

openpgp
Internet-Draft
Intended status: Informational
Expires: 1 August 2026

D. K. Gillmor
ACLU
H. Krekel
merlinux gmbh
F. Ziegelmayer
n0 Inc.
28 January 2026

Autocrypt v2 OpenPGP Certificates and Transferable Secret Keys
draft-autocrypt-openpgp-v2-cert-00

Abstract

This document describes the "Autocrypt v2 Certificate", a standard structure for an OpenPGP certificate for Internet messaging (such as email) that offers a defense against both store-now-decrypt-later attacks from quantum computer and future key exfiltration attacks. It is also structured to support reasonable in-band transmission, using established mechanisms like the Autocrypt header field. This document also describes the structure, use, and maintenance of the OpenPGP Transferable Secret Key that corresponds with the Autocrypt v2 Certificate.

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://autocrypt2.codeberg.page/autocrypt-v2-cert/>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-autocrypt-openpgp-v2-cert/>.

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://codeberg.org/autocrypt2/autocrypt-v2-cert/>.

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 1 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
1.1. Requirements Language	4
1.2. Terminology	4
1.3. Goals	5
1.4. Non-Goals	6
2. Certificate Structure	8
2.1. Encryption subkey rotation	8
2.2. Packet Composition	9
2.3. Certificate Size	10
2.4. Use of OpenPGP format	10
3. Identifying an Autocrypt v2 Transferable Secret Key	10
3.1. Identification By TSK Structure	10
3.2. Determining min _{rd} and max _{rd}	11
4. Reliable Deletion Strategy	11
4.1. Keyholder Certificate and Secret Key Management	11
4.1.1. Subkey Ratchet	11
4.1.2. Ratcheting Schedule	15
4.1.3. Custom Ratcheting Schedules	16
4.1.4. Timing of Secret Key Destruction	17
4.1.5. Autocrypt v2 TSK and Certificate Timeline	17

4.2.	Receiving and Merging Certificates	18
4.2.1.	Minimization and pruning of Autocrypt v2 certificates	18
4.3.	Encrypting to an Autocrypt v2 Certificate	19
4.4.	Message Deletion	19
4.4.1.	Keyholder Processing of Encrypted Messages	19
4.4.2.	Summary of Message Processing Strategies	21
5.	Security Considerations	21
5.1.	Reliable Deletion Threat Model	22
5.2.	The Cleartext Is The Targeted Asset	22
5.3.	Key Destruction: Time-based vs. Message-based	22
5.4.	The Window of Recoverability	23
5.5.	System Clock Reliance	24
5.5.1.	Avoiding System Clock Skew	24
6.	Usability Considerations	25
7.	IANA Considerations	26
8.	References	26
8.1.	Normative References	26
8.2.	Informative References	27
Appendix A.	Historical notes	28
Appendix B.	Design Choices	28
B.1.	Subkey Validity Windows	29
B.1.1.	Back-to-Back Validity Windows	29
B.1.2.	Overlapping Validity Windows	30
B.1.3.	Superset Validity Windows	30
B.1.4.	Validity Window Conclusion	31
B.2.	Ed25519 for Primary Key	31
B.3.	Balancing Rotation and Reliable Deletion Timing	31
B.4.	No Explicit Annotations	32
B.5.	Seed-free Ratcheting	32
Appendix C.	Test vectors	32
C.1.	Key and Certificate Created	32
C.1.1.	Initial Certificate	32
C.1.2.	Initial TSK	34
C.2.	After New Subkey Added	36
C.2.1.	Certificate With New Subkey	38
C.2.2.	TSK With New Subkey	40
C.2.3.	Merged Certificate	42
C.3.	Old Subkey Removed	44
C.3.1.	TSK With Old Subkey Removed	44
Acknowledgements	46
Document History	46
Authors' Addresses	46

1. Introduction

An OpenPGP certificate can be structured in a bewildering variety of ways. The "Autocrypt v2 Certificate" is a modern OpenPGP structure that aims toward robustness, cryptographic resilience, and straightforward deployment in the Internet messaging context.

An OpenPGP implementation that produces and can handle certificates and secret keys structured in this way can provide the user with reasonable protection against a variety of plausible attacks, while slotting cleanly into existing mechanisms for end-to-end cryptographically protected email and other messaging systems.

This mechanism also enables an archiving messenger to support robust deletion of a received message in a way that the deleted message will not be recoverable, even by an adversary who can capture messages in transit, and later compromises the user's message archive and secret keys.

1.1. Requirements Language

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.

1.2. Terminology

- * "OpenPGP certificate" or just "certificate" refers to an OpenPGP Transferable Public Key (see Section 10.1 of [RFC9580]). This specification distinguishes between outgoing certificates as distributed by the keyholder, with a fixed structure and size, and incoming certificates as received and stored by a peer. This specification does not prescribe any particular mechanism for how certificates are transferred between parties.
- * "Transferable Secret Key" or just "TSK" refers to an OpenPGP Transferable Secret Key (see Section 10.2 of [RFC9580]).
- * "Component key" refers to a single cryptographic object found within an OpenPGP certificate. A certificate's primary key is a "component key", and any subkey in the certificate is also a "component key".

- * "Keyholder" is the party that has legitimate access to the secret key material corresponding to the component keys in a certificate. The keyholder can sign messages that can be verified with the certificate, decrypt messages that were encrypted to the certificate, and update the certificate itself over time.
- * "User Messaging Agent" or UMA refers to a program used to compose, send, receive, and render messages. This term is similar to "Mail User Agent" (MUA), but the variation in naming emphasizes that Autocrypt v2 certificates do not prescribe a specific transport mechanism nor do they prescribe a particular message content format. A Keyholder may have one or more UMAs that share access to the same secret key material, messaging inbox, and other account details.
- * "Peer" means a party who communicates with the keyholder via messages, as well as potentially with other peers. The peer does not have access to the keyholder's asymmetric secret key material, but typically has access to a copy of each message exchanged between the peer and keyholder.
- * "Reliable Deletion" refers to the property that enables a keyholder to render a message unrecoverable after deletion, even if an attacker has archived all messages in transit and subsequently obtains the keyholder's secret key material and message archive. Reliable deletion therefore requires the destruction of both secret key material and message cleartext. This property is commonly referred to as "Forward Secrecy"; however, this specification uses the term "Reliable Deletion" to emphasize the keyholder's ability to permanently delete messages.

1.3. Goals

A certificate following this specification has the following goals:

- * Post-quantum confidentiality. It should defend against a store-now-decrypt-later attack from a cryptographically relevant quantum computer.
- * Size matters. When network conditions are constrained by limited bandwidth, high latency, or intermittent connectivity, there often is a heightened need for encryption with reliably deletable messages. To preserve availability under such conditions, the size of cryptographic material should be minimized and it should be possible to include transmission of a few dozen certificates in a single message without impairing common messaging infrastructure.

- * Computational resources matter. It should be possible to promptly encrypt a single message to up to a few dozen of these certificates with low-powered devices. The same hardware should be able to quickly and cheaply verify a signature from one of these certificates. And with access to the corresponding secret keys, it should be inexpensive to decrypt a message encrypted to one, or to sign a message with one.
- * Reliable deletion. The keyholder should be able to robustly and permanently delete an encrypted message received some time in the past, rendering it unrecoverable even to a powerful attacker in the future (see Section 5.1).
- * No network synchronization needed. The keyholder should be able to encrypt and decrypt messages with local key material only, without requiring network synchronization between their own devices or with peers. To the extent that a keyholder has multiple UMAs of their own, each UMA should be able to operate independently with no ongoing network synchronization between them beyond the initial configuration.
- * Backward compatibility. It must be possible for a keyholder with an Autocrypt v1 key to sign a message that is encrypted to an Autocrypt v2 certificate, and vice versa. It must also be possible to encrypt a message to a mixed group of Autocrypt v1 and v2 certificates (though such a message may not meet the "Reliable deletion" goal).
- * Drop-in OpenPGP replacement for existing peers. The keyholder might need to update their OpenPGP implementation, UMA(s), and regular workflow to use the secret key material to meet these goals successfully. But a peer whose UMA supports Autocrypt v1.1 already only needs to update their OpenPGP implementation in order to interoperate.

1.4. Non-Goals

This specification does not attempt to accommodate all possible scenarios that might occur with OpenPGP, email, or other messaging systems. The following concerns are explicitly not in-scope for this specification:

- * No support for legacy OpenPGP clients. The keyholder needs to use an up-to-date OpenPGP implementation, and expects their peers to do the same. There is no attempt at backward compatibility for a peer with a legacy OpenPGP implementation.

- * No in-cert human-readable identifiers. There is no attempt to store a "real name" or other human-readable identity information in the certificate. If human-readable identity information is to be associated with the certificate, it is expected to be supplied elsewhere, such as with a local petname system, or some external cryptographic bindings.
- * No in-cert transport-layer addressing. There is no attempt to bind transport-layer routing information (e.g., email addresses) to the certificate. Information about how to get an encrypted message to the keyholder is presumed to be transmitted via some other channel, such as the Autocrypt header field.
- * No defense against quantum impersonation. While the certificate is designed to defend against store-now-decrypt-later attacks, it does not defend against an adversary with a cryptographically relevant quantum computer that breaks the signing key and updates the certificate in order to compromise future messages encrypted to the certificate. Such an attacker must expose themselves to the peer at least (putting the attacker at risk due to visibility), and these attacks cannot take place retroactively.
- * No transitioning from old certs. There is no specific support for transitioning older or alternative OpenPGP certificate formats to the structure defined in this specification. Such a migration path may be defined in future specifications, but this document is limited to new adopters going forward.
- * No certificate discovery and no refresh. This specification does not define a mechanism to request a refreshed certificate from a peer. Reliable message deletion depends on timely updates of each correspondent's certificate, but imposing a particular mechanism is unlikely to accommodate the constraints of diverse messaging implementations. That said, this specification does not preclude the use of discovery or refresh mechanisms.
- * No coordinated deletion of messages. This specification addresses unilateral deletability, protecting against a future attacker who compromises the keyholder's UMA or secret key storage. Messages are typically stored in multiple locations, across peers, devices, and UMAs. Guaranteeing deletion of all copies of a message, including through negotiated "disappearing messages" mechanisms, is outside the scope of this draft.
- * No post-compromise security. Some messaging schemes attempt to defend against an attacker who has compromised the user's secret key material at some point in the past, typically by regularly mixing new entropy into the secret key material and coordinating

those updates across the user's shared devices. This specification does not defend against such an attacker. In the case of past key compromise, the user may need to move to an entirely new certificate.

2. Certificate Structure

An Autocrypt v2 certificate is composed of a specific sequence of version 6 OpenPGP packets. The precise requirements for these packets including algorithm IDs, key flags, and binding signatures are detailed in Section 2.2.

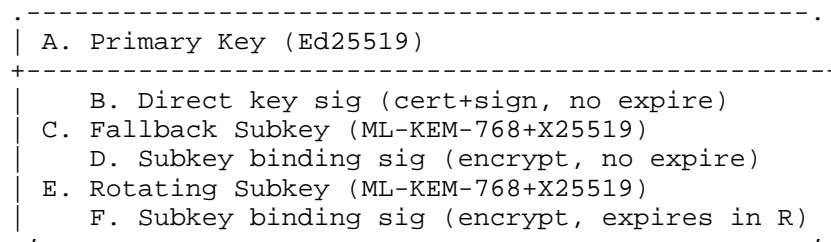


Figure 1: Autocrypt v2 Certificate Structure

2.1. Encryption subkey rotation

The expiring encryption subkey is rotated on a regular schedule. The window of subkey validity (from creation and binding to expiration) is known as `max_rd`. When the current subkey has only `min_rd` time left until expiration, the keyholder adds a new rotating subkey. A rotating subkey `N+1` is created exactly at `min_rd` away from the end of the validity window of rotating subkey `N`. By convention, `max_rd` is 10 days (864000 seconds), and `min_rd` is 5 days (43200 seconds). See Section 3.2 for the formal definition and concrete guidance about `min_rd` and `max_rd`.

In order for the keyholder to use the same TSK across multiple UMAs without explicit network coordination, the values of `min_rd` and `max_rd` MUST be known to the keyholder's UMAs. See Section 4.1.3 for more information.

This arrangement means the validity window of each subsequent rotating subkey overlaps with the validity window of the prior rotating subkey. See Appendix B.1 for more discussion about temporal layout of subkey validity windows.

2.2. Packet Composition

An outbound certificate consists of the following six packets:

- * A. "Primary" Public Key Packet (packet type ID 6), version 6, public key algorithm Ed25519 (algorithm ID 27), creation time T
- * B. Direct Key Signature (packet type ID 2), version 6, signature type 0x1f, hashed subpackets include:
 - signature creation time (probably T)
 - key flags: certify, sign
 - issuer fingerprint (fingerprint of A)
 - features: SEIPDv2
 - preferred AEAD ciphersuites: AES-256+OCB ## TBD: do we need this?
 - no expiration subpacket
- * C. "Fallback" Public Subkey Packet (packet type 14), version 6, public key algorithm ML-KEM-768+X25519 (algorithm ID 35) (see Section 4.2 of [I-D.ietf-openpgp-pqc-14]), creation time T
- * D. Subkey binding signature (packet type ID 2), version 6, signature type 0x18, hashed subpackets include:
 - signature creation time (probably matching time in B)
 - key flags: encrypt to storage, encrypt to comms
 - issuer fingerprint (fingerprint of A)
 - no expiration subpacket
- * E. "Rotating" Public Subkey Packet (packet type 14), version 6, public key algorithm ML-KEM-768+X25519 (algorithm ID 35) (see Section 4.2 of [I-D.ietf-openpgp-pqc-14]), creation time T or later
- * F. Subkey binding signature (packet type ID 2), version 6, signature type 0x18, hashed subpackets:
 - signature creation time (same as creation time of E)

- key expiration time: max_rd (by convention, 10 days = 864000 seconds)
- key flags: encrypt to comms
- issuer fingerprint (fingerprint of A)

2.3. Certificate Size

An outbound certificate is 2938 octets in binary form.

