openpgp
Internet-Draft
Updates: 9580 (if approved)
Intended status: Standards Track
Expires: 6 December 2026

A. Gallagher, Ed.
PGPKeys.EU

4 June 2026

OpenPGP Signatures
draft-gallagher-openpgp-signatures-03

Abstract

This document specifies several updates and clarifications to the grammar and semantics of OpenPGP signatures.

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-signatures. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-gallagher-openpgp-signatures/.

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-signatures.

Status of This Memo

Table of Contents

1. Introduction

OpenPGP signatures have a rich vocabulary, however this is often under-specified. This document attempts to address this by:

* Expanding on specifications where [RFC9580] does not fully describe the existing or expected behaviour of deployed implementations.

* Adding clarification where deployed implementations differ in their interpretation of [RFC9580] and its predecessors.

* Deprecating unused or error-prone features.

This document does not specify any new wire formats.

2. Conventions and Definitions

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

The term "Component key" is used in this document to mean either a primary key or subkey.

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.

3. Signature Types

Several signature types are specified in incomplete, confusing or contradictory ways. We update their specifications as follows.

3.1. Timestamp Signature (0x40)

Section 6.2.1 of [RFC1991] defined the Timestamp signature as:

> <40> - time stamping ("I saw this document")
>
> Type <40> is intended to be a signature of a signature, as a notary seal on a signed document.

The second statement implies that a v3 0x40 sig is made over a signature packet. But the first statement implies a signature over a document, just with different semantics.

By Section 5.2.1.14 of [RFC9580], this has changed to:

> 0x40: Timestamp signature. This signature is only meaningful for the timestamp contained in it.

This avoids the apparent contradiction of [RFC1991], but is less informative. And there is no explicit construction given in Section 5.2.4 of [RFC9580].

We note also that [RFC9580] introduced a Key Flag for timestamping. This indicates that timestamping documents is sufficiently different from signing them that separate keys should be used. This is consistent with the idea that "I wrote this document" and "I saw this document" are distinct statements with different consequences. This is crucial in the case of an automated timestamping service that makes no claims about the accuracy of document contents.

We define type 0x40 Timestamp signatures as follows:

> A type 0x40 Timestamp signature is made over a Literal Data Packet. It is constructed and distributed in the same way as a type 0x00 Binary Document Signature. If the message is a text document, it MUST already be in Canonical Text form. By default a Timestamp signature conveys no opinion about the validity of the document; it only claims that the document existed at the timestamp of signature creation. This interpretation MAY be modified by adding notation subpackets, the meaning of which are application-dependent. It can be made over an otherwise unsigned document, or it can be one of many signatures over the same document. The Cleartext Signature Framework MUST NOT be used with Timestamp signatures.

Countersigning a Signature packet only (including blind countersigning) is done using the type 0x50 Third-Party Confirmation signature.

3.2. Third-Party Confirmation Signature (0x50)

Section 5.2.1.15 of [RFC9580] defines a Third-Party Confirmation signature as:

> This signature is a signature over some other OpenPGP Signature packet(s). It is analogous to a notary seal on the signed data. A Third-Party Confirmation signature SHOULD include a Signature Target subpacket that identifies the confirmed signature.

A concrete construction is provided, but the placement and semantics are still not well-defined. We clarify these as follows:

> By default, a Third-Party Confirmation signature makes no claim about the validity of the other signature, just its existence, and makes no claim whatsoever about the subject of that signature. This interpretation MAY be modified by adding notation subpackets, the meaning of which are application-dependent. It SHOULD be distributed in an Embedded Signature subpacket in the unhashed area of the signature it notarizes. A Signature Target subpacket is therefore unnecessary and SHOULD NOT be included.

See Section 4.4 for further discussion of the Signature Target subpacket.

3.2.1. Terminology Subtleties

Implementers should note that "Third-Party Confirmation" signatures (type 0x50) are distinct from "third-party Certification" signatures (types 0x10..0x13 when issued by a primary key other than the one signed over), and beware that older RFCs do not always use sufficiently precise terminology to distinguish them.

4. Signature Packets

Receiving implementations currently have insufficient guidance for when to reject non-idiomatic signature packets.

