signature Subpacket Registry

## 9. References

## 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [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/rfc/rfc8174>>.
- [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/rfc/rfc9580>>.

## 9.2. Informative References

- [Autocrypt] Breitmoser, V., Krekel, H., and D. K. Gillmor, "Autocrypt - Convenient End-to-End Encryption for E-Mail", n.d., <<https://autocrypt.org/>>.
- [I-D.koch-openpgp-webkey-service] Koch, W., "OpenPGP Web Key Directory", Work in Progress, Internet-Draft, draft-koch-openpgp-webkey-service-20, 2 June 2025, <<https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-20>>.

## Appendix A. Example Workflows

In the following, Alice has a v4 primary keypair and subsequently generates a new v6 primary keypair, then at a later date she revokes her v4 primary key. Bob is her correspondent who consumes the updated certificate(s). We do not assume that Alice's secret keys are stored on the same hardware device.

### A.1. Alice Generates a New Primary Key

#### A.1.1. Alice's Actions

If Alice's client supports v6 certificates, and Alice only has a v4 certificate, it may suggest to Alice that she should generate a v6 primary keypair and certificate. If the secret key material for the v4 certificate is available, the client may offer to generate the new keypair and certificate automatically. The client should warn Alice of potential compatibility issues with any other devices that she may have which share the v4 secret key material. The client should also perform pre-flight checks that may include:



- \* Refreshing the certificate from the internet to check whether another client has already created a replacement.
- \* Attempting to sync with other devices in a multi-device deployment, e.g. by emailing itself.

On creation of a v6 primary keypair and certificate, Alice's client will:

- \* Add a Direct Key signature to Alice's new (v6) certificate, including a backward Replacement Key subpacket that references her deprecated v4 certificate.
- \* If the deprecated (v4) secret key is available, add a new Direct Key signature to its certificate with a forward Replacement Key subpacket that references the v6 replacement.
- \* Publish the updated certificate(s) to a keyserver.
- \* Back up the new secret key material and (if necessary) sync it to any other devices.

#### A.1.2. Bob's Actions

- \* Bob wants to send Alice a message and has Alice's deprecated (v4) certificate.
- \* Either Bob's copy of Alice's deprecated certificate already has a Replacement Key subpacket that references her replacement (v6) fingerprint, or Bob refreshes Alice's deprecated certificate from a keyserver and sees the new Replacement Key subpacket.
- \* If Bob has a v6 implementation, it can proceed with fetching Alice's v6 replacement certificate, validating it, and using it to send his message to Alice.
- \* If Bob doesn't have a v6 implementation, it can continue to use Alice's v4 deprecated certificate.

(WKD does not currently allow more than one valid certificate to be returned for a query, therefore it cannot easily support this use case.)

Bob's client has previously marked Alice's deprecated certificate as usable for her email address. If this was done directly (as opposed to via the Web of Trust), Bob's client may also directly mark her replacement certificate as usable.

### A.1.3. Alice's Second Device

Alice's client should perform regular consistency tests, by performing the pre-flight checks above to discover whether another client or device has published a different replacement certificate. In such a scenario, the client should warn the user and attempt to reconcile the differences by first syncing the secret key material, and then updating the older "replacement" certificate to be an additional deprecated certificate for the newer replacement, and finally updating the proper replacement certificate to refer to the additional deprecated certificate.

## A.2. Alice Revokes Her Deprecated Primary Key

### A.2.1. Alice's Actions

On revocation of her v4 primary key, Alice's client will:

- \* Add a forward Replacement Key subpacket to the Primary Key Revocation signature, referencing the replacement.
- \* If the replacement secret key is available, add a new Direct Key signature to its certificate with a new or updated backward Replacement Key subpacket that references the deprecated certificate.
- \* Publish the updated certificate(s) to a keyserver.

### A.2.2. Bob's Actions

Bob already has both Alice's v4 and v6 certificates.