A certificate for a peer should be merged into the locally cached certificate which may thus contain multiple rotating subkeys and grow accordingly.

2.4. Use of OpenPGP format

OpenPGP has two different packet framing formats: the "OpenPGP format" (Section 4.2.1 of [RFC9580]) and the "Legacy format" (Section 4.2.2 of [RFC9580]). Autocrypt v2 certificates and TSKs use only the OpenPGP format.

This is particularly relevant for the deterministic ratcheting step described in Section 4.1.1 because that step needs a canonical octet-string representation of several OpenPGP packets.

3. Identifying an Autocrypt v2 Transferable Secret Key

Peers interacting with an Autocrypt v2 certificate do not need to identify it as an Autocrypt v2 certificate at all. They simply need to apply common OpenPGP semantics to the certificate.

However, all UMAs operated by the keyholder need to identify an Autocrypt v2 Transferable Secret Key (TSK) in order to perform aligned key ratcheting Section 4.1.1 and achieve reliable deletion for the keyholder.

3.1. Identification By TSK Structure

A Transferable Secret Key (TSK) is identified as an Autocrypt v2 TSK if its internal structure corresponds to the packets defined in Section 2.2. The keyholder's UMA must recognize this structure to perform the aligned key ratcheting necessary for reliable deletion.

3.2. Determining min_rd and max_rd

The subkey rotation cadence is derived from the Key Expiration Time subpacket (Section 5.2.3.13 of [RFC9580]) found in Packet F of Section 2.2.

By definition, the value of the Key Expiration Time subpacket (which is measured as an integer number of seconds) is max_rd. The default value for max_rd is 864000 seconds. max_rd represents the maximum possible time that the peer can send a reliably deletable message to the keyholder, starting from when the peer retrieves a fresh copy of the keyholder's certificate.

min_rd is derived from max_rd. By convention, min_rd is half of max_rd. In particular, when max_rd is the default value of 864000 seconds, min_rd is 432000 seconds. min_rd represents the minimum possible time that the peer will be able to send a reliably deletable message to the keyholder, starting from when the peer retrieves a fresh copy of the keyholder's certificate.

Anyone designing a similar system with a different ratcheting cadence where min_rd that is anything other than half of max_rd MUST explicitly coordinate min_rd somehow with all other UMAs and MUST set max_rd to something other than 864000.

4. Reliable Deletion Strategy

An Autocrypt v2 certificate evolves over time, as new rotating encryption subkeys are added to it. It also sees older rotating encryption subkeys expire and potentially be removed. This requires reasonable behavior by the keyholder and their peers.

4.1. Keyholder Certificate and Secret Key Management

The keyholder must update the certificate regularly by ratcheting its secret key material forward to a new subkey. Since the ratchet is deterministic, based on time and the old key material, and the ratcheting schedule is standardized, each UMA that the keyholder uses will ratchet forward in the same way, without needing any additional network coordination.

4.1.1. Subkey Ratchet

Each successive encryption-capable subkey is derived deterministically from the subkey before it.

This section describes how to derive the secret key material and a deterministic subkey binding signature for the new encryption-capable subkey based on the following values:

- * `secret_key`, the Transferable Secret Key to be ratcheted.
- * `start`, the creation timestamp for the new subkey, represented as a number of seconds elapsed since 1970-01-01T00:00:00Z, as a big-endian, 32-bit unsigned integer.

Note that both `start`, and the non-secret parts of `secret_key` (that is, `secret_key` with `get_public()` applied to all its Secret Key packets) are publicly visible.

The ratchet function relies on several deterministic subfunctions:

- * `get_primary_key(TSK)` -> `K` takes a Transferable Secret Key and returns the Secret Key packet of its primary key.
- * `get_last_rotating_subkey(TSK)` -> (`K`, `sbs`) takes a Transferable Secret Key and returns the Secret Subkey packet and corresponding Subkey Binding Signature packet of its latest-expiring subkey that is marked with "encrypt to comms"
- * `extract_secret_key_material(K)` -> `M` retrieves 96 octets of secret key material `M` from an OpenPGP ML-KEM-768+X25519 secret key packet `K`. The octets of `M` are structured exactly as they are on the wire. (32 octets of X25519 key material + a 64 octet seed of ML-KEM-768).
- * `get_public(K)` -> `P` takes an OpenPGP Secret Key (or Subkey) packet and produces the corresponding OpenPGP Public Key (or Subkey) packet in OpenPGP format.
- * `make_secret_subkey(M,T)` -> `K` takes 96 octets of secret key material `M` and OpenPGP timestamp `T` and produces a ML-KEM-768+X25519 secret subkey packet with creation time `T`.
- * `normalize_x25519_scalar(M)` -> `M` takes 96 octets of OpenPGP ML-KEM-768+X25519 secret key material, and normalizes the first 32 octets, which are an X25519 secret key. The full 96 octets are returned after normalization. Normalization clears the three least-significant bits of the first octet, clears the most-significant bit of the 32nd octet, and sets the second-most-significant bit of the 32nd octet, as described in `decodeScalar25519` in Section 5 of [RFC7748].

- * `get_hashed_subpackets(S)` -> `subpackets` takes a Signature packet and returns the ordered list of hashed subpackets from that Signature.
- * `get_expiration_duration(S)` -> `secs` takes a Signature packet with a hashed Key Expiration subpacket and returns the contents of the hashed Key Expiration subpacket, represented as four octets, interpretable as a big-endian unsigned integer number of seconds.
- * `replace_creation_time(subpackets, T)` -> `subpackets` takes an ordered list of OpenPGP signature subpackets `sps` and returns the same list but with the value of the Signature Creation Time subpacket replaced by `T`.
- * `bind_subkey(P,K,T,sps,salt)` -> `S` takes primary Ed25519 secret key `P`, public subkey `K`, timestamp `T`, an ordered list of OpenPGP signature subpackets `subp`, and a 16-octet salt, and produces an OpenPGP v6 Subkey Binding Signature `S` using an Ed25519 signature. Ed25519 signatures are deterministic as described in Section 8.2 of [RFC8032].

The next secret subkey and subkey binding signature are derived via [HKDF] using SHA2-512, with the following construction, where `||` represents concatenation:

```
AC2_Ratchet(secret_key
            start)
    -> (next_secret_subkey, subkey_binding_sig):
    primary_key = get_primary_key(secret_key)
    (base_subkey, oldsubs) = get_last_rotating_subkey(secret_key)
    max_rd = get_expiration_duration(oldsubs)

    salt = start || get_public(base_key)
    info = "Autocrypt_v2_ratchet" || get_public(primary_key) || max_rd
    IKM = extract_secret_key_material(base_key)
    IKM = normalize_x25519_scalar(IKM)
    L = 160
    ks = HKDF_SHA2_512(salt, info, IKM, L)
    bssalt = SHA2_512(ks[0:64])[0:16]
    new_secret = normalize_x25519_scalar(ks[64:160])
    new_subkey = make_secret_subkey(new_secret, start)
    subp = replace_creation_time(get_hashed_subpackets(oldsubs), start)
    binding_sig = bind_subkey(primary_key,
                             get_public(new_subkey),
                             subp, bssalt)
    return (new_subkey, binding_sig)
```

The "Autocrypt_v2_ratchet" string is 20 octets as represented in US-ASCII.

```

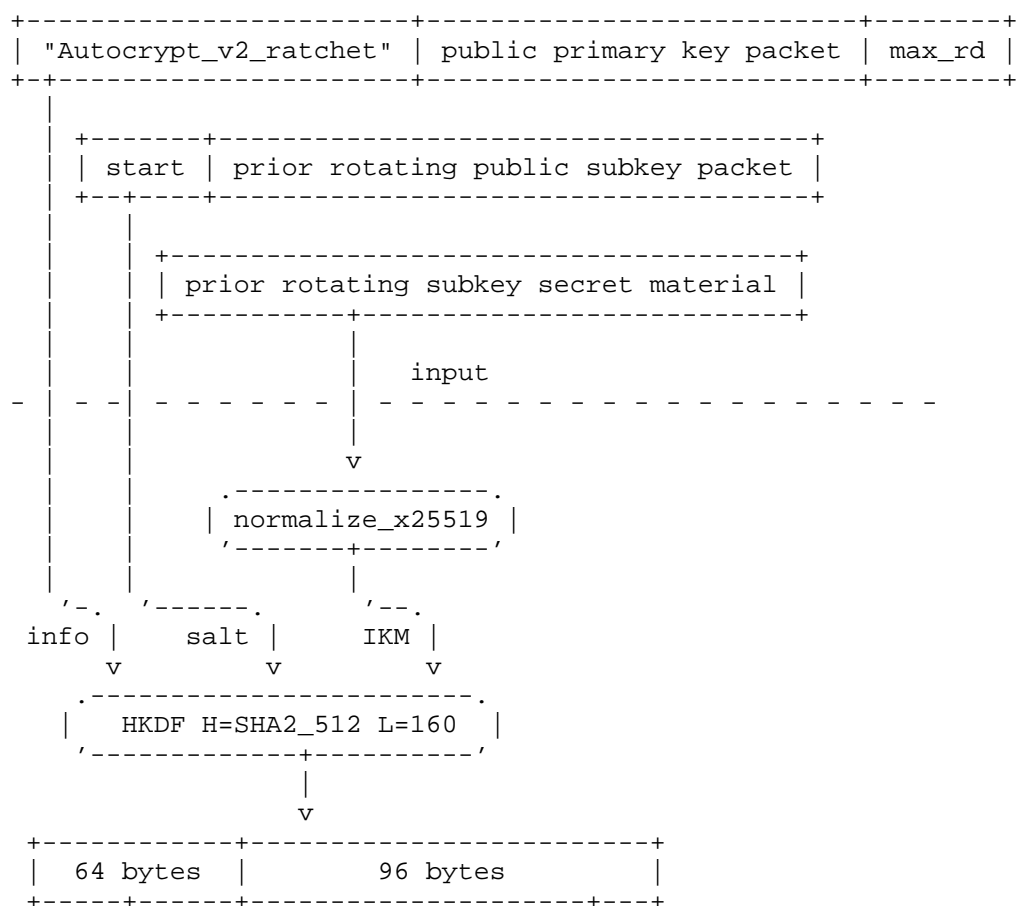
000  41 75 74 6f 63 72 79 70 |Autocryp|
008  74 5f 76 32 5f 72 61 74 |t_v2_rat|
010  63 68 65 74              |chet|
014

```

Note that a UMA without access to the secret key material of the primary key can still use parts of AC2_Ratchet to derive the new secret key subkey without producing the subkey binding signature. Such a UMA would be able to decrypt incoming new messages.

4.1.1.1. AC2_Ratchet Diagram

The core of the algorithm above can be visualized as follows:



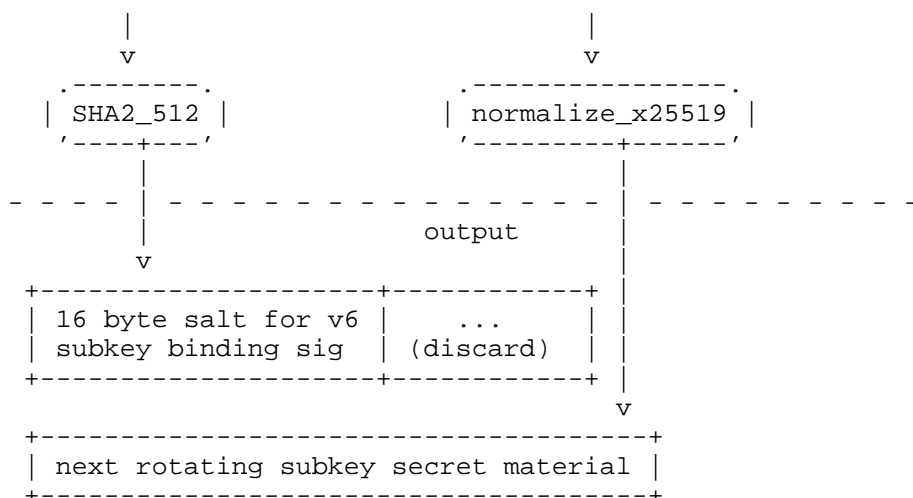


Figure 2: AC2_Ratchet Diagram

4.1.2. Ratcheting Schedule

This section describes a straightforward algorithm for a keyholder's UMA to decide when to create a new encryption-capable subkey. A UMA can execute this algorithm at any time, and SHOULD execute it in at least two specific cases:

- * When preparing to extract an OpenPGP certificate for publication or transmission to a peer, and
- * When receiving an encrypted message that is encrypted to a key that the implementation does not know about.

Note that this algorithm is specifically about new subkey creation. Please see Section 4.1.4 for recommendations about subkey destruction.

Knowing when to produce a new secret subkey requires knowledge of four distinct values. All timestamps referenced here are values computed in the OpenPGP standard, that is, as an integer number of seconds elapsed since 1970-01-01T00:00:00Z.

Two values are taken from the Autocrypt v2 TSK:

- * `base_start`, the creation timestamp of the most recent rotating subkey.

- * `max_rd`, the number of seconds that the rotating subkey will expire.