4.1. Recursive Embedding Inside Signature Subpackets

Section 5.2.3 of [RFC9580] specifies two subpackets which could recursively include a signature inside a signature:

* Embedded Signature (type 32): contains a signature packet

* Key Block (type 38, experimental): contains an entire certificate, which may itself include signature packets

In order to prevent excessive recursion via nested signature subpackets:

* Signatures contained within Embedded Signature subpackets MUST NOT contain any Embedded Signature subpackets:

  - An Embedded Signature subpacket MUST contain a signature of an Embeddable signature type.

  - An Embeddable signature MUST NOT contain Embedded Signature subpackets.

  - Initially, only the Primary Key Binding and Third-Party Confirmation signature types are specified as Embeddable.

* A signature of a type other than in the Literal Data Signature Category (Section 5) MUST NOT contain a Key Block subpacket.

A receiving implementation MUST invalidate any signature that does not conform to the above guidance.

4.2. Subpackets with Conflicting Information

Section 5.2.3.9 of [RFC9580] gives a receiving implementation significant leeway in interpreting conflicting combinations of subpackets:

> It is certainly possible for a signature to contain conflicting information in subpackets. For example, a signature may contain multiple copies of a preference or multiple expiration times. In most cases, an implementation SHOULD use the last subpacket in the hashed section of the signature, but it MAY use any conflict resolution scheme that makes more sense.

We hereby tighten this guidance:

> A signature MUST NOT contain more than one subpacket of any given type in its hashed subpackets area, unless otherwise specified. A receiving implementation MUST invalidate a signature that contains in its hashed area more than one subpacket of any type for which this is not explicitly permitted.

Multiple copies of the following subpacket types are already explicitly permitted:

* Issuer Key ID (type 16) Section 5.2.3.9 of [RFC9580]
* Notation Data (type 20) Section 5.2.3.24 of [RFC9580]
* Intended Recipient Fingerprint (type 35) Section 5.2.3.36 of [RFC9580]

In addition, the following interpretations are natural extensions of specified behaviour, and hereby permitted:

4.2.1. Multiple Revocation Key (type 12) Subpackets

If multiple Revocation Key subpackets are present, any of the listed keys MAY generate key revocation signatures.

4.2.2. Multiple Preferred Key Server (type 24) Subpackets

If multiple Preferred Key Server subpackets are present, updates MAY be obtained from any of the listed keyservers.

4.2.3. Multiple Embedded Signature (type 32) Subpackets

If multiple Embedded Signature subpackets are present, a receiving implementation MAY attempt to process them all in turn.

4.2.4. Multiple Issuer Fingerprint (type 33) Subpackets

Multiple Issuer Fingerprint subpackets are permitted, with the same interpretation as multiple Issuer Key ID subpackets.

Note however that Section 5.2.3.35 of [RFC9580] states of the Issuer Fingerprint subpacket:

> If the version of the issuing key is 4 and an Issuer Key ID subpacket (Section 5.2.3.12) is also included in the signature, the Key ID of the Issuer Key ID subpacket MUST match the low 64 bits of the fingerprint.

Generalizing to multiple subpackets, we replace this with:

> If both Issuer Key ID and Issuer Fingerprint subpackets are included in a signature then each Issuer Key ID subpacket MUST match the low 64 bits of only one v4 Issuer Fingerprint subpacket, and all v4 Issuer Fingerprint subpackets MUST have a corresponding Key ID subpacket.

4.2.5. Multiple Key Block (type 38, experimental) Subpackets

If multiple Key Block subpackets are present, a receiving implementation MAY attempt to process them all in turn.

4.3. Deprecation of the "Revocable" Signature Subpacket (type 7)

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

4.3.1. 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, "revocability" is only applicable to self-signatures and third party certifications.

* If a key is compromised, then the timestamp on any signature can be trivially backdated, so the timestamps on any signatures cannot be relied upon. Since "revocability" only concerns "later revocation signatures", it is only meaningful for signatures made 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 typically have a creation time significantly in the past compared to when it is 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.

4.4. Deprecation of the Signature Target Subpacket (type 31)

The Signature Target subpacket (Section 5.2.3.33 of [RFC9580]) fulfils the following roles:

* In a Timestamp or Third-Party Confirmation signature, it identifies the signature that is being countersigned

* In a Revocation signature, it identifies the signature being revoked

It is imprecisely defined and does not uniquely identify a particular Signature subpacket, and so when used as described is effectively non-functional. It is hereby deprecated.

