openpgp
Internet-Draft
Updates: 9580 (if approved)
Intended status: Standards Track
Expires: 18 September 2025

A. Gallagher, Ed.
PGPKeys.EU

17 March 2025

OpenPGP Signatures and Signed Messages
draft-gallagher-openpgp-signatures-01

Abstract

This document specifies several updates and clarifications to the OpenPGP signature and message format specifications.

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

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 18 September 2025.

Table of Contents

1. Introduction 3
2. Conventions and Definitions 3
3. Signature Types 4
3.1. Certification Signature Types (0x10..0x13) 4
3.2. Primary Key Revocation Signature (Type 0x20) 5
3.3. Subkey Revocation Signature (Type 0x28) 5
3.4. Certification Revocation Signature (Type 0x30) 6
3.5. Timestamp Signature (0x40) 6
3.6. Third Party Confirmation Signature (0x50) 7
3.6.1. Signature Target Subpacket 8
3.6.2. Use of Third Party Confirmation Signatures by Applications 8
4. Message Grammar 8
4.1. OPS Message Constraints 9
4.2. Subject Normalisation 10
4.3. Nested Signatures 11
4.4. Formal Grammar 11
4.5. Unwrapping Encrypted and Compressed Messages 12
4.6. Marker Packet 13
5. Recursive embedding inside Signature Subpackets 13
6. Signature Categories 14
6.1. Key Flags 16
6.2. Authentication Signatures 17
7. Signature Subpacket Categories 17
7.1. General subpackets. 17
7.2. Context subpackets. 17
7.2.1. Direct subpackets. 18
7.2.2. Revocation subpackets. 18
7.2.3. Key Binding subpackets. 18
7.2.4. First-party Certification subpackets. 18
7.2.5. Third-party Certification subpackets. 19
7.2.6. Literal Data subpackets. 19

7.2.7. Attribute Value subpackets. 19
7.3. Subpackets summary 19
7.4. Guidance for management of the Signature Subpacket Registry 22
7.5. Unhashed Subpacket Deduplication 23
8. Time Evolution of Signatures 23
8.1. Key and Certification Validity Periods 23
8.2. Key Binding Temporal Validity 24
8.3. Certification Temporal Validity 25
8.4. Cumulation of Signatures 25
9. Security Considerations 26
10. IANA Considerations 26
10.1. OpenPGP Signature Types Registry 26
10.2. OpenPGP Key Flags Registry 27
10.3. OpenPGP Signature Subpacket Types Registry 28
11. References 28
11.1. Normative References 28
11.2. Informative References 29
Appendix A. Acknowledgments 31
Appendix B. Document History 31
B.1. Changes Between draft-gallagher-openpgp-signatures-00 and draft-gallagher-openpgp-signatures-01 31
Author’s Address 32

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.

* Constraining the formal message grammar.

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. Certification Signature Types (0x10..0x13)

Section 5.2.1 of [RFC9580] defines four types of certification signature (0x10..0x13). All may be created by either the key owner or a third party, and may be calculated over either a User ID packet or a User Attribute packet. In addition, a Certification Revocation signature revokes signatures of all four types.

Historically, certifications were only made by third parties. First-party self-certifications only became customary later, and were made mandatory when preference subpackets were introduced.

The semantic distinctions between the certification signature types were left ill-defined and most software treats them as equivalent. The following convention has evolved over time [ASKCERTLEVEL], and is hereby specified:

* 0x10 Generic Certification SHOULD only be used for third-party certifications.

* 0x11 Persona Certification is deprecated and SHOULD NOT be created.

* 0x12 Casual Certification is deprecated and SHOULD NOT be created.

* 0x13 Positive Certification SHOULD only be used for self-certifications.

A receiving implementation MUST treat a third-party certification of any of the above types as equivalent to a type 0x10 signature, and a first-party certification of any of the above types as equivalent to a type 0x13 signature.

3.2. Primary Key Revocation Signature (Type 0x20)

Section 5.2.1.11 of [RFC9580] defines the Key Revocation Signature as:

> This signature is calculated directly on the key being revoked. A revoked key is not to be used. Only Revocation Signatures by the key being revoked, or by a (deprecated) Revocation Key, should be considered valid Revocation Signatures.