One value is derived from the above:

- * `min_rd` is typically half of `max_rd` (see Section 3.2)

And a final value is taken from general consensus:

- * `now`, the current "wall-clock" timestamp.

Add new rotating subkeys with the following routine:

1. If `now < base_start`, abort. (something is probably wrong with the system clock or the secret key material)
2. Let `next_start` be `base_start + max_rd - min_rd`.
3. If `now < next_start`, terminate successfully (no need to ratchet).
4. Generate new secret subkey `next_key`, using key material deterministically derived from the certificate's primary key, the most recent rotating secret subkey, `max_rd`, and `next_start`. Bind `next_key` as an encryption-capable subkey, using a deterministic subkey binding signature packet. `next_key`'s creation timestamp (and the subkey binding signature) is `next_start`. See Section 4.1.1 for how to deterministically derive the new secret key and subkey binding signature.
5. Restart this process, using the new subkey.

Note that the recursive nature of the above scheduler may end up generating multiple secret subkeys if the device has been offline for a long time.

4.1.3. Custom Ratcheting Schedules

While this specification defines a 10-day validity window with a 50% overlap as the standard convention, a system could be designed with a different rotation window or a different overlap algorithm. Such a system **MUST** guarantee that all the keyholder's own UMAs follow an identical adjusted rotation schedules and key destruction logic in order to maintain synchronization.

A peer receiving an incoming certificate produced under a custom schedule does not need to be aware of the keyholder's specific rotation logic. Because these certificates adhere to standard OpenPGP structures, the peer will properly handle them by merging the

new subkeys into their local cache using standard OpenPGP semantics (see Section 4.2). As long as the primary key remains constant, the peer's UMA will simply see a sequence of valid encryption subkeys and can continue to encrypt messages following the logic described in Section 4.3.

4.1.4. Timing of Secret Key Destruction

To achieve reliable deletion, a keyholder SHOULD destroy the secret key material for a rotating encryption subkey as soon as it is no longer required for decryption. Because message delivery is not instantaneous, a subkey should be retained for a grace period beyond its validity window. This delay accounts for the maximum expected transit time of the underlying transport mechanism. For example, a 10-day delay (864000 seconds) is reasonable for Internet mail.

The destruction process is tied to the successful retrieval and processing of messages. A UMA SHOULD follow this routine:

1. **Track Transport Endpoints:** For every transport endpoint associated with a certificate, the UMA maintains a `last_checked` timestamp and an expected maximum delivery delay.
2. **Determine the Safety Horizon:** The UMA calculates an `oldest_update` timestamp, which is the minimum value of (`last_checked` - delay) across all associated transport endpoints.
3. **Destroy Expired Material:** Any rotating secret subkey with an expiration time earlier than the `oldest_update` is destroyed.

In cases of persistent transport unreachability, a UMA must balance the need for decryption with the goal of reliable deletion. If a transport endpoint remains inaccessible for a prolonged period, the UMA SHOULD eventually disassociate that endpoint from the certificate. Failing to do so would prevent `oldest_update` from advancing, indefinitely stalling key destruction and widening the window of recoverability (see Section 5.4). Once the UMA gives up on fetching messages from an inaccessible transport, it can proceed with the standard destruction of the corresponding secret key material.

4.1.5. Autocrypt v2 TSK and Certificate Timeline

The following diagram illustrates the timeline for evolution of an Autocrypt v2 TSK and Cert.

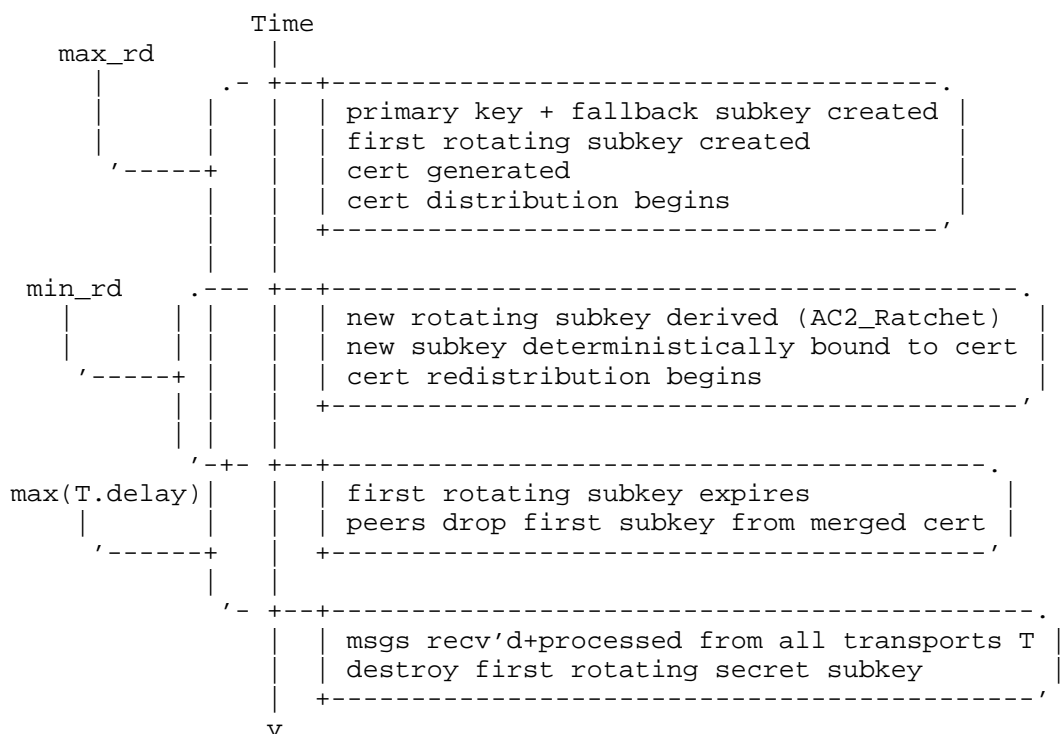


Figure 3: Autocrypt V2 TSK and Certificate Timeline

4.2. Receiving and Merging Certificates

An outbound Autocrypt v2 certificate always has the same primary key, but new encryption-capable subkeys appear on a regular basis. A peer UMA that encounters a certificate SHOULD merge the known subkeys into the locally cached certificate, using a mechanism like `sop merge-certs` (Section 5.2.6 of [I-D.dkg-openpgp-stateless-cli]).

A peer that replaces but does not merge certificates may end up encrypting a message to a subkey with a wider window of recoverability than necessary.

4.2.1. Minimization and pruning of Autocrypt v2 certificates

To ensure good data hygiene and minimize storage use, implementations should regularly prune Autocrypt v2 certificates, for example when merging incoming certificates. According to the ratcheting schedule algorithm (Section 4.1.1), a merged Autocrypt v2 certificate typically can contain at most two valid rotating subkeys at any given time.

Because expired subkeys can no longer be used for encryption, it is RECOMMENDED that implementations drop all expired encryption subkeys (and their corresponding subkey binding signatures) from certificates on a continuous basis. This maintenance strategy keeps the certificate size minimal while allowing communication with reliable deletion.

Note that deletion of secret key material (as opposed to the public key material in certificates) happens at a different cadence, see Section 4.1.4.

4.3. Encrypting to an Autocrypt v2 Certificate

When encrypting a message to an Autocrypt v2 certificate, the peer should encrypt the message to only one of the encryption-capable subkeys.

If any rotating encryption subkey is valid, the peer MUST NOT encrypt to the fallback encryption subkey.

If no rotating encryption subkey is valid, and the peer has no way to update the certificate, the peer MAY encrypt the message to the fallback encryption subkey.

If more than one rotating encryption subkey is currently valid, the peer SHOULD encrypt to the valid rotating encryption subkey with the nearest revocation date. This hastens the time when the message becomes reliably deletable.

4.4. Message Deletion

When a message is to be deleted, the UMA MUST delete all known copies of the message, as well as any set-aside session keys. It MUST also remove traces of the message from any index it may have created, as indexing tends to create an equivalent copy of the indexed content. The message will be unrecoverable by the adversary once the rotating subkey's secret key material has been destroyed (see Section 4.1.4).

4.4.1. Keyholder Processing of Encrypted Messages

To facilitate robust deletion of some messages while retaining the ability to read others from its archive, a UMA needs to plan how to process a message upon ingestion.

When the keyholder first receives an encrypted message, their UMA typically places the message in an archive visible to the UMA. If the archive contains only the original ciphertext as received, then the message will be useless in the archive when the rotating subkey's

secret key material is destroyed (see Section 4.1.4). The goal is to have the message available in the archive when access is desired, and to be able to safely remove it from the archive when deletion is desired.

There are at least three sensible ways to handle the message so that it can be deleted safely in the future:

- * Archiving cleartext
- * Stashing session keys
- * Re-encrypting

4.4.1.1. Archiving Cleartext

The simplest model is that the keyholder's UMA retains the cleartext of each message, entirely discarding the in-transit form.

The UMA can re-visit the content of such a message trivially, regardless of whether the asymmetric secret key used for decryption has been destroyed or not.

This approach presumes that the message archive is not typically accessible to the adversary.

4.4.1.2. Stashing Session Keys

In this approach, the keyholder's UMA retains the in-transit form of the message, but associates the OpenPGP session key of the encrypted content with the message. For example, the UMA could store the session keys in a separate look-aside table, indexed by Message-ID.

In this case, the UMA can access the cleartext of the message by decrypting it with the set-aside session key, even when the asymmetric secret key has been destroyed.

This approach can be used where the adversary has regular access to the archive, effectively treating the archive as though it were as at-risk as the message in transit. But the area where the session keys are stored MUST NOT be typically accessible to the adversary. If message cleartext is indexed, the session key storage MUST be at least as secure from the adversary as the message index.

4.4.1.3. Re-Encrypting

Finally, if the UMA retains some other long-term secret key for archival access, it can process each message by re-encrypting it to the long-term archival key. This can be done either by prepending each OpenPGP Encrypted Message with a new Public Key Encrypted Session Key (PKESK) packet (See Section 5.1 of [RFC9580]) that contains the existing message's session key, or by unwrapping the SEIPD packet entirely, and re-encrypting it to the long-term archival key.

This approach presumes that the message archive is not typically accessible to the adversary. If the archive were regularly accessible to the adversary, the adversary could store the re-encrypted message before deletion. Then, during the compromise of the user's long-term key, the adversary could unlock their copy of the previously deleted message.

4.4.2. Summary of Message Processing Strategies

Archiving Strategy	raw message retained?	simple indexing	OpenPGP packet handling	Deletable if adversary can access archive
Archived Cleartext	N	Y	N	N
Stashed Session Key	Y	N	N	Y
Re-encrypted Message	N	N	Y	N

Table 1: Message Archiving Strategies

5. Security Considerations

See the discussion of quantum impersonation in Section 1.4.

A message encrypted to the long-term fallback encryption subkey can not be reliably deleted, because an adversary who captures the message and in the future exfiltrates the keyholder's secret key material will be able to decrypt their stored copy of the message.

5.1. Reliable Deletion Threat Model

The threat model for reliable deletion of messages has three core expectations:

- * The adversary has access to any message in transit, and can retain the in-transit form of each message indefinitely.
- * At some point in the future, the adversary can compromise the keyholder's UMA to be able to get access to their secret key material.
- * The archive maintained by the keyholder's UMA is subject to the same level of threat as the keyholder's secret key storage.

The goal for the keyholder is that when the adversary compromises either the keyholder's archive or the keyholder's secret key material (or both), any message that has been deleted by the keyholder is in fact unrecoverable by the adversary.

5.2. The Cleartext Is The Targeted Asset

Some "Forward Secrecy" schemes emphasize key ephemerality so heavily that they lose track of the adversary's goal.

This document uses the "Reliable Deletion" framing because the adversary wants to compromise the confidentiality of the message itself, which means the key is only an intermediate target.

Many UMAs are effectively systems both for messaging and for archiving. For archiving UMAs, exfiltration of the archive is often as easy as (or even easier than) exfiltration of the secret key. This means that confidentiality protection needs to consider being able to delete the message cleartext from the archive as well as the relevant secret key material.

5.3. Key Destruction: Time-based vs. Message-based

Automated encryption key destruction is a necessary component to achieving reliable deletion. There are two main approaches to scheduling key destruction: time-based and message-based.

Some messaging protocols (e.g., [Double-Ratchet]) ratchet keys on every message (or nearly every message). This message-based key destruction schedule gives a narrower window of temporal availability for each key, and fewer messages are encrypted to each key.

The time-based approach used in this specification is simpler and relies primarily on all UMAs controlled by a single keyholder to have a shared sense of the global clock.

While the temporal window of key availability is often wider in the time-based approach, its simplicity means much less inter-client coordination. And the narrower temporal window of key validity offered by the message-based approach is often not the limiting factor in the temporal window of message recoverability (see Section 5.4).

5.4. The Window of Recoverability

Any message system offering reliable deletion has a frame of time during which the message cannot be robustly deleted because it can be recovered by an attacker. For example, in a scheme where each message is encrypted to its own unique key, if one of the recipients of the message has not yet received the message, a copy of that key must be available on the lagging recipient's device.

An adversary who is capable of:

- * capturing a copy of the message in transit, and
- * preventing its delivery to one of the recipients, and
- * compromising the recipient's device

will be able to recover the message cleartext even if all other parties involved with the message have deleted it long ago.

Realistically, a message is only deleted once every cleartext copy of it has been destroyed, and every system with access to the message's secret key has destroyed the secret key. At a minimum, a message that is in flight is not yet deletable, and for a message sent to a group, message deletion is only as robust as the slowest, most detached and uncoordinated member of the group.