- \* Bob's client refreshes Alice's certificates from a keyserver; her deprecated certificate contains a Primary Key Revocation signature with a Replacement Key subpacket but the preferred certificate is unchanged.
- \* There is an Identity Equivalence Binding between the deprecated and replacement certificates, which Bob's client automatically validates.
- \* Bob's v6-aware client continues to use Alice's preferred certificate as before.

Bob's client has previously marked Alice's deprecated certificate as usable for her email address. If it has not yet directly marked Alice's replacement certificate as usable, it should do so now to remove the requirement to validate the now-revoked v4 certificate.

### A.2.3. Carol's Actions

Carol wants to send Alice a message; Carol has Alice's deprecated (v4) certificate but they have not corresponded for some time.

- \* Carol's client refreshes Alice's deprecated certificate from a keyserver; it contains a Primary Key Revocation signature with a Replacement Key subpacket.
- \* Carol's client looks up Alice's replacement (v6) certificate on a keyserver.
- \* There is an Identity Equivalence Binding between the deprecated and replacement certificates, which Carol's client automatically validates.
- \* Carol's client uses Alice's replacement certificate instead of the deprecated certificate.

(There are other means to achieve a similar result, such as WKD [I-D.koch-openpgp-webkey-service] or [Autocrypt], but they may not be available. For example, Alice's service provider may not support WKD, and Alice may not have sent Carol an autocrypt message since revoking her deprecated primary key.)

Carol's client has previously marked Alice's deprecated certificate as usable for her email address. Carol's client should now directly mark Alice's replacement certificate as usable.

### A.2.4. Alice's Second Device

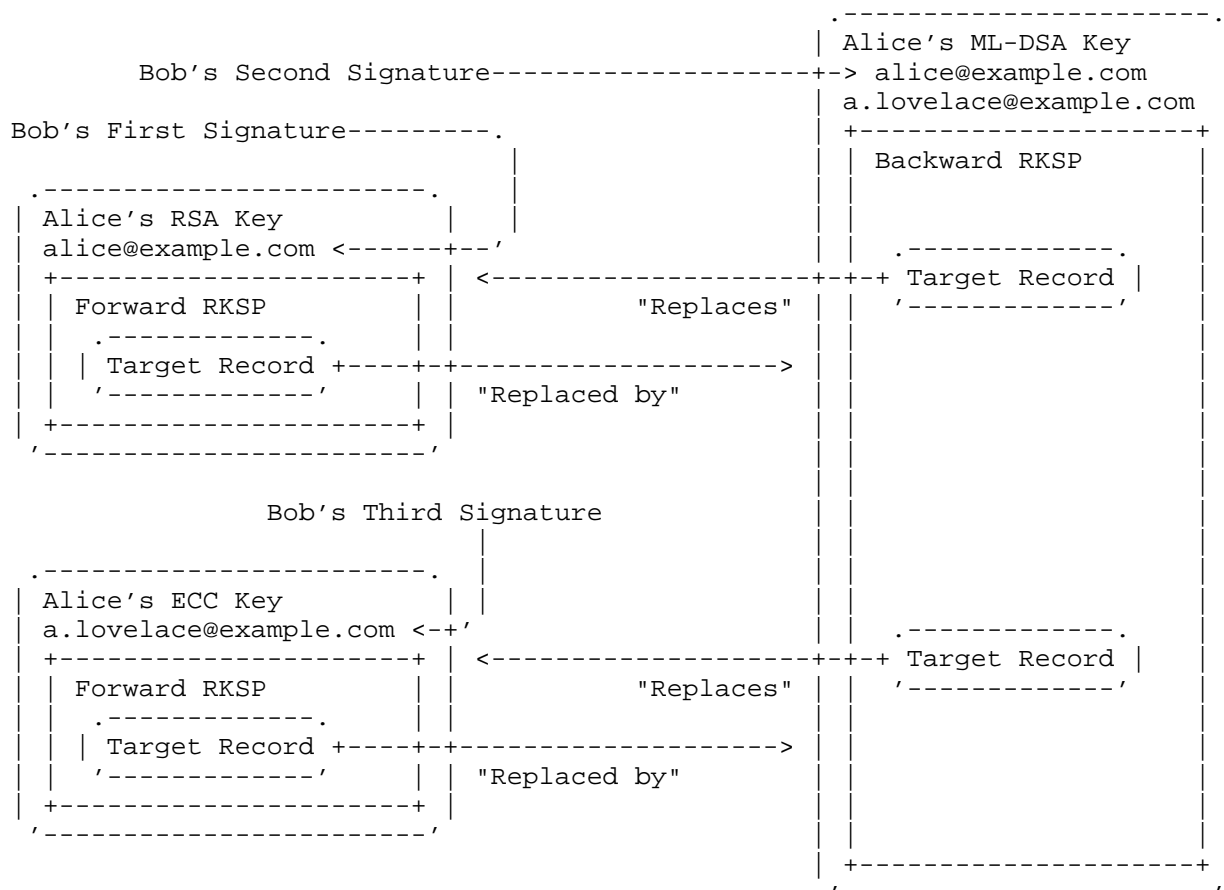
On receipt of the new (v6) certificate on a second device, Alice's second client will:

- \* Check to see if the certificate contains an unpaired backward Replacement Key subpacket that refers to any of the available secret keys.
- \* If the reference is correct and Alice consents, add a Direct Key signature to the affected certificate with a forward Replacement Key subpacket.
- \* Publish the updated certificate to a keyserver.

If Alice has a local encryption subkey that she prefers for encryption to herself, her client MAY use it regardless of any Replacement Key subpacket(s) in her published certificate.

### A.3. Time Evolution of an Identity Equivalence Group

Let's say that Alice has two deprecated keys (one RSA and one ECC), and a replacement key (ML-DSA). Let's also say that she has the identity `alice@example.com` on the RSA and ML-DSA keys, and the identity `a.lovelace@example.com` on the ECC and ML-DSA keys.



"RKSP" = Replacement Key Subpacket

Figure 2: Bob's View of Alice's Certificates

If Bob has made a certification signature (1) over `alice@example.com` and the RSA key, then from his point of view there is a direct identity link between `alice@example.com` and her RSA key, and a derived identity link (via equivalence) between it and both of the other two keys. These derived links are subsidiary to the direct link and do not have an independent existence - if Bob's

certification expires, the direct link is invalidated and by extension the derived links are also invalidated. If however Bob also made a certification signature (2) over Alice's ML-DSA key, then the expiration of the first certification would not invalidate the second, and all three keys would remain linked to `alice@example.com` - but now Bob's certification over the ML-DSA key would anchor the direct identity link, and the link between `alice@example.com` and her RSA key would now be equivalence-derived.

None of the above has any consequence for the identity `a.lovelace@example.com`. Bob knows of no proof that Alice owns that email address, and while her ECC certificate may claim `a.lovelace@example.com`, and we may therefore infer that the owner of the other certificates also claims `a.lovelace@example.com` (because equivalence implies they are the same person), there is no supporting evidence to elevate any of these claims to a concrete link.

But let's say that Bob then receives an email from `a.lovelace@example.com` with the ML-DSA certificate in its [Autocrypt] header. He MAY therefore infer that the identity `a.lovelace@example.com` is directly linked (via the TOFU principle) to the ML-DSA certificate - and thereby also to the other two certificates. This TOFU-based direct identity link may be considered "weaker" than one based on a certification signature, and so the derived identity links on the other two certificates SHOULD be considered equally "weak". If Bob subsequently certifies (3) the identity `a.lovelace@example.com` on the ECC key (for example), all the derived links would become subsidiary to that link, and also become equally "strong".

## Appendix B. Acknowledgments

The authors would like to thank Bart Butler, Kai Engert, Daniel Kahn Gillmor, Daniel Huigens, Simon Josefsson, Johannes Roth, Heiko Schfer, Falko Strenzke, Neal Walfield, Justus Winter and Aron Wussler for suggestions and discussions.