The name and description are potentially confusing, as it can only revoke a Primary Key and not a Subkey -- other OpenPGP artifacts that are named "Key" without a qualifier (such as the "Key Flags" and "Key Expiration Time" subpackets) apply to both Primary Keys and Subkeys.

We therefore rename the 0x20 signature type to "_Primary_ Key Revocation Signature" for clarity, and update its definition as follows:

> This signature is calculated directly on the primary key being revoked. A revoked primary key is not to be used. Only Revocation Signatures by the primary key being revoked, or by a (deprecated) Revocation Key, should be considered valid Primary Key Revocation Signatures.

3.3. Subkey Revocation Signature (Type 0x28)

Section 5.2.1.11 of [RFC9580] defines the Subkey Revocation Signature as:

> This signature is calculated directly on the primary key and the subkey being revoked. A revoked subkey is not to be used. Only Revocation Signatures by the top-level signature key that is bound to this subkey, or by a (deprecated) Revocation Key, should be considered valid Revocation Signatures.

The phrasing "top-level signature key that is bound to this subkey" is confusing. Instead, we update the definition for clarity:

> This signature is calculated directly on the primary key and the subkey being revoked. A revoked subkey is not to be used. Only Revocation Signatures by the primary key, or by a (deprecated) Revocation Key, should be considered valid Subkey Revocation Signatures.

3.4. Certification Revocation Signature (Type 0x30)

Section 5.2.1.13 of [RFC9580] defines the Certification Revocation Signature as:

> This signature revokes an earlier User ID certification signature (Type IDs 0x10 through 0x13) or Direct Key signature (Type ID 0x1F). It should be issued by the same key that issued the revoked signature or by a (deprecated) Revocation Key. The signature is computed over the same data as the certification that it revokes, and it should have a later creation date than that certification.

Section 5.2.4 of [RFC9580] is clear that Direct Key signatures and Certification Signatures have completely different constructions. This implies that there are two different ways to construct a Type 0x30 signature, each of which appears in a different part of an OpenPGP certificate.

The above definition dates back to [RFC2440], except for the "or Direct Key Signature" clause which was added to the first sentence in [RFC4880]. But the third sentence still defines the construction unconditionally by reference to "the certification that it revokes", even though it does not necessarily revoke a certification.

The use of a Certification Revocation Signature to revoke a Direct Key Signature is imprecise and not widely supported, and is hereby deprecated.

(See also Section 6.1)

3.5. 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, and is constructed 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.

3.6. 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 MAY be included in an Embedded Signature packet in the unhashed area of the signature it notarises; if it is so located, a Signature Target SHOULD NOT be included. Otherwise, it SHOULD be distributed as a detached signature.

3.6.1. Signature Target Subpacket

The Signature Target subpacket is not a unique identifier of a signature packet [REVOC-13] and so does not fulfil its design goals.

It is deprecated for use in revocation signatures by [I-D.dkg-openpgp-revocation] ((TBC: this is just an issue for now, not in the draft)), and we hereby deprecate it entirely.

((TODO: use the Approved Certifications subpacket instead! But rename it to "target signatures"...))

3.6.2. Use of Third Party Confirmation Signatures by Applications

We may wish to allow the application layer to make validity claims using countersignatures. For example, a key server may wish to record that it has verified a User ID by automated means. The key server may not wish to make a Certification signature, to prevent the cumulation of many such automated signatures (Section 8.4). For the same reason, it may not wish to embed its countersignature in an unhashed area of a signature packet in the certificate.

It could make a Third-Party Confirmation signature over the most recent self-certification, and distribute it as a detached signature, perhaps in a mixed keyring (Section 8.1 of [I-D.gallagher-openpgp-hkp]).

((TBC))

4. Message Grammar

The accepted convention is that a prefixed Signature packet signs over the next literal packet only, skipping any intervening signatures - however this is not explicitly specified in [RFC9580]. Historically, PGP 2.X treated a prefixed Signature packet as applying to the entire following sequence of packets, but this usage is deprecated [FINNEY1998]. See Section 4.3 for an alternative construction.