Automated message deletion policies (such as "disappearing messages" functionality common in many messenger apps) can assist in message deletion, but they are similarly bound by the duration of secret key retention.

Schemes with rapid key rotation tend to need more complex internal state, and are more likely to result in unreadable messages. For initial messages, these schemes might require all parties to a communication to be online at the same time, or require any offline participants to distribute some introductory key material ("prekeys")

to a reliably online third party. A "prekey" scheme is itself at risk of attacks that can cause initial messages to lose the desired deletability property, or can expose additional metadata to an attacker (see, for example, [PREKEY-POGO]).

This document accepts a wider window of message recovery (at most 20 days, by default) which lets implementers avoid the need for any network synchronization between the keyholder's UMAs. This tradeoff provides a decentralized and robust way to achieve reliable deletion.

5.5. System Clock Reliance

A wrong system clock can have three serious impacts on a system interacting with Autocrypt v2 keys and certificates. A UMA should confirm that its system clock is within a reasonable range of the global consensus on what time it is.

There are at least three possible security-relevant failures that could happen as a result of a broken system clock:

- * With a system clock set too far in the past, a UMA might encrypt a message to an actually-expired encryption subkey. If the recipient has already destroyed their secret key material according to schedule, such a message would be undecryptable.
- * With a system clock set too far in the future, a UMA might decide that no rotating encryption subkey is available for a peer, and therefore encrypt a message to the fallback encryption key, thereby losing the reliable deletion property for that message.
- * With a system clock set too far in the future, the UMA of a keyholder with an Autocrypt v2 secret key might ratchet their secret key very far forward and destroy secret key material that is still being used to encrypt messages to them. This would render all incoming messages undecryptable.

5.5.1. Avoiding System Clock Skew

There are several ways to try to ensure that the local system clock matches the general consensus about what time it is. Broadly, the UMA (or the device on which it runs) can glean an understanding about the consensus time from dedicated time servers, from servers that it otherwise communicates with directly, or from the peers that send it messages.

An attacker in control of any of these sources of time consensus might use their influence to try to defeat reliable deletion (e.g., by forcing a message to be encrypted to the fallback key, or by

delaying secret key deletion), or to create a denial of service (e.g., by encouraging encryption of messages to an already deleted subkey, or by encouraging early deletion of a secret key).

One approach to learning about the local clock's skew from consensus time is to use [NTP] against one or more known-good time servers.

Another approach is to glean a reasonable time bounds from the transport layer the UMA connects with. For example, when using [TLS] (or [QUIC] with the [TLS] handshake), it's possible to infer a range of plausible times the server operator thinks it could be by looking at the Validity member (from notBefore to notAfter) of the server's [X.509] certificate. Narrower Validity members are more common today than they were in the past and can help to detect a clock skew that is greater than the bounds of Validity.

To the extent that the server participates in [Certificate-Transparency], the UMA could also determine reasonable time bounds from timestamp member of the SignedCertificateTimestamp extension. If the server offers a stapled [OCSP] response via the OCSP Status Request extension, the various timestamps (e.g. producedAt and thisUpdate and nextUpdate) in a stapled OCSPResponse can also be used as an estimate of the server's rough sense of the wall clock. Estimates like these based on information provided from the server in the TLS handshake are valuable because the UMA has typically not yet authenticated to the server during the handshake, which makes a targeted attack from the server in this context less likely.

The UMA can also get a rough estimate of their correspondents' view of the wall clock time by looking, for example, at the Date header in recently-received e-mail messages.

These methods can generate estimates of the likelihood of clock skew. In the event that the keyholder's UMA believes that its clock is substantially advanced beyond the consensus time, it MAY postpone secret key deletion until it is more confident in its clock alignment.

6. Usability Considerations

The keyholder needs to regularly update their certificate, and re-transmit it to those peers who might want to write them a confidential message.

Refreshing the encryption-capable subkeys might be done with a suitable invocation of `sop update-key` (see Section 5.2.5 of [I-D.dkg-openpgp-stateless-cli]).

After subkey rollover, re-transmission to peers might be done with [Autocrypt] or some other in-band communication process. Alternately, if the keyholder's peers refresh certificates before sending mail, the updated certificate could be uploaded to keyservers using [HKP] or a comparable protocol.

If a peer is willing to encrypt to an expired encryption key, they will create a message that the keyholder cannot decrypt, because the keyholder regularly destroys the secret key material associated with expired subkeys. The peer MUST NOT encrypt to an expired encryption key.

7. IANA Considerations

This document does not require any action from IANA.

8. References

8.1. Normative References

- [HKDF] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/rfc/rfc5869>>.
- [I-D.ietf-openpgp-pqc-14] Kousidis, S., Roth, J., Strenzke, F., and A. Wussler, "Post-Quantum Cryptography in OpenPGP", Work in Progress, Internet-Draft, draft-ietf-openpgp-pqc-14, 13 November 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-pqc-14>>.
- [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>>.
- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/rfc/rfc7748>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/rfc/rfc8032>>.

- [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>>.

8.2. Informative References

- [Autocrypt]
Breitmoser, V., Krekel, H., and D. K. Gillmor, "Autocrypt - Convenient End-to-End Encryption for E-Mail", 10 February 2019, <<https://autocrypt.org/>>.
- [Certificate-Transparency]
Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <<https://www.rfc-editor.org/rfc/rfc6962>>.
- [Double-Ratchet]
Perrin, T., Marlinspike, M., and R. Schmidt, "The Double Ratchet Algorithm", 4 November 2025, <<https://signal.org/docs/specifications/doubleratchet/>>.
- [HKP]
Shaw, D., Gallagher, A., and D. Huigens, "OpenPGP HTTP Keyserver Protocol", Work in Progress, Internet-Draft, draft-gallagher-openpgp-hkp-09, 3 November 2025, <<https://datatracker.ietf.org/doc/html/draft-gallagher-openpgp-hkp-09>>.
- [I-D.brown-pgp-pfs]
Brown, I. and B. Laurie, "Forward Secrecy Extensions for OpenPGP", Work in Progress, Internet-Draft, draft-brown-pgp-pfs-03, 8 October 2001, <<https://datatracker.ietf.org/doc/html/draft-brown-pgp-pfs-03>>.
- [I-D.dkg-openpgp-stateless-cli]
Gillmor, D. K., "Stateless OpenPGP Command Line Interface", Work in Progress, Internet-Draft, draft-dkg-openpgp-stateless-cli-15, 2 January 2026, <<https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-stateless-cli-15>>.

- [NTP] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/rfc/rfc5905>>.
- [OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, DOI 10.17487/RFC2560, June 1999, <<https://www.rfc-editor.org/rfc/rfc2560>>.
- [PREKEY-POGO] Gegenhuber, G., Frenzel, P., Gnther, M., and A. Judmayer, "Prekey Pogo: Investigating Security and Privacy Issues in WhatsApp's Handshake Mechanism", arXiv, DOI 10.48550/ARXIV.2504.07323, 2025, <<https://doi.org/10.48550/ARXIV.2504.07323>>.
- [QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/rfc/rfc9000>>.
- [TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/rfc/rfc8446>>.
- [X.509] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/rfc/rfc5280>>.

Appendix A. Historical notes

Prior work on "forward secrecy" in OpenPGP dates back decades, including [I-D.brown-pgp-pfs]. Much of the guidance in that document is useful, and it is worth reading. As far as the authors are aware, the mechanisms described there were never widely adopted, and no tooling is currently available to make certificates compliant with the policies or "single-use" subpackets outlined there. That specification also never grappled with the complexities of coordinating ephemeral subkeys across multiple UMAs, or considering how "reply all" would work in a group messaging context for single-use keys.

Appendix B. Design Choices

B.1. Subkey Validity Windows

When designing the subkey rotation mechanism, it's possible to create back-to-back, overlapping, or superset validity windows.

B.1.1. Back-to-Back Validity Windows

With back-to-back validity windows, Each subkey is valid for a period that doesn't overlap with any other subkey.

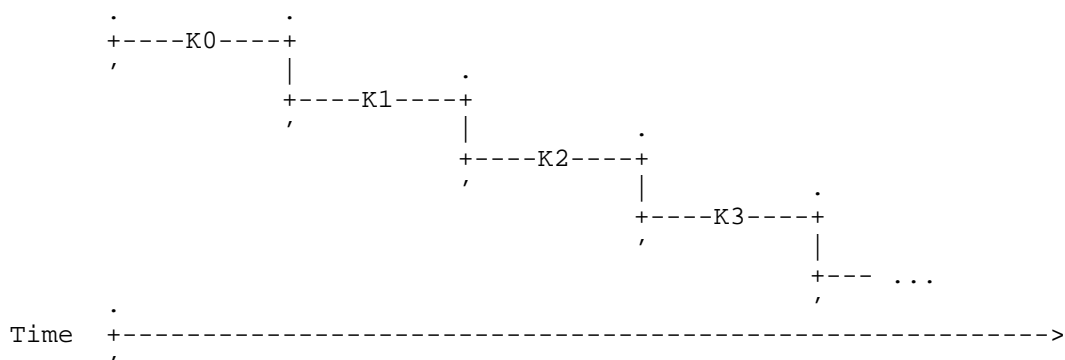


Figure 4: Back-to-Back Validity Windows

In this arrangement, it is unambiguous which subkey to encrypt to, and validity windows can be shorter. But toward the end of one subkey's validity window, the keyholder needs to distribute the next subkey as well as the current subkey, in case an incoming message is produced after the cutover. When the keyholder does this, they are distributing two subkeys (which makes the certificate larger).

Also, the upcoming subkey will have a creation time and signature creation time in the future, and peer implementation support for subkeys with creation times or subkey binding times in the future is dubious. Some peers may reject the subkey for being "from the future", and strip it before storing the certificate. Those peers won't have it available when they need it, and might end up having to encrypt to the fallback subkey. Other peers may ignore the creation timestamps, and go ahead and encrypt to the subkey before its validity window begins.

Policy needs a decision about when to start distributing the new subkey, but that decision does not affect the wireformat.

If keyholder's UMA X distributes the new subkey earlier than keyholder's UMA Y, and a peer encrypts a message to the new subkey before the new subkey is valid, keyholder's UMA Y may need to trial-ratchet until it finds the right subkey.

B.1.2. Overlapping Validity Windows

With overlapping validity windows, each subsequent rotating subkey's validity starts during the validity window of the prior rotating subkey.

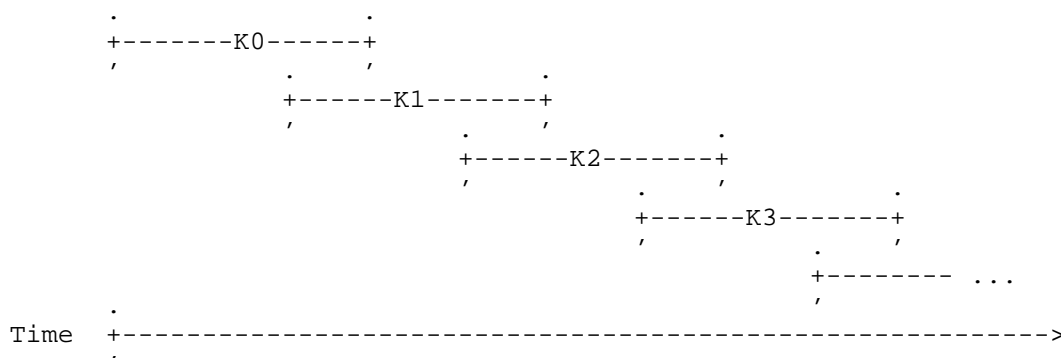


Figure 5: Overlapping Validity Windows

The validity window for any subkey in this scheme is wider than back-to-back. Keyholder only needs to distribute a single rotating subkey.

Policy needs a decision about how much the validity windows should overlap, which affects the wireformat. This document records the convention that the validity windows overlap by 50% (new subkey created halfway through the prior subkey's validity).

The keyholder's UMAs can blindly coordinate subkey distribution by only distributing subkeys whose validity window is currently active.

B.1.3. Superset Validity Windows

With superset validity windows, each rotating subkey has an identical creation time, but each subsequent subkey has a later expiration date.

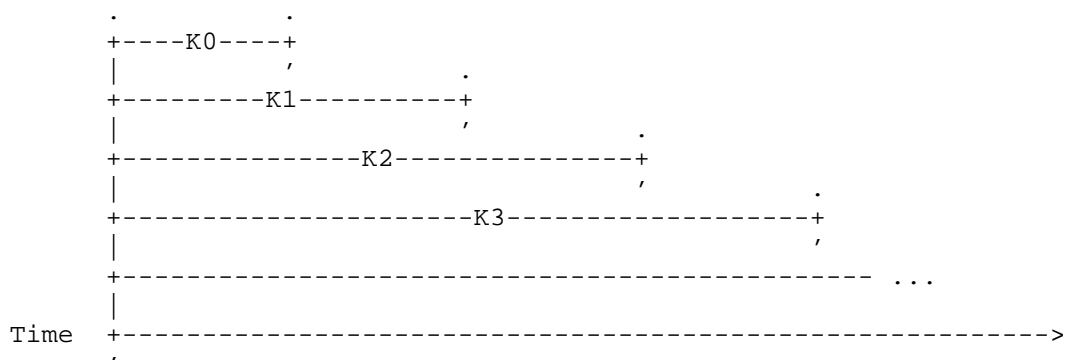


Figure 6: Superset Validity Windows

This scheme relies on the keyholder's machines delaying generation of each subsequent rotating subkey, as a prematurely-generated (and distributed) subkey might get used by a peer.