## Appendix C. Document History

Note to RFC Editor: this section should be removed before publication.

### C.1. Changes Between draft-ietf-openpgp-replacementkey-05 and -06

- \* Specified "current" Replacement Key subpacket.
- \* Record length is now one octet again.

- \* Removed references to "hard" and "soft" revocation reasons, and tightened BCP14 language around reasons.
- \* Cleaned up artwork.
- \* Improved example workflows.
- \* Various minor clarifications.

#### C.2. Changes Between draft-ietf-openpgp-replacementkey-04 and -05

- \* Key Equivalence is now Identity Equivalence.
- \* Specified Identity Equivalence Sets.
- \* Defined terms "identity", "identity claim", "identity link", "direct link", "derived link".
- \* Moved 0x40 to 0x01 and renamed from "inverse relationship" to "backward reference(s)".
- \* Substituted "original" with "deprecated".
- \* Expanded and improved "Terminology" section.
- \* Merged content of "Placement of the Replacement Key Subpacket" section into other sections.
- \* Removed reference to draft-ietf-openpgp-persistent-symmetric-keys (but kept the surrounding text).
- \* Renamed and reordered some sections.
- \* Added receiving implementation MUSTs in several places.
- \* Various minor clarifications.

#### C.3. Changes Between draft-ietf-openpgp-replacementkey-03 and -04

- \* Made target records forward-compatible.
- \* Added additional guidance for Alice's client.
- \* Removed unnecessary reference to WKD.
- \* Removed requirement for unilateral reference to be treated as a preference.

- \* Modified wording to avoid incompatibility with future encryption subkey selection draft.

#### C.4. Changes Between draft-ietf-openpgp-replacementkey-02 and -03

- \* Added section clarifying time evolution of equivalence.
- \* Added section clarifying how to prevent resource exhaustion when following chains.
- \* Clarified description of equivalence transitivity.

#### C.5. Changes Between draft-ietf-openpgp-replacementkey-01 and -02

- \* Added explanation of hard vs soft revocations.
- \* Remove the "No Replacement" bit and use the Reason for Revocation subpacket instead.
- \* Record length field is now two octets.
- \* Inverted treatment of undefined flag bits.
- \* Remove references to the Preferred Key Server subpacket.
- \* Expanded example workflows section and add reference to draft-ietf-openpgp-persistent-symmetric-keys.
- \* Various terminology nitpicking.

#### C.6. Changes Between draft-ietf-openpgp-replacementkey-00 and -01

- \* Updated references to RFC9580.
- \* Removed subpacket version octet.
- \* Added guidance for encryption subkey selection.
- \* Added record length fields.
- \* Renamed to "OpenPGP Key Replacement" and normalised terminology.

#### C.7. Changes Between draft-gallagher-openpgp-replacementkey-02 and draft-ietf-openpgp-replacementkey-00

- \* Standardised capitalisation and terminology.

## C.8. Changes Between draft-gallagher-openpgp-replacementkey-01 and -02

- \* Specified Public Key Imprints.
- \* Specified inverse relationship flag and packet format.
- \* Restricted graph topology.
- \* Specified Identity Equivalence Binding.
- \* Guidance re subpacket placement escalated from SHOULD to MUST, and critical bit to SHOULD NOT.

## C.9. Changes Between draft-gallagher-openpgp-replacementkey-00 and -01

- \* Added example workflows.
- \* Specifically describe "deprecation without expiry or revocation" use case.
- \* Add note about weakness of signatures over fingerprints.
- \* Miscellaneous clarifications.

## C.10. Changes Between draft-shaw-openpgp-replacementkey-00 and draft-gallagher-openpgp-replacementkey-00

- \* Changed algid octet to key version octet.
- \* Changed initial subpacket version number to 1.
- \* Clarified semantics of some edge cases.

## Authors' Addresses

Daphne Shaw  
Jabberwocky Tech  
Email: dshaw@jabberwocky.com

Andrew Gallagher (editor)  
PGPKeys.EU  
Email: andrewg@andrewg.com