In addition, One-Pass Signature (OPS) nesting semantics are complex, and under-specified [SCHAUB2022]. Section 5.4 of [RFC9580] defines the nesting octet as:

A 1-octet number holding a flag showing whether the signature is nested. A zero value indicates that the next packet is another One-Pass Signature packet that describes another signature to be applied to the same message data.

The terminology is imprecise, and non-zero "nesting" flags are completely unspecified. One self-consistent interpretation is as follows:

* A zero nesting octet means that the following OPS and its counterpart signature are not signed over by the current OPS.
  - This process is recursive if multiple sequential OPS packets have a nesting octet of zero.
* To add multiple OPS signatures over the same message data, all OPS constructions except the innermost one have the nesting octet zeroed.
  - It is not clear what happens if the innermost nesting octet is zero but no OPS packet follows.

The above implies that an OPS with a nonzero nesting octet signs over all packets between the OPS packet and its matching signature packet, including any further signatures, however it is not clear whether any current implementation supports this.

This is further expanded in [OPENPGPDEVBOOK].

This still leaves us with an overly complex grammar that resists rigorous formalisation. We attempt to improve the formalism below.

4.1. OPS Message Constraints

We constrain OPS structures to a subset of previously-allowed configurations:

* A prefixed Signature packet signs over the next literal or compressed packet, ignoring any intervening signature or OPS packets.
* Prefixed signatures and OPS signatures MUST NOT both be used in the same message.
* When generating a version 4 OPS packet that is not followed by another OPS packet, the nesting octet SHOULD be set to 1.
  - Otherwise, the nesting octet SHOULD be set to 0.

* When consuming an OPS packet, the nesting octet MUST be ignored.

This effectively deprecates the nesting octet, while maintaining backwards compatibility with legacy code.

4.2. Subject Normalisation

The _subject_ of an OpenPGP signature refers to the packet(s) that are signed over. The _type-specific data_ of an OpenPGP signature refers to the section of the data stream that is passed to the signature's digest function after the optional salt and before the trailer. The type-specific data differs from the subject in that it has been normalised, the details of which are dependent on the signature type.

The subject of a signature in the Literal Data category (Section 6) is the Literal Data packet that immediately follows one or more prefixed signatures, or is enclosed by one or more OPS constructions. The Literal Data packet MAY be encoded in a Compressed Data packet. If no Literal Data or Compressed Data packet is present, or if the Compressed Data packet does not decompress to a Literal packet, the signature is malformed.

The following normalisation steps are applied to the subject of the signature to produce the type-specific data:

* Any Compressed Data packet is replaced by its uncompressed contents, which MUST be a Literal Data packet.

* The framing of the Literal Data packet is discarded, and any partial-length packets are concatenated.

* If the Signature Type is 0x01, the Literal Data packet body is converted to Canonical Text, by converting line endings to CRLF and removing any trailing whitespace.

A One-Pass Signature over a Literal Data packet, a prefixed Signature over the same packet, and a detached signature over a file containing the body of the same packet are all calculated the same way. This means that they can be losslessly transformed into each other with the exception of the Literal Data metadata fields, which an application MAY assume contain their recommended default values as per Section 5.9 of [RFC9580].

4.3. Nested Signatures

To sign over an entire signed message together with its signatures, the wire format of the inner message SHOULD first be encapsulated in a Literal Data packet. A Canonical Text signature MUST NOT be made over such a nested message, and the Cleartext Signature Framework MUST NOT be used.

Beware that the outer signature will thus be sensitive to the inner message's packet framing, i.e. the otherwise inconsequential choice of packet header format and partial body lengths. If the inner message is parsed and re-serialised unmodified, but using a different framing, the outer signature will no longer validate.

4.4. Formal Grammar

The message grammar in Section 10.3 of [RFC9580] is therefore updated to:

* OpenPGP Message:
  Encrypted Message | Unencrypted Message.

* Unencrypted Message:
  Signed Message | Unsigned Message.

* Unsigned Message:
  Compressed Message | Literal Message.

* Compressed Message:
  Compressed Data Packet.

* Literal Message:
  Literal Data Packet.