4.4.1. Non-functionality of the "Signature Target" Signature Subpacket

The Signature Target subpacket is imprecisely defined:

> (1 octet public key algorithm, 1 octet hash algorithm, N octets hash)
>
> This subpacket identifies a specific target signature to which a signature refers. For Revocation Signatures, this subpacket provides explicit designation of which signature is being revoked. For a Third-Party Confirmation or Timestamp signature, this designates what signature is signed. All arguments are an identifier of that target signature.
>
> The N octets of hash data MUST be the size of the signature's hash. For example, a target signature with a SHA-1 hash MUST have 20 octets of hash data.

It is unclear from this text alone whether the hash field refers to the digest that the target signature is made over, or a digest over the resulting signature packet. The definition of a Third-Party Confirmation signature in Section 5.2.1 of [RFC4880] gives us a hint however:

> A third-party signature SHOULD include Signature Target subpacket(s) to give easy identification. Note that we really do mean SHOULD. There are plausible uses for this (such as a blind party that only sees the signature, not the key or source document) that cannot include a target subpacket.

(Beware that "third-party signature" in the above should be read as "Third-Party Confirmation signature"; see Section 3.2.1.)

The only way that a blind party would be unable to generate a Signature Target subpacket is if the hash is the digest that the original signature was made over. But if so it is not a unique identifier of a signature packet, since multiple distinct signatures can be made over the exact same material, including subpackets. In particular, if there was no Issuer Key ID or Issuer Fingerprint subpacket in the target signature's hashed area, a Signature Target subpacket could not distinguish between the original signature or an otherwise valid one issued by a completely different signing key.

The Signature Target subpacket is therefore not functional when used in a Third-Party Confirmation signature. A more reliable mechanism for identifying the target of a Third-Party Confirmation signature is to include it an Embedded Signature subpacket, directly in the unhashed area of the signature being countersigned.

The other specified use for the Signature Target subpacket is in a revocation signature. Certification Revocations are customarily understood to mean "I retract all my previous statements that this

key is related to this user" (Section 6.2.1 of [RFC1991]), so a Certification Revocation is not specific to any particular Certification signature. No other signature types can be revoked - primary key and subkey revocation signatures revoke the key, not the previous binding signature(s), and so are not specific to any particular binding signatures either.

The Signature Target subpacket is therefore not functional when used in a revocation signature. It is normally sufficient to distribute a revocation signature in an OpenPGP certificate or revocation certificate.

Timestamp signatures as specified in Section 3.1 do not require a Signature Target subpacket, since the signed message grammar identifies the material being signed over.

We therefore deprecate the Signature Target subpacket in all contexts.

5. Signature Categories

Signature Type code points are spaced out into identifiable ranges of types with similar semantics. These also mostly correspond to the various Key Flags. These ranges and their mapping to the Key Flags are not specified in [RFC9580].

We define Signature Categories to cover each range of type values:

* Literal Data Signature Category (0x00..0x07)
  - 0x00 Signature over a Binary document
  - 0x01 Signature over a Canonical Text document
  - 0x02 Standalone signature
* Unassigned (0x08..0x0F)
* Certification Category (0x10..0x17)
  - 0x10 Generic certification
  - 0x11 Persona certification
  - 0x12 Casual certification
  - 0x13 Positive certification

- (0x16 Approved certifications)

* Key Binding Category (0x18..0x1F)
  - 0x18 Subkey binding
  - 0x19 Primary key binding
  - 0x1F Direct key
* Primary Key Revocation Category (0x20..0x27)
  - 0x20 Primary Key revocation
* Subkey Revocation Category (0x28..0x2F)
  - 0x28 Subkey revocation
* Certification Revocation Category (0x30..0x37)
  - 0x30 Certification revocation
* Unassigned (0x38..0x3F)
* Timestamping Category (0x40..0x47)
  - 0x40 Timestamp
* Unassigned (0x48..0x4F)
* Countersignature Category (0x50..0x57)
  - 0x50 Third-Party confirmation
* Unassigned (0x58..0x5F)
* Private and Experimental Range (0x60..0x6F)
* Unassigned (0x70..0xFE)
* RESERVED (0xFF)

The Standalone signature type is effectively a "signature over an empty document". The Direct Key signature type contains key usage preference subpackets, similar to the Subkey Binding signature type, and is therefore effectively a "self-binding signature".