B.1.4. Validity Window Conclusion

Decision: better to go with overlapping, so that the common case is a certificate with only a single rotating subkey.

B.2. Ed25519 for Primary Key

Why not go with PQ/T primary key? Because signatures from all PQ/T keys are very large, and each cert needs at least three signatures in it.

Why not go with Ed448 primary key? Because Ed25519 implementations have endured significantly more scrutiny than Ed448 implementations. Also, Ed25519 is already mandatory-to-implement in OpenPGP, and signatures are marginally smaller.

B.3. Balancing Rotation and Reliable Deletion Timing

The default 5-day rotation period, 10-day expiration, and 20-day deletion schedule balance reliability with security. The 10-day buffer after expiration accounts for potential message transmission delays in systems like SMTP. Deleting the secret key material after 20 days ensures that reliable message deletion occurs in the near future. This timeline prevents secret keys from being stored longer than necessary while remaining compatible with the pace of modern email.

B.4. No Explicit Annotations

An Autocrypt v2 certificate is identifiable by its structure and parameterization alone. There is no attempt to mark a certificate explicitly as an attempt at Autocrypt v2.

The design goal of interop with peers with a merely upgraded OpenPGP implementation suggests that an explicit indicator would be unnecessary.

B.5. Seed-free Ratcheting

When ratcheting a secret subkey forward, this design only uses secret entropy from the current ratcheting subkey's secret key material.

Introducing additional data in the form of a seed would require special coordination between cooperating peers for the initial synchronization of that seed, which complicates the goal of minimizing network synchronization.

For example, a keyholder that adds another UMA to their account currently needs to transfer an OpenPGP TSK to the new UMA. How would a distinct seed be transmitted to the other UMA in such a way that it would not be accidentally re-transmitted to a peer by legacy tooling that extracts an OpenPGP certificate from the TSK?

Another choice of additional seed material could be the secret key material of the primary key itself, or of the fallback's secret key material. By not including any of that secret key material in the input to the ratchet, we enable the keyholder to distribute these two long-term keys to their various UMAs via a tamper-resistant hardware device.

Appendix C. Test vectors

C.1. Key and Certificate Created

At 2025-11-01T17:33:45Z, this Autocrypt v2 secret key was created, with the associated certificate.

C.1.1. Initial Certificate

This outbound certificate is distributed to peers that the key holder wants to communicate with.

-----BEGIN PGP PUBLIC KEY BLOCK-----

xioGaQZEeRsAAAAGcRJVRUdqGTNMnwwTeQZv+o8XiaDdVTTPXnjPNH2FR4zCkgYf
GwgAAAAzBYJpBkR5ApsDAh4IIiEGH0FrgTQndqxcVMDB0oQzP8aKmI9bzomuqD7i
+bPY0t4DJwkCAAAAAGXjENl3uzg/tNOjW2+khGm+tpsbjJOiUZ0J9I7TZsmLWtcH
lk6y4YJfsZTJl6A4Xdf7n+t3bz8im8Z5aPRfp1SL3OrBK+2DY3GX7xBalEAg1jON
zsQKBmkGRHkjAAAEWmNqYCYyPLlqXKM8d+oKEe+almaw5IdqzQRl3fDQA8WlOlA
/Ywqa+MA9Ni7ZMBE9pBRUKefJGZOu2NuNIJSzsB4cltYPDqrMSR29JqqnOxXpISJ
VOMIxhClj5VAKqnO4AuJ8zEa3imhDxUr4JORP8eRi5gwkgu8BMQJfDA9kBgeYzQz
x5JMJ5Mh/dW8/WOLdzIE77I8L6dX2NB0v6uAV0A6eFy+LCuivxV/aDRH/9SWltDG
WocNMYqQ75JkFSpe4RVcYvVBEliO6XBInjGAm/hSzwE6RglwbpQdy3WajfYsVuqL
QgRqpZhuKmul8vaij0JGHRhAEDNJuvmbtdgnCneezGtbOPF4zcax5Da+EekANUKh
aClxZjSAFpAcpii0GELEQIPYxaWas6C2fsNxmsh57d/U0p0SAAiOrmVLrdCC6kp
CnuqRiQ8jAC0+/tmZRe3YVi6lBiNhlVXtcoUCOCH4jWiMAMB04sTb1uz+3EtAJkt
ECDJ5muyIHhCBBQMKBKl9rMCujJfIQApZlIHq8qOl2G1NfcdGfsUQ9sbnKZRYik6
5FBMIOsco+Q7XppgxyehKSmEpxJfwCtfc6YIosud5GSDOZMzCVQViDUawRk58KII
S8YDRNq9SrdqY9uLx4gq23Jyvfmkald7EsbAaOKbPQNqPaRpJTO7W9QGmFdCL4Kk
04hOoVZHLtq5Pko+/zyQick7kro5pvPDUAhVxuEu/GilZWNOR9fO4TIIIZwmIjwR
5YN0MVC7CyFEaBySPHhYZDdJJ7dj+EKncGts02cO7HmM8WUDPAAELlViZ/k10KjH
5oO+jWM6YBZrkLOPVngJSMB43EpJICEySsGc2qCNK5NMtLvNSGsvsuUdr/WDtANT
rWI2XOW9oVs9APuiKrHT+OGQTKnK+lKhUlJMEV6Iqt/6jt2ereLCzglS8BlfceG
6disSDiz6jKn/lAPsSInlcsQqYR8MwoaE5glTKx8SGFL3xK3VvMrSMED/8g4bsGa
yfeOM7lKrOIrV8nCyXrINODrW5Sgmmml02GR63l0qdybFsQZ0BFmlWWlTRFKekRo
meMB2lyZXbc/qVhqILS4loYtUhSxJfBBbClKcAlHIzxNJeJaofALanq2VbKEbnBT
pCm+0HhH8Xu+/ActZew8QliFrcoRSPM2zcy/nqPAN4QsvRFkykoDiDcId3cjwFAA
EKWKIshR5jNZoSws+fcKcGklhoYBhVY0WhQiZTSwNZsOOqaujNNhLgKZ5bebExqz
AkIrEhozFSS0ZBtzjCs1IMRchHOHzgvp3HWl0ZuwtXKu9MY1h+dOjbqx Fai7IwOs
3xC0PMN03GIWkcNL8UACrLNEFPW6sVwE1ROXcTy5Lde63WsmxipfJEtDwfEgrvxZ
97yDKfRMGILApby2CFuApkYLiQd+lFGwyTNxYbqI7pJKOd1VZ5s/7kBhlCwSedZ9
luK/3JijNotlNrt/7atasask2kKjFpRGpUvArJRJGdVJsZm5sBZ0PTWn94Zxunaz
ICo15jhCENG++qdeJ6ZbGHwMX807LWNXHUC1YtRVTUR0mSFE2mwxLPFeeRJlIOkY
FzqWfDCMwXTwllBLbyLQWhk/JW9MNw+ewkrxPD7CiwYYGwgAAAAsBYJpBkR5ApsM
IiEGH0FrgTQndqxcVMDB0oQzP8aKmI9bzomuqD7i+bPY0t4AAAAAR6cQz5l1AOOR
vovCj4Bu3ec92BkTY3LUETvAr6HFyS3+Llv9HvuzcEJ/i+cw9MKuoly0JXzHzfK
waKnA/SDe0nzYpe3CE+jGj3A4z2uDr4jIAD0xAoGaQZEeSMAAATAmCRkirW5CYNd
ucEJs50OewXTIDpd+lBRD883x65KmmPy8VABXdlc4unIE7ynMNXTJvWkV8jUE3H
URen4k9UULzRkF3uwKh46VCn94Shmw7rs5wU44QXc7ays0Ypa3XFVnjoMEo+C8N5
NqLQipIHpmJjgEjVGh8/E10g5FCTc2sx8XAJdzxpBLRclKagpAULabPjYIOowoOI
ZTs9Cx05wgy49lpEcFuvWAWbdLbxVh9tq1MdV2FJdiU8OhvIg2LHdK2KAK7h8gBQ
waSiY0n+mhy/1HJ3gcf8FxTKfMQPAh2Ym0W5oVqc1HTYAlEucgKVZZl55TX1bDnd
9QDr52rNEcf5FFIySbxZbDc/ummER8T6oIXXmbLtURKdind4WstWV4XEBGEvoB2g
65igKI+vl8bH8z0V2XznScIfKM5qNEMle5JcugGaew9BxaNkuot6dLT+8n7IoTqO
LHQzg5SxcccNwSLstWbQ2I2OXYPQcLerMwbOIdrUISFqQzXd4nZN6S0y3gFOYig
Qcmmljx5A24eaam4pJ0I4APVlYnRypCyKWrMyiOYrKQZy3cYoUpycMMKwXR2YS9q
QawPI57Wso67KsQAjYvWB397t0sRzE3FF6eRwlkIZly32rJ86MDJ4FXoQWkE2YTG
Jrh3Co8K2cWWJYjzPDdAMJ6LksEb3LXYcGPhGTlmJaGY6wWLG083oJpIeDFMXB0e
CRfO5ouqRR8196wRySod5kRKKgKiQy4PIiLzIQCuSLcnCUBwqX47FLQBkbvG0M+c
sK6NlykFVjCed4CDMyf5Acho8nH96XbdunPWWXzyas2boiLDx8HoQDdpkXepucWN

```
SQ+uoLViZLyW9Ypa+2mMCHroyRnJUywuIyUNpIB+x7KW4Fx0YD8kShJq1WBEuEH8
Am19mmUkk71jBsdLG5o5QwkxiASBRWu42IH6hcF1q5TxZqwBmDxj5AKqUxDoyBL1
3EEdlnz/+MzwcAPxZQU/+VeEdbTHfLUI1CXtJCpVSMjJ83poSxPtFHx0Anc4lwcT
+IRp8BJZtF2QCgZC8Q3I2rtsE7nIt4pZRYV0RbGvwMxznH2Z97PlcCmyccYJaXGW
kghwNb6dcInPtXr81WRxcIBx44ggWgFwkLdSho68pmg60cWcioEKxGwBBIcaQq+p
JDDoqVC2AU8cATkHdZhzi8aRXffWX4xsJ0/4a/1NI0F0mxvZRF01Qjg0ofYpkPS
OmnyjDlYKCMZRkvz1Ykv0LL0V6RhCw2uaKDhg3siZzgfBUocGwZ8kaTY7K/60qC4
x5oqIlnctCN58chMOHU7YGNzaUCvgbUTUWshIGt7yLf+AikpmTq9IjkyjLkuak0b
VCQpEwHyBSiRVqo2CFmGdzU6wR8dAFiusQfLYCLF8X8514DkEYpwsYitUIs+PMB2
UoOVulZJhbB20rrmTMSZ92EGuBivy4rJos5tI3t7Z8gRkD9N2LymR3xT2xgv9p+R
NxglSM2WlxMzED/MqDq20apCd5pBQTdEq7ep3BTGtAtO5a2Y8sxrmcQd4TsPN7Cm
MVDsUXno9U1RKjKgOI+Xwn7MXAsNJ0df2ZnmowWT1I66ccOovlX5XMLghOsmtpBD
iZiJLCICOCkRBhgbCAAAADIFgmKGRHkFCQANLwACmwQiIQYfQWuBNCD2rFxUwMHS
hDM/xoqYj1vOia6oPuL5s9jS3gAAAAB/KRDjNtQHkZBvJC02Qn1fsEFW8USzkJcd
C4BmSoNwl6iB5zKz/85ZwOhFh+mJEnoZWggS9ZM9Y1jhiJFNyAei81RcO/j8PCMC
K3H73ZjWMHxBcQ==
-----END PGP PUBLIC KEY BLOCK-----
```

C.1.2. Initial TSK

This secret key is made available only to the keyholder's own devices.