* ESK:
  Public Key Encrypted Session Key Packet | Symmetric Key Encrypted Session Key Packet.

* ESK Sequence:
  ESK | ESK Sequence, ESK.

* Encrypted Data:
  Symmetrically Encrypted Data Packet | Symmetrically Encrypted and Integrity Protected Data Packet.

* Encrypted Message:
  Encrypted Data | ESK Sequence, Encrypted Data.

* Nested One-Pass Signed Message:
  Nested One-Pass Signature Packet, One-Pass Signed Message,
  Corresponding Signature Packet.

* Literal Body One-Pass Signed Message:
  Literal Body One-Pass Signature Packet, Unsigned Message,
  Corresponding Signature Packet.

* Verbatim One-Pass Signed Message:
  Verbatim One-Pass Signature Packet, Unencrypted Message,
  Corresponding Signature Packet.

* One-Pass Signed Message:
  Nested One-Pass Signed Message | Literal Body One-Pass Signed
  Message | Verbatim One-Pass Signed Message.

* Signed Message:
  Signature Packet, Unencrypted Message | One-Pass Signed Message.

* Optionally Padded Unencrypted Message:
  Unencrypted Message | Unencrypted Message, Padding Packet.

In addition to these rules, a Marker packet (Section 5.8 of [RFC9580]) can appear anywhere in the sequence.

4.5. Unwrapping Encrypted and Compressed Messages

[RFC9580] permits an encrypted message to contain another encrypted message, and a compressed message to contain another compressed message, possibly recursively. Such messages require potentially unbounded resources for negligible added utility, and therefore MUST NOT be created.

In addition, encrypt-then-sign messages are not idiomatic OpenPGP, and SHOULD NOT be generated.

The guidance in Section 10.3.1 of [RFC9580] is therefore updated to:

* Decrypting a version 2 Symmetrically Encrypted and Integrity Protected Data packet MUST yield a valid Optionally Padded Unencrypted Message.

* Decrypting a version 1 Symmetrically Encrypted and Integrity Protected Data packet or -- for historic data -- a Symmetrically Encrypted Data packet MUST yield a valid Unencrypted Message.

* Decompressing a Compressed Data packet MUST yield a valid Literal Message.

((TODO: is this too constraining on compressed data packet contents?))

4.6. Marker Packet

Section 5.8 of [RFC9580] defines the Marker Packet as follows:

> The body of the Marker packet consists of:
>
> - The three octets 0x50, 0x47, 0x50 (which spell "PGP" in UTF-8).
>
> Such a packet MUST be ignored when received.

We update this to include:

* If a receiving implementation encounters a Marker Packet with any other contents, the entire packet sequence SHOULD be ignored.

* A Marker Packet MAY be added by an application to notify non-OpenPGP software that a data stream contains OpenPGP data. If so, the Marker Packet SHOULD be the first packet in the sequence, and SHOULD NOT use a Legacy header, so that it can be easily detected.

5. 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

We wish to prevent infinite recursion via embedded signatures, in order to avoid resource exhaustion. This can be achieved as follows:

* 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 Key Block subpacket MUST only be used inside a signature type in the Literal Data Signature Category.

((TODO: Key Block allows us to smuggle a key into the OpenPGP layer without requiring support by the application layer. We could instead update the message grammar to allow TPKs to be appended to a Literal Message. Going further, we could unify the packet sequence grammar so that there is only one kind of sequence, which would include both messages and keyrings. See also "mixed keyrings" in Section 8.1 of [I-D.gallagher-openpgp-hkp].))

6. 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 (null document)
* 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 bind
  - 0x19 Primary key bind

    - 0x1F Direct key (self bind)

* 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)

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.

6.1. 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)) Primary Key Revocation 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, Key Revocation and Subkey Revocation Categories.

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

* Any key 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. This is not widely supported and is hereby deprecated.

6.2. 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.

It should be noted that cross-protocol usage can be exploited to evade the domain separation protections of Key Flags. For example, there is no distinction between signature, 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.))

7. Signature Subpacket Categories

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

7.1. 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 categorised here as general subpackets, however the notations within them may have arbitrary semantics at the application layer)

7.2. Context subpackets.

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

7.2.1. 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). The Replacement Key subpacket MAY also be used as a key revocation subpacket. 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: (Additional Decryption Key), 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).