We have defined a Private and Experimental signature type range. This is 0x60..0x6F (96..111) for consistency with the existing private and experimental range in other registries. It does not form a Category and does not have a corresponding Key Flag.

Self-certifications over v4 Primary User IDs are used to convey the same information as Key Binding signatures. Therefore, unless specifically stated otherwise, any stipulations that apply to Key Binding signatures also apply to self-certifications over v4 Primary User IDs.

5.1. Key Flags

A Key Flags subpacket SHOULD be included in a Direct Key or Subkey Binding signature (or for v4 keys, a self-certification over the primary User ID). It applies only to a single key material packet; for a Direct Key signature (or primary User ID self-cert) it applies to the primary key only, and for a Subkey Binding signature, it applies only to that subkey.

Previously, it was also specified for use in third-party Certification Signatures. This is not widely supported and is hereby deprecated.

The following Key Flags permit the creation of signatures in one or more Signature Categories:

* 0x01.. Third-party signatures in the Certification and Certification Revocation Categories

* 0x02.. Literal Data Signature Category

* 0x0008.. Timestamping Category

* ((TBC)) Countersignature Category

The following exceptional usages are always permitted regardless of Key Flags:

* Primary keys are always permitted to make self-signatures in the Certification, Key Binding, Certification Revocation, Primary Key Revocation and Subkey Revocation Categories.

* Subkeys with signing-capable algorithms are always permitted to make Primary Key Binding signatures.

* Any key with a signing-capable algorithm is permitted to make a signature in the Private and Experimental range.

Otherwise:

* A signature made by a key that does not have the corresponding Key Flag MUST be considered invalid.

* A key with no Key Flags subpacket MUST NOT create signatures.

Section 5.2.1.10 of [RFC9580] also explicitly allows keys with the 0x01 Key Flag to create third-party 0x1F Direct Key Signatures. These are used for trust delegation in [SQ-WOT].

5.2. Authentication Signatures

OpenPGP defines no authentication signature types, but does have an authentication Key Flag. Traditionally, authentication is performed by converting the key material into that of another protocol (usually OpenSSH) and performing authentication in that protocol.

Beware that cross-protocol usage can be exploited to evade the domain separation protections of Key Flags. For example, there is no distinction between document signing, certification and authentication usage in OpenSSH, and once converted an OpenPGP authentication key may be used as a OpenSSH CA or to sign git commits.

((TODO: Guidance for the use of authentication keys should be provided. #12))

6. Signature Subpacket Categories

Signature subpacket types may also be categorized, depending on where they are used:

6.1. General subpackets.

These may be attached to any signature type, and define properties of the signature itself. Some of these subpackets are self-verifying (SV), i.e. they contain hints to locate the issuing key that can be confirmed after the fact. It MAY be reasonable to place self-verifying general subpackets in the unhashed area. All other general subpackets MUST be placed in the hashed area.

Subpacket types: Signature Creation Time, Signature Expiration Time, Issuer Key ID (SV), Notation Data, Signer’s User ID, Issuer Fingerprint (SV).

(Notation subpackets are categorized here as general subpackets, however the notations within them may have arbitrary semantics at the application layer)

TODO: should Signature Expiration Time subpackets be more restricted, e.g. to certifications? (issue #8)

6.2. Context subpackets.

These have semantics that are meaningful only when used in signatures of a particular type or category:

6.2.1. Direct subpackets.

These are normally only meaningful in a direct self-sig (or for v4 keys, a self-cert over the primary User ID) and define usage preferences for the certificate as a whole. They MAY be used in self-certs over other User IDs, in which case they define usage preferences for just that User ID (but this is not always meaningful or universally supported). They SHOULD NOT be used elsewhere. They MUST be placed in the hashed area.

A Direct subpacket MUST be ignored if it is in a self-cert made over a User ID by a v6 or later primary key.

Subpacket types: Preferred Symmetric Ciphers, Revocation Key (deprecated), Preferred Hash Algorithms, Preferred Compression Algorithms, Key Server Preferences, Preferred Key Server, Features, (Preferred AEAD Algorithms), Preferred AEAD Ciphersuites, Replacement Key.