-----BEGIN PGP PRIVATE KEY BLOCK-----

```
xUsGaQZEeRsAAAAGcRJVRUdqGTNMnwwTeQZv+o8XiaDdVTTPXnjPNH2FR4wAET8I
Q3fQeNm71ldOZSkB/260JmbL1QIX/q4ugYzBOFKLCkgYfGwgAAAAzBYJpBkR5ApsD
Ah4IIiEGH0FrgTQndqxcVMDBoQzP8aKmI9bzomuqD7i+bPY0t4DJwkCAAAAAGXj
ENl3uzg/tNojW2+khGm+tpsbjJ0iUZOJ9I7TZsMLWtch1k6y4YJfsZTJl6A4Xdf7
n+t3bz8im8Z5aPRfp1SL3OrBK+2DY3GX7xBa1EAgljOnX8RrBmkGRHkjAAAEwMnQ
yYCYyPLlqXKM8d+oKEe+almaw5IdqzQRl3fDQA8Wl0lA/Ywqa+MA9Ni7ZMBE9pBR
UkefJGZOu2NuNIJSzsB4cltYpDqrMSR29JqqnOxXpISJvOmIxhClj5VAKqnO4AuJ
8zEa3imhDxUr4JORP8eRi5gwkgu8BMQJfDA9kBgEYzQzx5JMj5Mh/dW8/WOLdzIE
77I8L6dX2NB0v6uAV0A6eFy+LCuivxV/aDRH/9SW1tDGWocNMYqQ75JkFSpE4RVc
YvVBE1iO6XBInjGAm/hSzwE6RglwbpQdy3WajfYsVuqLQgRppqZhuKmul8vaij0JG
HRhAEDNJuvmbtdgncneezGtbOPF4zcax5Da+EekANUKhaClxZjSAFpAcpii0GE1E
QIPYxaWas6C2fsNxmsh57d/U0p0SAAiOrmVLrdCC6kpCNuqRiQ8jAC0+/tmZRe3
YVi6lBiNHlVXtcoUCOCH4jWiMAMB04stBluz+3EtAJkteCDJ5muyIHhCBBQMKBKl
9rMCujJfIQApZlIHq8qOl2G1NfcdGfsUQ9sbnKZRyik65FBMIosco+Q7Xppgxyeh
KSmEpxJfWctfC6Yiosud5GSD0ZMzCVQViDUawRk58KIIS8YDRNq9SrdqY9uLx4gq
23Jyvfmkald7EsbAaOKbPQNgPaRpJT07W9QGmFdCL4KkO4hOoVZHLtq5Pko+/zYQ
ick7kro5pvPDUaVhxuEu/GilZWNOR9fO4TIIItZwmIjwR5YN0MVC7CyFEaBySPHhY
ZDdJJ7dj+EKncGts02cO7HmM8WUDPAAELlViZ/k10KjH5oO+jWM6YBZrkLOPVngJ
SMB43EpJICeYsGc2qCNK5NMtLvNSGsvsuUdr/WDtANtrWI2Xow9oVs9AppuiKrH
T+OGQTKnK+lKhUlJMEV6Iqt/6jt2ereLCzglS8B1fceG6disSDiz6jKn/lAPsSiN
lcsQqyR8MwoaE5glT�x8SGFL3xK3VvMrSMED/8g4bsGayfEOM7lKr0Irv8nCyXrI
NOdrW5Sgmmml02GR63l0qdybFsQZ0BFmlWWlTRFKekRomeMB2lyZXbc/qVhqILS4
loYtUhSxJfBBbClKcAlHIzxNJeJaofALanq2VbKEbnBTpCm+0HhH8Xu+/ACtZew8
```

QliFrcoRsPM2zcy/nqPAN4QsvRFkykoDiDcId3cjwfAAEKWKIshr5jNZoSWS+fcK
cGklhoYBhVY0WhQiZTSwNZs0OqaujNNhLgKZ5beExqzAkIrEhozfSS0ZBtzjCs1
IMRchHOHzgvp3HWL0ZuwtxKu9MY1h+dOjbqx Fai7IwOs3xC0PMN03GIWkcNL8UAC
rLNEFPW6sVwElROXcTy5Lde63WsmxipfJEtDwFEqrvxZ97yDKfRMGILApbY2CFuA
PkYLiQd+lFGwytNxYbqi7pJKOd1VZ5s/7kBhlCwSedZ91uK/3JijNotlNrt/7ata
saSk2kkjFpRGpUvArJRJGdVJsZm5sBZ0PTWn94ZxunazICo15jhCENG++qdEJ6Zb
GHwMX807LWNXHUC1YtRVTUR0mSFE2mwXLPFeerJlIOkYFzqWfDCMwXTw11BLbyLQ
Whk/JW9MNw+ewkrxPD4AaGZh9fcInOX2YpJXNKENHCNh08uWEosBWRDzpeOVGja
b3dBAAafhaFGEhx4WqVYOan5ZDw/WNqlTVIRDLWvDuQjHvRMLCoqgQ/E+04G6uS6
arNh3q8Bzija4mELayyhwsGGBsIAAAALAWCaQZEeQKbDCIhBh9Ba4E0J3asXFTA
wdKEMz/GipiPW86Jrqq+4vmz2NLeAAAAAEenEM+ZdQDjkb6Lwo+Abt3gvdgZE2Ny
1BE7wK+hXckt/i5b/R77s3BCf4vnMPTCrqJctCV8x2mX5MGipwP0g3tJ82KXtwhP
oxo9wOM9rg6+IyAAx8RrBmkGRHkjAAAEWJgkZiQ1uQmDXbnBCbOdDnsF0yA6XftQ
UQ/PN8euSppqW2PFQAV3ZXOLpyB08pzDV0yblpFfi1BNx1EXp+JPVFJc0ZBd7sCo
eOlQp/eEoZs067OcFOOEF3O2srNGKwt1xVZ46DBKpgvDeTai0IqSB6ZiY4BI1Rof
PxNdIORQk3NrMfFwI3c8aWy0XJSmoKQFJWmz42CDqMKDiGU7PQsdOcIMuPdaRHBV
LlgFm3S28VYfbatTHVdhSXYlPDobyINix3StigCu4fIAUMGkiMtJ/pocv9Ryd4HH
/BcUynzEDwIdmJtFuaFanNR02AJRLnIClWWZeeU19Ww53fUA6+dqzRHH+RRSMkm8
WWw3P7pphEfE+qCF15my7VESnYp3eFkrVleFxARhL6AdoOuYoCiPr5fGx/M9Fd18
50nCHyjoAjrJtXuSXLoBmnsPQcWjSrqlenS0/vJ+yKE6jix0M4OUsXHAjVrC0rcB
20NiNj12D0ApXkTMGzia61CLBakM13eJ2TektMt4BTmIoEHJppY8eQNuHmmpuKSd
COAD1ZWJ0cqQsilqzMoJmKyKgt3GKFKcnDDCsF0dmEvakGsDyOelrKOuyrEAI2L
lgd/e7dLEcxNxRenkcJZCGZct9qyf0jAyeBV6EFpBNmExia4dwqPcTnFliWI8zw3
QDCei5LBG9y12HBj4Rk9ZiWhmOsFixtPN6CaSHgxTFwTngkXzuaLqkUfNfesEckq
HeZESioCoqsuDYiI8yEArki3JwlAcKl+OxS0AZG7xtDPnLCuJdcpBVYwnneAgzMn
+QHIApJx/el23bpz11188mrNm6Iiw8fB6EA3aZF3qbnFjUkPrqC1YmS8lvWKWvtp
jAh66MkZyVMsLtclDaSafseyluBcdGA/JEoSATVgRLhB/AJtfZplJJ05YwbHSxua
OUMJMYgEgUVruNiB+oXBdauU8WasAZg8Y+QCqlMQ6MgS5dxBHZZ8//jM8HAD8WUF
P/lXhHW0x3y1CNQl7SQQVUjIyfn6aEst7Xx8dAJ3ONCHE/iEafASWbRdkAoGQvEN
yNq7bBO5yLeKWUWfDEWxr8DMc5x9mfez9XApsnHGCWlxlpiIcDW+nXCJz7V6/NV
kCXCAceOIIFoBcJC3UoaOvKZoOthFnIqBCsRsAQSHGkKvqSQw6KlQtgFPHAE5B3WY
c6yPGkV3311+MbCdP+Gv9TSNBdJsb2UXztUI4NKH2KZD0jpp8ow5WCgpmUSr85WJ
L9Cy9FekYQsNrmig4YN7Imc4HwVKHBsGfJGk2Oyv+tKguMeaKiJZ3LQjefHITDh1
02Bjc21Ar4G1ElFrByBre8i3/gIpKZk6vSI5Moy5LmpNG1QkKRMB8gUokVaQNgH
Zhncl0SEfHQBYrrEHY2AixfF/OZeA5BGKcLGIRVCLPjzAdlKDLbtWSYwWdtK65kzE
mfdhBrgYr8uKyaLOBSN7e2fIEZA/Tdi8pkd8U9sYL/afkTcYJUjNlpcTMxA/zKg6
ttGqQneaQUE3RKu3qdwUxrQLTuWtmPLMa5nKg+E7DzewpjFXULF56PVNUSoyoDiP
l8J+zFwLDSdHX9mZ5qM1k9SOunHDqL5V+VzJYITrJrTwQ4mYiSwiAjkAoNfHW/ZS
GRonYC/yjFWxppDkKws8HKbmd4ncr6jhGHKcrDYOdOJcnNaNqffg0un/l4Fm9n4J
yoKSFRdc3rktMarVYkYyKsxQRdHrtmZhYMLox9tldjjZah5GvxE/eZguwpEGGBsI
AAAAMgWCaQZEeQUJAA0vAAKbBCIhBh9Ba4E0J3asXFTAwdKEMz/GipiPW86Jrqqg
+4vmz2NLeAAAAAH8pEOM21AerKG8kLTZCfV+wQVbxRLOqlx0LgGZKg3CXqIhNMrP/
zlnA6EWH6YkSehlaCBL1kz1jWOGKMULgB6LzVFw7+Pw8IwIrcfvdMNYwFEEJ
-----END PGP PRIVATE KEY BLOCK-----

C.2. After New Subkey Added

After 2025-11-06T17:33:45Z (min_rd after the initial key creation), a new rotating subkey is added.

The new subkey is derived via HKDF as described in Section 4.1.1, with the following parameters:

Salt:

```
000 69 0c db f9 ce c4 0a 06 69 06 44 79 23 00 00 04
010 c0 98 24 64 8a b5 b9 09 83 5d b9 c1 09 b3 9d 0e
020 7b 05 d3 20 3a 5d fb 50 51 0f cf 37 c7 ae 4a 9a
030 6a 5b 63 c5 40 05 77 65 73 8b a7 20 4e f2 9c c3
040 57 4c 9b d6 91 5f 23 50 4d c7 51 17 a7 e2 4f 54
050 52 5c d1 90 5d ee c0 a8 78 e9 50 a7 f7 84 a1 9b
060 0e eb b3 9c 14 e3 84 17 73 b6 b2 b3 46 29 6b 75
070 c5 56 78 e8 30 4a 3e 0b c3 79 36 a2 d0 8a 92 07
080 a6 62 63 80 48 d5 1a 1f 3f 13 5d 20 e4 50 93 73
090 6b 31 f1 70 23 77 3c 69 6c b4 5c 94 a6 a0 a4 05
0a0 25 69 b3 e3 60 83 a8 c2 83 88 65 3b 3d 0b 1d 39
0b0 c2 0c b8 f7 5a 44 70 55 2f 58 05 9b 74 b6 f1 56
0c0 1f 6d ab 53 1d 57 61 49 76 25 3c 3a 1b c8 83 62
0d0 c7 74 ad 8a 00 ae e1 f2 00 50 c1 a4 88 cb 49 fe
0e0 9a 1c bf d4 72 77 81 c7 fc 17 14 ca 7c c4 0f 02
0f0 1d 98 9b 45 b9 a1 5a 9c d4 74 d8 02 51 2e 72 02
100 95 65 99 79 e5 35 f5 6c 39 dd f5 00 eb e7 6a cd
110 11 c7 f9 14 52 32 49 bc 59 6c 37 3f ba 69 84 47
120 c4 fa a0 85 d7 99 b2 ed 51 12 9d 8a 77 78 59 2b
130 56 57 85 c4 04 61 2f a0 1d a0 eb 98 a0 28 8f af
140 97 c6 c7 f3 3d 15 d9 7c e7 49 c2 1f 28 ce 6a 34
150 49 b5 7b 92 5c ba 01 9a 7b 0f 41 c5 a3 4a ba 8b
160 7a 74 b4 fe f2 7e c8 a1 3a 8e 2c 74 33 83 94 b1
170 71 c0 8d 5a c2 d2 b7 01 db 43 62 36 39 76 0f 40
180 29 5e 44 cc 1b 38 80 eb 50 8b 05 a9 0c d7 77 89
190 d9 37 a4 b4 cb 78 05 39 88 a0 41 c9 a6 96 3c 79
1a0 03 6e 1e 69 a9 b8 a4 9d 08 e0 03 d5 95 89 d1 ca
1b0 90 b2 29 6a cc ca 23 98 ac a4 19 cb 77 18 a1 4a
1c0 72 70 c3 0a c1 74 76 61 2f 6a 41 ac 0f 23 9e d6
1d0 b2 8e bb 2a c4 00 8d 8b d6 07 7f 7b b7 4b 11 cc
1e0 4d c5 17 a7 91 c2 59 08 66 5c b7 da b2 7c e8 c0
1f0 c9 e0 55 e8 41 69 04 d9 84 c6 26 b8 77 0a 8f 0a
200 d9 c5 96 25 88 f3 3c 37 40 30 9e 8b 92 c1 1b dc
210 b5 d8 70 63 e1 19 3d 66 25 a1 98 eb 05 8b 1b 4f
220 37 a0 9a 48 78 31 4c 5c 13 9e 09 17 ce e6 8b aa
230 45 1f 35 f7 ac 11 c9 2a 1d e6 44 4a 2a 02 a2 ab
240 2e 0f 22 22 f3 21 00 ae 48 b7 27 09 40 70 a9 7e
250 3b 14 b4 01 91 bb c6 d0 cf 9c b0 ae 8d d7 29 05
```

```
260 56 30 9e 77 80 83 33 27 f9 01 c8 68 f2 71 fd e9
270 76 dd ba 73 d6 59 7c f2 6a cd 9b a2 22 c3 c7 c1
280 e8 40 37 69 91 77 a9 b9 c5 8d 49 0f ae a0 b5 62
290 64 bc 96 f5 8a 5a fb 69 8c 08 7a e8 c9 19 c9 53
2a0 2c 2e d7 25 0d a4 80 7e c7 b2 96 e0 5c 74 60 3f
2b0 24 4a 12 6a d5 60 44 b8 41 fc 02 6d 7d 9a 65 24
2c0 93 b9 63 06 c7 4b 1b 9a 39 43 09 31 88 04 81 45
2d0 6b b8 d8 81 fa 85 c1 75 ab 94 f1 66 ac 01 98 3c
2e0 63 e4 02 aa 53 10 e8 c8 12 e5 dc 41 1d 96 7c ff
2f0 f8 cc f0 70 03 f1 65 05 3f f9 57 84 75 b4 c7 7c
300 b5 08 d4 25 ed 24 2a 55 48 c8 c9 f3 7a 68 4b 13
310 ed 7c 7c 74 02 77 38 d7 07 13 f8 84 69 f0 12 59
320 b4 5d 90 0a 06 42 f1 0d c8 da bb 6c 13 b9 c8 b7
330 8a 59 45 85 74 45 b1 af c0 cc 73 9c 7d 99 f7 b3
340 f5 70 29 b2 71 c6 09 69 71 96 92 08 70 35 be 9d
350 70 89 cf b5 7a fc d5 64 71 70 80 71 e3 88 20 5a
360 01 70 90 b7 52 86 8e bc a6 68 3a d1 c5 9c 8a 81
370 0a c4 6c 01 04 87 1a 42 af a9 24 30 e8 a9 50 b6
380 01 4f 1c 01 39 07 75 98 73 ac 8f 1a 45 77 df 59
390 7e 31 b0 9d 3f e1 af f5 34 8d 05 d2 6c 6f 65 17
3a0 ce d5 08 e0 d2 87 d8 a6 43 d2 3a 69 f2 8c 39 58
3b0 28 29 99 44 ab f3 95 89 2f d0 b2 f4 57 a4 61 0b
3c0 0d ae 68 a0 e1 83 7b 22 67 38 1f 05 4a 1c 1b 06
3d0 7c 91 a4 d8 ec af fa d2 a0 b8 c7 9a 2a 22 59 dc
3e0 b4 23 79 f1 c8 4c 38 75 3b 60 63 73 69 40 af 81
3f0 b5 13 51 6b 07 20 6b 7b c8 b7 fe 02 29 29 99 3a
400 bd 22 39 32 8c b9 2e 6a 4d 1b 54 24 29 13 01 f2
410 05 28 91 56 aa 36 08 59 86 77 35 3a c1 1f 1d 00
420 58 ae b1 07 cb 60 22 c5 f1 7f 39 97 80 e4 11 8a
430 70 b1 88 ad 50 8b 3e 3c c0 76 52 83 95 bb 56 49
440 85 b0 76 d2 ba e6 4c c4 99 f7 61 06 b8 18 af cb
450 8a c9 a2 ce 6d 23 7b 7b 67 c8 11 90 3f 4d d8 bc
460 a6 47 7c 53 db 18 2f f6 9f 91 37 18 25 48 cd 96
470 97 13 33 10 3f cc a8 3a b6 d1 aa 42 77 9a 41 41
480 37 44 ab b7 a9 dc 14 c6 b4 0b 4e e5 ad 98 f2 cc
490 6b 99 ca 83 e1 3b 0f 37 b0 a6 31 57 52 51 79 e8
4a0 f5 4d 51 2a 32 a0 38 8f 97 c2 7e cc 5c 0b 0d 27
4b0 47 5f d9 99 e6 a3 35 93 d4 8e ba 71 c3 a8 be 55
4c0 f9 5c c9 60 84 eb 26 b4 f0 43 89 98 89 2c 22 02
4d0 39
4d1
```