7.2.2. 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.

7.2.3. 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.

7.2.4. 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

7.2.5. 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).

7.2.6. 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 Meta Hash).

7.2.7. Attribute Value subpackets.

These are only meaningful in signature types whose specification explicitly requires them. They SHOULD NOT be used elsewhere. They have no intrinsic semantics; all semantics are defined by the enclosing signature.

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

7.3. Subpackets summary

Type	Name	Category	Critical	Self-Verifying	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	Third	MUST IFF			boolean, default true

|

	Certification	Party	false			
5	Trust Signature	Third Party				
6	Regular Expression	Third Party	SHOULD			
7	Revocable	Third Party				boolean, default false (deprecated in [I-D.dkg-openpgp-revocation])
8	Reserved	-				never used
9	Key Expiration Time	Key Binding	SHOULD			
10	(Additional Decryption Key/ARR)	Key Binding				PGP.com proprietary feature
11	Preferred Symmetric Ciphers	Direct				
12	Revocation Key (deprecated)	Direct				
13-15	Reserved	-				never used
16	Issuer Key ID	General		Yes		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	Direct				

	Server					
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	Attr Value			0x50 3-p conf	not a unique identifier [REVOC-13]
32	Embedded Signature	Attr Value		Yes IFF it contains an 0x19 signature	0x18 sbind	
33	Issuer Fingerprint	General		Yes		

34	(Preferred AEAD Algorithms)	Direct				[I-D.ietf-openpgp-rfc4880bis]
35	Intended Recipient Fingerprint	Literal Data	SHOULD			
36	(Delegated Revoker)	Attr Value	MUST		TBD	[I-D.dkg-openpgp-revocation]
37	(Approved Certifications)	Attr Value			0x16 1pa3pc	[I-D.dkg-openpgp-1pa3pc]
38	(Key Block)	Literal Data		Yes		[I-D.ietf-openpgp-rfc4880bis]

39	Preferred AEAD Ciphersuites	Direct				
40	(Literal Data Meta Hash)	Literal Data				not yet implemented [I-D.koch-librepgp]
41	(Trust Alias)	Third Party				not yet implemented [I-D.koch-librepgp]
TBD	(Replacement Key)	Direct	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).

7.4. 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.

7.5. 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.

8. Time Evolution of Signatures

Validation of a Signature packet is performed in several stages:

1. Formal Validation (the signature packet is well-formed and parseable)

2. Cryptographic Validation (the signature data was calculated correctly)

3. Structural Validation (the signature packet is placed in the correct context)

4. Temporal Validation (the signature has not expired or been revoked)

5. Issuer Validation (the signature was made by a valid key)

Included in the Issuer Validation stage is validation (including Temporal Validation) of the binding signatures in the issuer’s certificate. If the Web of Trust is in use, this process is potentially recursive.

8.1. Key and Certification Validity Periods

Key Expiration Time subpackets are a rich source of footguns:

1. They specify an offset rather than an timestamp, but are not usable without first converting to a timestamp.

2. The offset is calculated relative to the creation timestamp of a different packet (the component key packet).

3. Some implementations interpret them as being inheritable in their raw form, so that the same offset value gets applied to different creation timestamps.

Further, their semantics overlaps that of Signature Expiration Time:

1. If the binding signature over a key expires, but the key does not, the key is nevertheless unusable due to lack of signatures.

2. If a key expires, but the signature over it does not, the signature is unusable.

This means there are effectively two expiration dates on a Key Binding signature, the key expiration and the signature expiration, but without distinct semantics.

In addition, the Signature Creation Time subpacket has an overloaded meaning in both Key Binding and Certification signatures:

1. It is used as the "valid from" timestamp of the object being signed over

2. It is used to order multiple similar signatures to determine which is valid

If this is interpreted strictly, it means that it is not possible to create a new Key Binding signature that reliably leaves the starting date of the key’s validity unchanged. Some implementations have worked around this by generating signatures with creation dates backdated to one second after that of the previous signature.