The Replacement Key subpacket MAY also be used as a key revocation subpacket.

6.2.2. Revocation subpackets.

These are only meaningful in signatures of the Key Revocation, Subkey Revocation or Certificate Revocation categories. They SHOULD NOT be used elsewhere. They MUST be placed in the hashed area.

Subpacket types: Reason for Revocation, Replacement Key (Primary Key Revocations only).

6.2.3. Key Binding subpackets.

These are only meaningful in a signature of the Key Binding category (or for v4 keys, a self-cert over the primary User ID) and define properties of that particular component key. They SHOULD NOT be used elsewhere. They MUST be placed in the hashed area.

A Key Binding subpacket MUST be ignored if it is in a self-cert over a User ID that is not currently the primary User ID, or in a self-cert made over a User ID by a v6 or later primary key.

Subpacket types: Key Expiration Time, Key Flags.

6.2.4. First-party Certification subpackets.

These are only meaningful in a self-certification over a User ID, and define properties of that User ID. They SHOULD NOT be used elsewhere. They MUST be placed in the hashed area.

Subpacket types: Primary User ID

6.2.5. Third-party Certification subpackets.

These are only meaningful in third-party certification signatures and define properties of the Web of Trust. They SHOULD NOT be used elsewhere. They MUST be placed in the hashed area.

Subpacket types: Exportable Certification, Trust Signature, Regular Expression, Revocable, Policy URI, (Trust Alias).

6.2.6. Literal Data subpackets.

These are only meaningful in signatures of the Literal Data category, and define properties of the document or message. They SHOULD NOT be used elsewhere. Some of these subpackets are self-verifying (SV) and MAY be placed in the unhashed area. All other Literal Data subpackets MUST be placed in the hashed area. (Beware that the usefulness of all of these subpackets has been questioned)

Subpacket types: Intended Recipient Fingerprint, (Key Block (SV)), (Literal Data Metadata).

6.2.7. Attribute Value subpackets.

These are only meaningful in signature types whose specification explicitly requires them. They SHOULD NOT be used elsewhere. It MAY be reasonable to place Embedded Signature subpackets in the unhashed area. All other Attribute Value subpackets MUST be placed in the hashed area. They have no intrinsic semantics; all semantics are defined by the enclosing signature.

Subpacket types: Signature Target, Embedded Signature, (Delegated Revoker), (Approved Certifications).

6.3. Subpackets summary

Type	Name	Category	Critical	Unhashed	Context	Notes
0	Reserved	-				never used
1	Reserved	-				never used
2	Signature Creation Time	General	SHOULD			MUST always be present in hashed area
3	Signature Expiration Time	General	SHOULD			
4	Exportable Certification	Third-Party	MUST IFF false			boolean, default true
5	Trust Signature	Third-Party				
6	Regular Expression	Third-Party	SHOULD			
7	Revocable (deprecated)	Third-Party				boolean, default false (Section 4.3)

8	Reserved	-				never used
9	Key Expiration Time	Key Binding	SHOULD			
10	Placeholder for backwards compatibility	-				PGP.com proprietary feature

11	Preferred Symmetric Ciphers	Direct				
12	Revocation Key (deprecated)	Direct				
13-15	Reserved	-				never used
16	Issuer Key ID	General		MAY		issuer fingerprint is preferred
17-19	Reserved	-				never used
20	Notation Data	General				notations may be further classified
21	Preferred Hash Algorithms	Direct				
22	Preferred Compression Algorithms	Direct				
23	Key Server Preferences	Direct				
24	Preferred Key Server	Direct				
25	Primary User ID	First Party				boolean, default false

26	Policy URI	Third-Party				(effectively a human-readable notation)
27	Key Flags	Key Binding	SHOULD			
28	Signer’s User ID	General				
29	Reason for Revocation	Revocation				free text field is effectively a human-readable notation
30	Features	Direct				
31	Signature Target (deprecated)	Attr Value		0x50 3-p conf		Section 4.4

