

openpgp
Internet-Draft
Updates: 9580 (if approved)
Intended status: Standards Track
Expires: 21 January 2026

D. Shaw
Jabberwocky Tech
A. Gallagher, Ed.
PGPKeys.EU
20 July 2025

OpenPGP Key Replacement
draft-ietf-openpgp-replacementkey-04

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 21 January 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	4
4. Format of the Replacement Key Subpacket	5
4.1. Key Imprints	7
4.2. Graph Topology	7
5. Trust and Validation of the Replacement Key Subpacket	7
5.1. Key Equivalence Binding	8
5.1.1. Reasons for Revocation	8
5.1.2. Time Evolution of Key Equivalence	9
5.2. Without a Key Equivalence Binding	9
5.2.1. Limiting Chaining of Non-Equivalent Replacements	10
6. Selection of Encryption Subkeys	10
7. Placement of the Replacement Key Subpacket	11
8. Security Considerations	11
9. IANA Considerations	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Appendix A. Example Workflows	12
A.1. Alice's Actions	13
A.2. Bob's Actions	14
Appendix B. Acknowledgments	15
Appendix C. Document History	15
C.1. Changes Between draft-ietf-openpgp-replacementkey-03 and -04	15
C.2. Changes Between draft-ietf-openpgp-replacementkey-02 and -03	16
C.3. Changes Between draft-ietf-openpgp-replacementkey-01 and -02	16

C.4.	Changes Between draft-ietf-openpgp-replacementkey-00 and -01	16
C.5.	Changes Between draft-gallagher-openpgp-replacementkey-02 and draft-ietf-openpgp-replacementkey-00	16
C.6.	Changes Between draft-gallagher-openpgp-replacementkey-01 and -02	17
C.7.	Changes Between draft-gallagher-openpgp-replacementkey-00 and -01	17
C.8.	Changes Between draft-shaw-openpgp-replacementkey-00 and draft-gallagher-openpgp-replacementkey-00	17
	Authors' Addresses	17

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 primary key that is signed over, or a primary key for which the current primary key is the suggested replacement. The corresponding replacement certificate may then be automatically retrieved by the key owner's correspondents and (if supported and validated) used instead of the original.

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 historically been used loosely to refer to several distinct concepts. Care is therefore required when talking about "keys" in a non-specific sense. In this document, we use the following convention:

- * "replacement primary key" and "original primary key" refer to a primary key as contained in a Transferable Public Key (certificate) or Transferable Secret Key.
- * "target key" refers to either a replacement or original primary key that is referred to by a record in a Replacement Key subpacket.
- * "current primary key" refers to the primary key that the self-signature currently under discussion belongs to.
- * "replacement certificate", "original certificate" and "current certificate" refer to the TPK within which the corresponding primary key is distributed.

3. The Replacement Key Subpacket

The Replacement Key subpacket is a Signature Subpacket (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 (insert this later).

To explicitly state that a revoked primary key has no replacement, a Reason for Revocation of "Key is retired and no longer used" SHOULD be used (see Section 5.1.1). The absence of a Replacement Key subpacket SHOULD NOT be interpreted as meaning that there is no replacement (or original) for the current primary key.

The Replacement Key subpacket MUST only be used in the hashed subpackets area of a primary key revocation or direct key signature.

4. Format of the Replacement Key Subpacket

The format of the Replacement Key subpacket is:

Octets	Field	Notes
1	Class	
2	Record length (1)	
1	Target Key Version (1)	
N1	Target Key Fingerprint (1)	
M	Target Key Imprint (1)	
2	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
0x40	Inverse relationship	Multiple targets may be given

Table 2: Replacement Key Subpacket Flags

The 0x40 bit of the class octet is the "inverse relationship" 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.

All other flag bits of the class octet are currently undefined. All undefined flags MUST be zero. 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 big-endian two-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 0x40 bit set, the subpacket MUST contain exactly one target record to identify the replacement primary key.
- * If the class octet has the 0x40 bit set, the subpacket contains one or more target records, to identify the original primary key(s) that the current primary key is a replacement for.

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 the intent is to state that the replacement (or original) primary key is unknown, then no Replacement Key subpacket should 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.

4.2. Graph Topology

A given signature MUST contain at most one Replacement Key subpacket. If a signature contains more than one such subpacket, a receiving implementation MUST disregard them all. This imposes a simple graph topology:

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

In addition, the order of the original primary keys specified in an inverse-relationship 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 original primary keys in its inverse Replacement Key subpacket (if one exists) to indicate which original primary key is preferred as a fallback. The original primary keys SHOULD therefore be listed in order of decreasing preference.

5. Trust and Validation of the Replacement Key Subpacket

5.1. Key Equivalence Binding

The existence of a matching pair of forward and inverse 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 a Key Equivalence Binding. If one primary key is validated for use in a particular context, then any primary key that has a Key Equivalence Binding with it (together with any bound subkeys) is also valid, regardless of any User ID certifications over the second primary key (or lack thereof).

The equivalence binding is invalidated under the following circumstances:

- * if either primary key is hard-revoked.
- * 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.
- * if either signature that forms the equivalence binding has expired.

Note however:

- * If either primary key is soft-revoked or expired, the equivalence binding is unaffected.
- * If either primary key is hard-revoked, then the equivalence binding is invalidated and the other key is unaffected.
- * Other properties (such as expiry dates, usage preferences, custom notations) SHOULD NOT be applied across the equivalence binding.
- * Key Equivalence is transitive; if A and B are both equivalent to C (e.g. if C replaces both A and B), then A is also equivalent to B.