The ability to create a new signature with an unchanged valid-from date allows historical signatures to be losslessly cleaned from a TPK, saving space. It is also more compatible with the historical interpretation favoured by PGP and GnuPG.

8.2. Key Binding Temporal Validity

To clean up the ambiguity in Key Binding signatures, we specify the following:

1. Key Binding signatures (Subkey Binding signatures, Primary Key Binding signatures, Direct Key signatures, BUT NOT self-certifications over v4 Primary User IDs) SHOULD NOT contain Signature Expiration Time subpackets.

2. The validity of a component key extends from its creation time until its revocation or key expiration time.

3. If the most recent Key Binding signature has no Key Expiration Time subpacket, then the key does not expire.

4. Key Binding signatures cannot be directly revoked; the corresponding revocation signatures affect the key, not the binding.

5. A Key Binding signature is temporally valid if its creation time is later than the creation time of the primary key that made it.

6. A Key Binding signature is temporally valid even if the primary has been hard-revoked (so that we can still associate the primary key with its subkeys).

7. The creation time of the Key Binding signature is used only for ordering, not for calculation of signature validity.

8. Key Expiration Time subpackets are only meaningful in Key Binding signatures; an implementation MUST ignore a Key Expiration Time subpacket in any other signature.

A signature other than a Key Binding signature is temporally valid if it was made by a component key during its validity period.

(See also [RFC4880BIS-71], [OPENPGPJS-1800], and [REVOC-19]).

8.3. Certification Temporal Validity

It is not customary for Certification signatures, even self-certifications over v4 Primary User IDs, to contain Key Expiration Time subpackets. Instead, the Signature Expiration Time subpacket is used. In addition, User IDs and User Attributes do not contain a creation time field. Together, this means that we cannot solve the temporal validity issue for Certifications as easily as we did for Key Bindings.

One possible solution is specified in Section 5 of [I-D.gallagher-openpgp-user-attributes].

8.4. Cumulation of Signatures

A cryptographically valid Key Binding, Certification or Literal Data signature automatically and permanently supersedes any earlier signature of the same Signature Category, by the same key pair, over the same subject. If a later such signature expires before an earlier one, the earlier signature does not become valid again.

For the purposes of the above:

* "same key pair" refers to the public key packet as identified by the Issuer KeyID or Issuer Fingerprint subpacket.

* "same subject" refers only to the packets being signed over, and not to the metadata contained in the Signature packets (including subpackets) or any corresponding OPS packet.

Note however that this does not apply to revocation signatures, which have their own cumulation rules (see [I-D.dkg-openpgp-revocation]).

(See also [SCHAUB2021])

9. Security Considerations

The OPS Subject Type octet is not signed over and is malleable in principle. An intermediary could swap a Nested OPS with its inner OPS by also swapping the type octets. The order of OPS nesting therefore MUST NOT be considered meaningful.

In addition, the normalisation applied during Literal Body signature calculation may result in semantic collisions. It is possible to construct distinct sequences of packets that map to the same sequence of octets after Literal Body normalisation is applied. It is not known whether such a pair of colliding packet sequences might also have different semantics.

10. IANA Considerations

10.1. 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 6

Table 2: OpenPGP Signature Types (new)

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

ID	Name	Embeddable	Reference
0x10	Generic Certification Signature		[RFC9580], Section 3.1
0x11	Persona Certification Signature (Deprecated)		[RFC9580], Section 3.1
0x12	Casual Certification Signature (Deprecated)		[RFC9580], Section 3.1
0x13	Positive Certification Signature		[RFC9580], Section 3.1
0x19	Primary Key Binding Signature	Yes	[RFC9580], Section 5
0x20	Primary Key Revocation Signature		[RFC9580], Section 3.2
0x40	Timestamp Signature		[RFC9580], Section 3.5
0x50	Third Party Confirmation Signature	Yes	[RFC9580], Section 3.6, Section 5

Table 3: OpenPGP Signature Types (updated)

10.2. OpenPGP Key Flags Registry

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

Flag	Definition	Reference
((TBC))	This key may be used to make signatures in the Primary Key Revocation Category (0x20..0x27)	Section 6, [I-D.dkg-openpgp-revocation]
((TBC))	This key may be used to make signatures in the Countersignature Category (0x50..0x57)	Section 6

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 6
0x02...	This key may be used to make signatures in the Literal Data Signature Category (0x00..0x07)	Section 6
0x0008...	This key may be used to make signatures in the Timestamping Category (0x40..0x47)	Section 6

Table 5: OpenPGP Key Flags (update)

10.3. 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.