32	Embedded Signature	Attr Value		MAY	0x18 sbind	
33	Issuer Fingerprint	General		MAY		
34	Reserved	-				
35	Intended Recipient Fingerprint	Literal Data	SHOULD			
36	(Delegated Revoker)	Attr Value	MUST		TBD	[I-D.dkg-openpgp-revocation]
37	Reserved (Approved Certifications)	Attr Value			0x16 1pa3pc	[I-D.dkg-openpgp-1pa3pc]
38	Reserved (Key Block)	Literal Data		MAY		
39	Preferred AEAD Ciphersuites	Direct				
40	(Literal Data Metadata)	Literal Data				[I-D.gallagher-openpgp-literal-metadata]
41	(Trust Alias)	Third-Party				

TBD	Replacement Key	Direct, Revocation	SHOULD NOT			[I-D.ietf-openpgp-replacementkey]

Table 1: OpenPGP Signature Subpacket Types

Three subpacket types are Boolean, with different default values for when they are absent (two true, one false). It is RECOMMENDED that these subpackets not be used to convey their default values, only the non-default value. The default value SHOULD instead be conveyed by the absence of the subpacket.

Unless otherwise indicated, subpackets SHOULD NOT be marked critical. In particular, a critical subpacket that invalidates a self-signature will leave the previous self-signature (or no self-signature!) as the most recent valid self-signature from the PoV of some receiving

implementations. A generating implementation MUST be sure that all receiving implementations will behave as intended if a signature containing a critical subpacket is invalidated. Otherwise, with the possible exception of Literal Data signatures, it is NOT RECOMMENDED to set the critical bit.

It is RECOMMENDED that a signature's creator places all subpackets in the hashed area, even self-verifying subpackets for which this is not strictly necessary. The unhashed area MAY be used for informational subpackets attached by third parties (which can be safely stripped).

6.4. Guidance for management of the Signature Subpacket Registry

* Future boolean subpackets SHOULD NOT contain an explicit value; a value of TRUE SHOULD be indicated by the presence of the subpacket, and FALSE otherwise.

* Specification of new subpackets SHOULD address classification, criticality and self-verification as outlined above.

* Subpackets SHOULD be implemented in the private/experimental area first, then reassigned to a permanent code point.

6.5. Unhashed Subpacket Deduplication

Unhashed subpacket areas are malleable and so may have subpackets added or removed in transit, either innocently or maliciously. A receiving implementation SHOULD clean the unhashed area of subpackets that are not meaningful or trustworthy outside the hashed area. If two signature packets are bitwise identical apart from differences in their unhashed subpacket areas, an implementation MAY merge them into a single signature. If two unhashed subpackets in the merged signature are bitwise identical, they MUST be deduplicated. Otherwise, the unhashed subpacket area of the merged signature SHOULD contain the useful subpackets from both original signatures, even if this means multiple subpackets of the same type.

7. Security Considerations

((TO BE COMPLETED))

8. IANA Considerations

8.1. OpenPGP Signature Types Registry

IANA is requested to add a column to the OpenPGP Signature Types registry, called "Embeddable". This column should be empty by default.

IANA is requested to register the following new entry in the registry:

ID	Name	Embeddable	Reference
0x60-0x6F	Private or Experimental Use		Section 5

Table 2: OpenPGP Signature Types (new)

IANA is requested to update the following existing entries in the registry:

ID	Name	Embeddable	Reference
0x19	Primary Key Binding Signature	Yes	[RFC9580], Section 4.1, ((TBC))
0x40	Timestamp Signature		[RFC9580], Section 3.1
0x50	Third-Party Confirmation Signature	Yes	[RFC9580], Section 3.2, Section 4.1

Table 3: OpenPGP Signature Types (updated)

((TODO: avoid clash between the updates to 0x19 here and in draft-certificates))

8.2. OpenPGP Key Flags Registry

IANA is requested to register the following new entry in the OpenPGP Key Flags registry:

Flag	Definition	Reference
((TBC))	This key may be used to make signatures in the Countersignature Category (0x50..0x57)	Section 5

Table 4: OpenPGP Key Flags (new)

IANA is requested to update the following existing entries in the registry:

Flag	Definition	Reference
0x01..	This key may be used to make signatures over other keys, in the Certification and Certification Revocation Categories (0x10..0x17 and 0x30..0x37)	Section 5
0x02...	This key may be used to make signatures in the Literal Data Signature Category (0x00..0x07)	Section 5
0x0008...	This key may be used to make signatures in the Timestamping Category (0x40..0x47)	Section 5

Table 5: OpenPGP Key Flags (update)

8.3. OpenPGP Signature Subpacket Types Registry

IANA is requested to add columns for "Category", "Critical", and "Self-Verifying" to the OpenPGP Signature Subpacket Types registry, and populate them with initial values as listed in Table 1.