5.1.1. Reasons for Revocation

For the purposes of Key Revocation signatures:

- * "hard-revoked" means the key has been revoked with a Reason for Revocation subpacket specifying "Key material has been compromised" or "No reason specified"; the absence of a Reason for Revocation subpacket is equivalent to "No reason specified".

- * "soft-revoked" means the key has been revoked with a Reason for Revocation subpacket specifying "Key is superseded" or "Key is retired and no longer used".

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", which is a hard revocation.

- * if a Replacement Key subpacket is included in a revocation signature, then the Reason For Revocation subpacket SHOULD indicate "Key is superseded".
- * if no Replacement Key subpacket is included in a 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".

If two or more primary keys have Key Equivalence Binding(s) between them, they MUST be treated as a single key for the purposes of the Web of Trust, particularly when calculating partial trust values.

5.1.2. Time Evolution of Key Equivalence

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

It is therefore RECOMMENDED that applications attempt to find alternative trust pathways for replacement certificates. The optimal method of obtaining alternative trust 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 WoT.

5.2. Without a Key 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 a Key Equivalence Binding, a receiving implementation SHOULD validate the replacement certificate as they would any other.

It is also suggested that the key owner asks the third parties who certified the User ID(s) on the original 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 forwards 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 Key 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 forwards 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.

6. Selection of Encryption Subkeys

If a replacement certificate has been validated, whether through key equivalence or other means, 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 original certificate SHOULD be considered next. If the subkeys in the first original certificate are not usable, then the subkeys in the next original 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 is not required to use the same encryption subkey selection algorithm as her correspondents.

7. Placement of the Replacement Key Subpacket

The Replacement Key subpacket is only meaningful on a primary key revocation or direct key signature, and MUST NOT appear elsewhere. The Replacement Key subpacket MUST be placed in the hashed subpackets area of the signature to prevent a possible key substitution attack. 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.

8. Security Considerations

A Key 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.

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 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 Key Equivalence Binding, the Replacement Key subpacket MUST be treated as merely advisory. In this scenario, it provides information for the purposes of key discovery only, without any trust statement regarding the replacement.

Implementations SHOULD NOT infer any trust value from a single Replacement Key subpacket, and SHOULD validate the replacement certificate as they would any other.

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

9. IANA Considerations

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

Type	Name	Specification
TBC	Replacement Key	This document

Table 3: Signature Subpacket Registry

10. References

10.1. 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>>.

10.2. Informative References

- ```
[I-D.ietf-openpgp-persistent-symmetric-keys]
 Huigens, D., "Persistent Symmetric Keys in OpenPGP", Work
 in Progress, Internet-Draft, draft-ietf-openpgp-
 persistent-symmetric-keys-01, 30 January 2025,
 <https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-
 persistent-symmetric-keys-01>.
```

## 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'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 an inverse Replacement Key subpacket that references her v4 original.
- \* If the original (v4) secret key is available, add a new Direct Key signature to its certificate with a forwards 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.

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 original for the newer replacement, and finally updating the proper replacement certificate to refer to the additional original.

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

- \* Add a forwards 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) inverse Replacement Key subpacket that references the original.
- \* Publish the updated certificate(s) to a keyserver.

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 inverse Replacement Key subpacket that refers to any of the available secret keys.
- \* If the reference is correct, add a Direct Key signature to the affected certificate with a forwards Replacement Key subpacket.
- \* Publish the updated certificate to a keyserver.

If Alice has a local encryption subkey that she prefers for encryption to herself, for example a Persistent Symmetric subkey [I-D.ietf-openpgp-persistent-symmetric-keys], her client SHOULD use it regardless of any Replacement Key subpacket(s) in her published certificate.

#### A.2. Bob's Actions

If Alice has revoked her original Primary Key:

- \* Bob wants to send Alice a message; Bob has Alice's original (v4) certificate but they have not corresponded for some time.
- \* Bob's client refreshes Alice's original certificate from a keyserver; it contains a Primary Key Revocation signature with a Replacement Key subpacket.
- \* Bob's client looks up Alice's replacement (v6) certificate on a keyserver.
- \* There is a Key Equivalence Binding between the original and replacement certificates, which Bob's client automatically validates.
- \* Bob's client uses Alice's replacement certificate instead of the original certificate.

There are other means to achieve a similar result, such as WKD or Autocrypt, but they may not be available. For example, Alice's service provider may not support WKD, and Alice may not have sent Bob an autocrypt message since revoking her original primary key.

If Alice has not revoked her original Primary Key:

- \* Bob wants to send Alice a message and has Alice's original (v4) certificate.
- \* Either Bob's copy of Alice's original certificate already has a Replacement Key subpacket that references her replacement (v6) fingerprint, or Bob refreshes Alice's original 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 original 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.

## Appendix B. Acknowledgments

The authors would like to thank Bart Butler, Kai Engert, Daniel Kahn Gillmor, Daniel Huigens, Simon Josefsson, Johannes Roth, Heiko Schaefer, 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-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.2. 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.3. 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.4. 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.5. Changes Between draft-gallagher-openpgp-replacementkey-02 and draft-ietf-openpgp-replacementkey-00

- \* Standardised capitalisation and terminology.



## C.6. Changes Between draft-gallagher-openpgp-replacementkey-01 and -02

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

## C.7. 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.8. 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