11. References

11.1. 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>.

11.2. Informative References

[ASKCERTLEVEL] Gillmor, D. K., "'gpg --ask-cert-level' Considered Harmful", 20 May 2013, <https://dkg.fifthhorseman.net/blog/gpg-ask-cert-level-considered-harmful.html>.

[FINNEY1998] Finney, H., "Re: More spec-ulations - update", 26 March 1998, <https://mailarchive.ietf.org/arch/msg/openpgp/U4Qg3Z9bj-RDgpwW5nmRNetOZKY/>.

[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., "Revocation in OpenPGP", Work in Progress, Internet-Draft, draft-dkg-openpgp-revocation-01, 17 August 2023, <https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-revocation-01>.

[I-D.gallagher-openpgp-hkp] Shaw, D. and A. Gallagher, "OpenPGP HTTP Keyserver Protocol", Work in Progress, Internet-Draft, draft-gallagher-openpgp-hkp-06, 31 December 2024, <https://datatracker.ietf.org/doc/html/draft-gallagher-openpgp-hkp-06>.

[I-D.gallagher-openpgp-user-attributes]
Gallagher, A., "User Attributes in OpenPGP", Work in Progress, Internet-Draft, draft-gallagher-openpgp-user-attributes-00, 11 February 2025, <https://datatracker.ietf.org/doc/html/draft-gallagher-openpgp-user-attributes-00>.

[I-D.ietf-openpgp-replacementkey]
Shaw, D. and A. Gallagher, "OpenPGP Key Replacement", Work in Progress, Internet-Draft, draft-ietf-openpgp-replacementkey-03, 3 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-replacementkey-03>.

[I-D.ietf-openpgp-rfc4880bis]
Koch, W., carlson, B. M., Tse, R. H., Atkins, D., and D. K. Gillmor, "OpenPGP Message Format", Work in Progress, Internet-Draft, draft-ietf-openpgp-rfc4880bis-10, 31 August 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-rfc4880bis-10>.

[I-D.koch-librepgp]
Koch, W. and R. H. Tse, "LibrePGP Message Format", Work in Progress, Internet-Draft, draft-koch-librepgp-02, 9 September 2024, <https://datatracker.ietf.org/doc/html/draft-koch-librepgp-02>.

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

[OPENPGPJS-1800]
Hell, I., "'Signature creation time is in the future' error for apparently valid signature", 28 October 2024, <https://github.com/openpgpjs/openpgpjs/issues/1800>.

[REVOC-13] Gallagher, A., "Deprecate use of 'Signature Target' subpacket in revocation signatures", 7 April 2024, <https://gitlab.com/dkg/openpgp-revocation/-/issues/13>.

[REVOC-19] Gallagher, A., "Add validity period guidance?", 30 October 2024, <https://gitlab.com/dkg/openpgp-revocation/-/issues/19>.

[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>.

[RFC2440] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", RFC 2440, DOI 10.17487/RFC2440, November 1998, <https://www.rfc-editor.org/rfc/rfc2440>.

[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>.

[RFC4880BIS-71] Gallagher, A., "Deprecate the use of 'Key Expiration Time' packets (type 9) in V5 sigs", 8 January 2022, <https://gitlab.com/openpgp-wg/rfc4880bis/-/issues/71>.

[SCHAUB2021] Schaub, P., "[openpgp] Question on Signature Expiration", 13 December 2021, <https://mailarchive.ietf.org/arch/msg/openpgp/C0P4MxwqJBbxS6H0YoXFF3oEJ3A/>.

[SCHAUB2022] Schaub, P., "[openpgp] Proposing a Simplification of Message Syntax", 7 October 2022, <https://mailarchive.ietf.org/arch/msg/openpgp/uepOF6XpSegMO4c59tt9e5H1i4g/>.

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, 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-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