IANA is requested to mark the "Revocable" subpacket entry as "deprecated", referencing this document, Section 4.3.

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

[I-D.dkg-openpgp-1pa3pc]
Gillmor, D. K., "First-Party Approved Third-Party Certifications in OpenPGP", Work in Progress, Internet-Draft, draft-dkg-openpgp-1pa3pc-02, 6 September 2024, <https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-1pa3pc-02>.

[I-D.dkg-openpgp-revocation]
Gillmor, D. K. and A. Gallagher, "Revocation in OpenPGP", Work in Progress, Internet-Draft, draft-dkg-openpgp-revocation-02, 28 March 2025, <https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-revocation-02>.

[I-D.gallagher-openpgp-literal-metadata]
Gallagher, A., "OpenPGP Literal Data Metadata Integrity", Work in Progress, Internet-Draft, draft-gallagher-openpgp-literal-metadata-00, 1 January 2024, <https://datatracker.ietf.org/doc/html/draft-gallagher-openpgp-literal-metadata-00>.

[I-D.ietf-openpgp-replacementkey]
Shaw, D. and A. Gallagher, "OpenPGP Key Replacement", Work in Progress, Internet-Draft, draft-ietf-openpgp-replacementkey-08, 29 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-replacementkey-08>.

[OPENPGPDEVBOOK]
"OpenPGP for Application Developers", 6 May 2024, <https://openpgp.dev/book/>.

[RFC1991] Atkins, D., Stallings, W., and P. Zimmermann, "PGP Message Exchange Formats", RFC 1991, DOI 10.17487/RFC1991, August 1996, <https://www.rfc-editor.org/rfc/rfc1991>.

[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/rfc/rfc4880>.

[SQ-WOT] Walfield, N., "OpenPGP Web of Trust", 3 February 2022, <https://sequoia-pgp.gitlab.io/sequoia-wot/>.

Appendix A. Acknowledgments

This document would not have been possible without the extensive work of the authors of [OPENPGPDEVBOOK].

The author would also like to thank Daniel Huigens, Daniel Kahn Gillmor, Heiko Sch辰fer, Neal Walfield, Justus Winter and Paul Schaub for additional discussions and suggestions.

Appendix B. Document History

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

B.1. Changes Between draft-gallagher-openpgp-signatures-02 and draft-gallagher-openpgp-signatures-03

* Renamed document.
* Split out certificate grammar and revocation sections into draft-gallagher-openpgp-certificates.
* Split out message grammar section into draft-gallagher-openpgp-messages.
* Moved sections deprecating revocable and signature target subpackets, with minor updates.
* Minor updates to Timestamp and Third-Party Confirmation signature guidance.
* Relaxed treatment of multiple Embedded Signature and Key Block subpackets.

B.2. Changes Between draft-gallagher-openpgp-signatures-01 and draft-gallagher-openpgp-signatures-02

* Merged in half of draft-dkg-openpgp-revocation and added DKG as co-author.
* Adapted key temporal validity rules for certification signatures.
* Specified use of Persona Certifications for non-trust statements.
* Added section for Primary Key Binding signatures.
* Added sections for conflicting subpackets and conflicting requirements.

* Specified line ending normalization of bare LF only.

* Deprecated revocation of Direct Key signatures.

* Deprecated Signature Target subpackets.

* Added section for issues with temporary identities.

* Refactored and constrained message grammar.

* Fixed some crufty terminology.

B.3. Changes Between draft-gallagher-openpgp-signatures-00 and draft-gallagher-openpgp-signatures-01

* Expanded temporal validity.

* Renamed "Document" and "Data Type" Signature Categories to "Literal Data" and "Attribute Value" respectively.

* Expanded experimental range to cover 0x60..0x6F (96..111).

* Add explicit category ranges to the Key Flags registry.

* Add explicit note about when to ignore Direct and Key Binding subpackets.

* Distinguish between signature subject and signature type-specific data.

* Deprecated the nesting octet.

* Minor errata.

Author’s Address

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