Info:

```
000 41 75 74 6f 63 72 79 70 74 5f 76 32 5f 72 61 74
010 63 68 65 74 c6 2a 06 69 06 44 79 1b 00 00 00 20
020 71 12 55 45 47 6a 19 33 4c 9f 0c 13 79 06 6f fa
030 8f 17 89 a0 dd 55 34 e9 5e 78 cf 34 7d 85 47 8c
040 00 0d 2f 00
044
```

IKM:

```
000 a0 d7 c7 5b f6 52 19 1a 27 60 2f f2 8c 55 b1 a6
010 90 e4 93 0b 3c 1c a6 dd 9b 89 dc af a8 e1 18 72
020 9c ac 36 0e 74 e2 5c 9c d6 8d a9 f7 e0 d2 e9 ff
030 97 81 66 f6 7e 09 ca 82 92 15 17 5c de b9 2d 31
040 a4 55 62 46 32 2a cc 50 ac 31 eb b6 66 61 60 c2
050 e8 c7 db 75 76 38 d9 6a 1e 46 bf 11 3f 79 98 2e
060
```

The HKDF output is:

```
000 46 d0 19 07 17 f2 68 a0 d1 52 26 2e e9 28 48 23
010 f9 db dc a5 99 54 00 b3 77 6c a8 8a c5 e5 6b 01
020 2e b4 11 60 a4 9e f9 f9 d5 e8 6f 41 6b 09 bf d3
030 a1 34 56 52 93 02 d4 c5 d1 f8 28 9b 5e 5a f6 f2
040 04 1f b5 2f c6 4d f7 01 bc 75 3d d4 5d d0 5a 19
050 98 fc 39 9a 13 26 18 c4 d6 f2 79 92 fb be 3f 51
060 89 80 49 b1 cb 63 fc 2a 68 47 5c a3 57 58 96 e8
070 ae a8 36 c1 10 6a 3f 23 ec 4d 1c 06 46 58 6e e8
080 b3 24 97 b3 13 a4 77 fa 1d 23 80 b0 6f 96 d4 ac
090 f3 84 f8 d0 46 ab 27 1a 7f b3 10 15 29 9d bc 22
0a0
```

The first 64 octets of this is fed through SHA2_512 to create the salt for the subkey binding signature.

The remaining octets are used to initialize the new subkey's secret key material.

This results in a new TSK and a new cert.

C.2.1. Certificate With New Subkey

This updated outbound certificate should be distributed to the keyholder's peers starting on 2025-11-06T17:33:45Z.

-----BEGIN PGP PUBLIC KEY BLOCK-----

xioGaQZEeRsAAAAGcRJVRUdqGTNMnwwTeQZv+o8XiaDdVTTPXnjPNH2FR4zCkgYf
GwgAAAAzBYJpBkR5ApsDAh4IIiEGH0FrGTQndqxcVMDb0oQzP8aKmI9bzomuqD7i
+bPY0t4DJwkCAAAAAGXjENl3uzg/tNOjW2+khGm+tpsbjJOiUZ0J9I7TZsmLWtcH
lk6y4YJfsZTJl6A4Xdf7n+t3bz8im8Z5aPRfp1SL3OrBK+2DY3GX7xBalEAg1jON
zsQKBmkGRHkjAAAEWmNqYCYyPLlqXKM8d+oKEe+almaw5IdqzQRl3fDQA8Wl0lA
/Ywqa+MA9Ni7ZMBE9pBRUKefJGZOu2NuNIJSzsB4cltYPDqrMSR29JqqnOxXpISJ
VOMIxhClj5VAKqnO4AuJ8zEa3imhDxUr4JORP8eRi5gwkgu8BMQJfDA9kBgeYzQz
x5JMJ5Mh/dW8/WOLdzIE77I8L6dX2NB0v6uAV0A6eFy+LCuivxV/aDRH/9SWltDG
WocNMYqQ75JkFSpe4RVcYvVBEliO6XBInjGAm/hSzwE6RglwbpQdy3WajfYsVuqL
QgRqpZhuKmul8vaij0JGHRhAEDNJuvmbtdgnCneezGtbOPF4zcax5Da+EekANUKh
aClxZjSAFpAcpii0GELEQIPYxaWas6C2fsNxmsh57d/U0p0SAAiOrmVLrdCC6kp
CnuqRiQ8jAC0+/tmZRe3YVi6lBiNhlVXtcoUCOCH4jWiMAMB04sTb1uz+3EtAJkt
ECDJ5muyIHhCBBQMKBKl9rMCujJfIQApZlIHq8q0l2G1NfcdGfsUQ9sbnKZRYik6
5FBMIOSco+Q7XppgxyehKSmEpxJfwCtfc6YIosud5GSDOZMzCVQViDUawRk58KII
S8YDRNq9SrdqY9uLx4gq23Jyvfmkald7EsbAaOKbPQNqPaRpJTO7W9QGmFdCL4Kk
04hOoVZHLtq5Pko+/zyQick7kro5pvPDUAhVxuEu/GilZWNOR9f04TIIIZwmIjwR
5YN0MVC7CyFEaBySPHhYZDdJJ7dj+EKncGts02c07HmM8WUDPAAELlViZ/k10KjH
5oO+jWM6YBZrkLOPVngJSMB43EpJICEySsGc2qCNK5NMtLvNSGsvsuUdr/WDtANT
rWI2XOW9oVs9ApuiKrHT+OGQTKnK+lKhUlJMEV6Iqt/6jt2ereLCzglS8BlfceG
6disSDiz6jKn/lAPsSInlcsQqYR8MwoaE5glTKx8SGFL3xK3VvMrSMED/8g4bsGa
yfeOM7lKrOIrV8nCyXrINODrW5Sgmmml02GR63l0qdybFsQZ0BFmlWWlTRFKekRo
meMB2lyZXbc/qVhqILS4loYtUhSxFjBBbClKcAlHIzxNJeJaofALanq2VbKEbnBT
pCm+0HhH8Xu+/ActZew8QliFrcoRSPM2zcy/nqPAN4QsvRFkykoDiDcId3cjwFAA
EKWKIshR5jNZoSws+fcKcGklhoYBhVY0WhQiZTSwNZsOOqaujNNhLgKZ5bebExqz
AkIrEhozFSS0ZBtzjCs1IMRchHOHzgvp3HWl0ZuwtXKu9MY1h+d0jbqx Fai7IwOs
3xC0PMN03GIWkcNL8UACrLNEFPW6sVwE1ROXcTy5Lde63WsmxipfJEtDwfEgrvxZ
97yDKfRMGILApby2CFuApkYLiQd+lFGwyTNxYbqI7pJKOd1VZ5s/7kBhlCwSedZ9
luK/3JijNotlNrt/7atasask2KkjFpRGpUvArJRJGdVJsZm5sBZ0PTWn94Zxunaz
ICo15jhCENG++qdeJ6ZbGHwMX807LWNXHUC1YtRVTUR0mSFE2mwxLPFeerJlIOkY
FzqWfDCMwXTwllBLbyLQWhk/JW9MNw+ewkrxPD7CiwYYGwgAAAAsBYJpBkR5ApsM
IiEGH0FrGTQndqxcVMDb0oQzP8aKmI9bzomuqD7i+bPY0t4AAAAAR6cQz5l1AOOR
vovCj4Bu3ec92BkTY3LUETvAr6HFyS3+Llv9HvuzcEJ/i+cw9MKuoly0JXzHaZfk
waKnA/SDe0nzYpe3CE+jGj3A4z2uDr4jIAD0xAoGaQzb+SMAAATAwFZa/VzFRtwk
gB6tZdnlrL2KrzH6EfSzXaOtMvCVijsARbDAwU6efCWxW2hBZjfq6kFY58LuEss0
7DcSkacXqJMr4nd/EwsGyDEUTbi76IwauTZ5BzPQEpAYS1ArwSTldg15TK5uW40m
fAczlMhTuZxONz0T5ze2jC1R4ivMtS+8s2QvF3YzN0TX0mb0wG2EhQHItsejJcpm
MDGn0heiQ8w+SnyR6mKNIA9HVW4ZgXnQ8HDecxrBaVZFndKjWlZJNoV2eHRGZY8y
ERbaB3/WXGAdTHM3o2i6W0FqGKwkhLJmyWwlg0zqOnSghCJ7Ri95+TSzd4kAvCTc
pIWr2DV8YWgpt3yS2JvZlCSzmLFnDMPzowVknZEChZIKrCI7Sm/d+D0EeCHPA0A
UZwDdWmdIkAa2qlcFwVVMhgqxQ7NYjVWwR5dygW5eltQ+DpRp4arFmguV2Hwhi67
EEYeyMndOFgyEgNfhybN0YmzObkQmSxqMA4RkitXRhEJqoM6x6D0K0jLBh6ZqL5D
Nhpq+QLf8Lb+BKMp5pv3kSCa8rUSgyuyOS34MDd98C5Wpymgc7g4FAzRdcPEdAp3
0YOOIMKc3HFTswDPpB5XmGfyKBCT+17l2J1vU4W6gAD+hGrA5kWPUpQaMlxhtxDz
ByCfUD6gVkw+QIiIWlUyPBQR28lu2RsRYSQhSwl5trpb4Gly+1cfB8uMJSqe0L6u
2Y0QqiwbLEGSakLBmYwuJlDj63NyzI+4600R0IJYxMdSB6wGlgGgs5oVxq4GpApU
EXNww4M+whff2ZI23FaAtIzW6m6vxpe/CH1EuCmlJZwGZGgwGKNHQLKMVI+ucqx6

```
IzxWm12a53oH6hufwzTlJTHIosqoASivhwNF9R5nQMwiQ5aeS488PLXVGxoGK5T7
6xaqxBxCfMG+RL7IeaLVV3JJloLCRjM0s2sMGY9occehWjp6WGDkB0z/k7LdjDuE
GGAs9Eo4dsdDExYCYXTFDZyIEkCsoriAnuL82zosxUj46tHa3ZFkXqfnC5r3MwQ
wMUDcA/T8ca7+5609mSHIo+Yzj6aVEj3uilnCmlw4JEeFlay3GtoN6SbBian5ka3
ajKsU5nStpYkKhYXmzcPkr3o1LUHNAXBkGKtcsXkI0R8axDlQ5LJw03GoZTpgVe9
eZwIHDQycI+QkB0x4X/FQCK0lm2qajTmmlNhPC4q7KftGKQ6gwnNxsGsdokH97IP
5Tzu+QrCCheGCMbwA2jkecelCcN6Y3gA4F+8228Wan7jwqr1U4f7BkYOh2jnWEnm
OYCXIKuBRkTouXJvOj12hFTNuJ/os4qqklqxq85MF56LM4W0oIEy1LExNQ7dPKje
8hEmSSWNxK0rGZ1c+yLehac2MaPH58ZIAGZuhz2511HWSksNASau0jB6NGaXyliF
YCKewE6SelM4KDNeCp26hlgc52C44Gakaz72u8kgxKgbcQpt2ki0KLmcJSj4xomP
6oXfSBtWBjHNYHCpRltuWTB+PFybxSgKJjHV+Z7vtDcwC7jpAUXN4R+9gAzRFLW0
0DDyYcFWvFTLYWl491xBBCBMx6rFlBgu/8DUpUEmRdJkBza11+17J1xCX9V0+1uO
qL5cp9D9n8KRBhgbCAAAADIFgmK2/kFCQANLwACmwQiIQYfQWuBNCD2rFxUwMHS
hDM/xoqYj1vOia6oPuL5s9jS3gAAAAB03hBpDlwq4VDDSL1+arNyH9TZC46aKabM
bswuKZDNMBhryioEX2mxYZTJofZa6qUHV2lfgxVyKmw239dMk9M7mleFUuKlls7T
aWzDxicPfxBFCw==
-----END PGP PUBLIC KEY BLOCK-----
```

C.2.2. TSK With New Subkey

Every device from the keyholder should be able to derive this exact secret key from the TSK found in Appendix C.1.2.

```
-----BEGIN PGP PRIVATE KEY BLOCK-----
```

```
xUsGaQZEeRsAAAAGcRJVRUDqGTNMnwwTeQZv+o8XiaDdVTTPXnjPNH2FR4wAET8I
Q3fQeNm71dOZSkB/260JmbL1QIX/q4ugYzBOFKLCkgYfGwgAAAAzBYJpBkR5ApsD
Ah4IIiEGH0FrgTQndqxcVMDB0oQzP8aKmI9bzomuqD7i+bPY0t4DJwkCAAAAAGXj
ENl3uzg/tNojW2+khGm+tpsbjJ0iUZOJ9I7TZsMLwtCh1k6y4YJfsZTJl6A4Xdf7
n+t3bz8im8Z5aPrfplSL3OrBK+2DY3GX7xBa1EAgljOnx8RrBmkGRHkjAAAEwMnQ
yYCYyPLlqXKM8d+oKEe+almaw5IdqzQRl3fDQA8Wl0lA/Ywqa+MA9Ni7ZMBE9pBR
UkefJGZOu2NuNIJSzsB4cltYPDqrMSR29JqqnOxXpISJVOmIhxClj5VAKqnO4AuJ
8zEa3imhDxUr4JORP8eRi5gwkgu8BMQJfDA9kBgEYzQzx5JMj5Mh/dW8/WOLdzIE
77I8L6dX2NB0v6uAV0A6eFy+LCuivxV/ADRH/9SW1tdGWocNMYqQ75JkFSpE4RVc
YvVBE1iO6XBInjGAm/hSzwE6RglwbpQdy3WajfYsVuqLQgRppqZhuKmul8vaij0JG
HRhAEDNJuvmbtdgnCneezGtbOPF4zcax5Da+EekANUKhaClxZjSAFpAcpii0GE1E
QIPYxaWas6C2fsNxmsh57d/U0p0SAaiOrmVLrdCC6kpCnuqRiQ8jAC0+/tmZRe3
YVi6lBiNHlVXtcoUCOCH4jWiMAMB04stBluz+3EtAJkteCDJ5muyIHhCBBQMKBKl
9rMCujJfIQApZlIHq8q0l2G1NfcdGfsUQ9sbnKZRyik65FBMIosco+Q7Xppgxyeh
KSmEpxJfwCtfc6YIosud5GSD0ZMzCVQViDUawRk58KIIS8YDRNq9SrdqY9uLx4gq
23Jyvfmkald7EsbAaOKbPQNgPaRpJT07W9QGmfDCL4Kk04h0oVZHLtq5Pko+/zYQ
ick7kro5pvPDUaVhxuEu/GilZWNOR9f04TIIItZwmIjwR5YN0MVC7CyFEaBySPHhY
ZDdJJ7dj+EKncGts02c07HmM8WUDPAAELlViZ/k10KjH5oO+jWM6YBZrkLOPVngJ
SMB43EpJICeYsGc2qCNK5NMtLvNSGsvsuUdr/WDtANTrWI2Xow9oVs9AppuiKrH
T+OGQTKnK+lKhUlJMEV6Iqt/6jt2ereLCzglS8B1fceG6disSDiz6jKn/1APsSiN
lcsQqyR8MwoaE5glT�X8SGFL3xK3VvMrSMED/8g4bsGayfEOM7lKr0Irv8nCyXrI
NOdrW5Sgmmml02GR63l0qdybFsQZ0BFmlWWlTRFKekRomeMB2lyZXbc/qVhqILS4
loYtUhSxJfBBbClKcAlHIzxNJeJaofALanq2VbKEbnBTpCm+0HhH8Xu+/ACtZew8
```


QliFrcoRsPM2zcy/nqPAN4QsvRFkykoDiDcId3cjwfAAEKWKIshr5jNZoSWS+fcK
cGklhoYBhVY0WhQiZTSwNZs0OgaujNNhLgKZ5bebExqzAkIrEhozfSS0ZBtzjCs1
IMRchHOHzgvp3HWL0ZuwtXKu9MY1h+d0jbqx Fai7IwOs3xC0PMN03GIWkcNL8UAC
rLNEFPW6sVwE1ROXcTy5Lde63WsmxipfJEtDwFEqrVxZ97yDKfRMGILApbY2CFuA
PkYLiQd+lFGwytNxYbqi7pJKOd1VZ5s/7kBhlCwSedZ91uK/3JijNotlNrt/7ata
saSk2kkjFpRGpUvArJRJGdVJszm5sBZ0PTWn94ZxunazICo15jhCENG++qdEJ6Zb
GHwMX807LWNXHUC1YtRVTUR0mSFE2mwXLPFeerJlIOkYFzqWfDCMwXTw11BLbyLQ
Whk/JW9MNw+ewkrxPD4AaGZh9fcInOX2YpJXNKEnHCNh08uWEosBWRDzpeOVGja
b3dBAAafhaFGEhx4WqVYOan5ZDw/WNqlTVIRDLWvDuQjHvRMLCoqgQ/E+04G6uS6
arNh3q8BziJa4mELayyhwsGGBsIAAAALAWCaQZEeQKbDCIhBh9Ba4E0J3asXFTA
wdKEMz/GipiPW86Jrqq+4vmz2NLeAAAAAEenEM+ZdQDjkb6Lwo+Abt3gvdgZE2Ny
1BE7wK+hXckt/i5b/R77s3BCf4vnMPTCrqJctCV8x2mX5MGipwP0g3tJ82KXtwhP
oxo9wOM9rg6+IyAAx8RrBmkGRHkjAAAEWJgkZiQ1uQmDXbnBCbOdDnsF0yA6XftQ
UQ/PN8euSppqW2PFQAV3ZXOLpyBO8pzDV0yblpFfi1BNx1EXp+JPVFJc0ZBd7sCo
eOlQp/eEoZs067OcFOOEF3O2srNGKwt1xVZ46DBKpgvDeTai0IqSB6ZiY4BI1Rof
PxNdIORQk3NrMfFwI3c8aWy0XJSmoKQFJWmz42CDqMKDiGU7PQsdOcIMuPdaRHBV
LlgFm3S28VYfbatTHVdhSXYlPDobyINix3StigCu4fIAUMGkiMtJ/pocv9Ryd4HH
/BcUynzEDwIdmJtFuaFanNR02AJRLnIClWWZeeU19Ww53fUA6+dqzRHH+RRSMkm8
WWw3P7pphEfE+qCF15my7VESnYp3eFkrVleFxAhL6AdoOuYoCiPr5fGx/M9Fd18
50nCHyjoAjrJtXuSXL0BmnsPQcWjSrqlenS0/vJ+yKE6jix0M40UsXHAjVrC0rcB
20NiNjl2D0ApXkTMGziA61CLBakM13eJ2TektMt4BTmIoEHJppY8eQNuHmmpuKSd
COAD1ZWJ0cqQsilqzMoJmKyKgt3GKFKcnDDCsF0dmEvakGsDyOelrKOuyrEAI2L
lgd/e7dLEcxNxRenkcJZCGZct9qyf0jAyeBV6EFpBNmExia4dwqPcTnFliWI8zw3
QDCei5LBG9y12HBj4Rk9ZiWhmOsFixtPN6CaSHgxTFwTngkXzuaLqkUfNfesEckq
HeZESioCoqsuDYiI8yEArki3JwlAcKl+OxS0AZG7xtDPnLCuJdcpBVYwnneAgzMn
+QHIApJx/el23bpz11l88mrNm6Iiw8fB6EA3aZF3qbnFjUkPrqC1YmS8lvWkWvtp
jAh66MkZyVMsLtclDaSafseyluBcdGA/JEOsAtVgRLhB/AJtfZplJJO5YwbHSxua
OUMJMYgEgUVruNiB+oXBdauU8WasAZg8Y+QCqlMQ6MgS5dxBHZZ8//jM8HAD8WUF
P/lXhHW0x3y1CNQl7SQQVUjIyfn6aEst7Xx8dAJ3ONCHE/iEafASWbRdkAoGQvEN
yNq7bBO5yLeKWUWfDEWxr8DMc5x9mfez9XapsnHGCWlxlpiIcDW+nXCJz7V6/NV
kXCACEoIIFoBcJC3UoaOvKZoOthfNIqBCsRsAQSHGkKvqSQw6KlQtgFPHAE5B3WY
c6yPGkV331l+MbCdP+Gv9TSNBdJsb2UXztUI4NKH2KZD0jpp8ow5WCgpmUSr85WJ
L9Cy9FekYQsNrmig4YN7Imc4HwVKHBsGfJGk2Oyv+tKguMeaKiJZ3LQjefHITDh1
02Bjc2lAr4G1ElFrByBre8i3/gIpKZk6vSI5Moy5LmpNG1QkKRMb8gUokVaQNgH
Zhncl0sEfHQBYrrEHY2AixfF/OZeA5BGKcLGIRVCLPjzAdlKDLbtWSYwWdtK65kzE
mfdhBrGyr8uKyaLOBSN7e2fIEZA/Tdi8pkd8U9sYL/afkTcYJUjNlpcTMxA/zKg6
ttGqQneaQUE3RKu3qdwUxrQLTuWtmPLMa5nKg+E7DzewpjFXULF56PVNUSoyDiP
l8J+zFwLDSdHX9mZ5qMlk9SOunHDqL5V+VzJYITrJrTwQ4mYiSwiAjkAoNfHW/ZS
GRonYC/yjFWxppDkKws8HKbmd4ncr6jhGHKcrDYOdOJcnNaNqffg0un/l4Fm9n4J
yoKSFRdc3rktMarVYkYyKsxQRdHrtmZhYMLox9tldjjZah5GvxE/eZguwpeGGBsI
AAAAMgWCaQZEeQUJAA0vAAKbBCIhBh9Ba4E0J3asXFTAwdKEMz/GipiPW86Jrqqg
+4vmz2NLeAAAAAH8pEOM21AerK8gkLTZCfV+wQVbxRLOqlx0LgGZKg3CXqIHnMrP/
zlnA6EWH6YkSehlaCBL1kz1jWOGKMULgB6LzVFw7+Pw8IwIrcfvdMNYwfEEJx8Rr
BmkM2/kjAAAEWMBWwv1cxUbcJIAerc3Z5ay9iq8x+hH0s12jrTLwlyo7AEWwwMFO
nnwll1toQWY30OpBWOfC7hLLDuw3EpGnF6iTK+J3fxMLBsgxFLW4u+iMGrk2eQcz
0BKQGETQK8Ek9XYNEuYubluNjnwHM5TIU7mctjc9E+cxNowtUeIrrzLUvvLNkLxd2
MzdE19Jm9MBthIUBylbHoyXKZjAxp9IXokPMPkp8kepijSGvRlcOGYF50PBw3nMa
wWlWRZw5i1pcyTaFdnh0RmWPMhEW2gd/1lxmg0xzN6NoultBahilpIZSZslsNYNM

```
6jp0oIQie0Yvefk0s3eJALwk3KSFq9glfGFoKbd8ktib2dQks5ixZwzD86MFZDWR
AoWSJKwi00pv3fg9BHghzwG9AFGcA3VpnSJAGtqtXBcFVZoYKsUOzWl1VsEeXcoF
uXtbUPg6UaeGqxZoLldh8IYuuxBGHsjJ3ThRmBIDX4cmzdGJszm5EJksajAOEZIr
V0YRCaqD0seg9CtIywYemai+QzYaavkC3/C2/gSjKeab95EgmvK1Ehsrsjkt+DA3
ffAuVqcpoH04OBQM0XXDXHQKd9GDjiDCnNxxU7MAz6QeV5hn8igXE/pe5didb1OF
uoAA/oRqwoZFqVKUGjJcYbcQ8wcgn1A+oFZMPkCIiFtVMjwUK9vJbtkeEWEkIUJ
eba6W+BpcvtXHwFLjCUqntC+rtmNEKosG5RBkgCiwZmFridQ4+tzcsyPuOtNedCC
WMTHUGesBtYBoLoaFcauBqQKVBFzcMODPsIX39mSNtxWgLSM1upur8aRPwh9RLgj
JSWcBmRoMBijR0CyjFSPRnKseiM8VjJdmud6B+obn8M05SUxyKLKqAEiL4cDRfUe
Z0DMiKOWnkuPPDyl1RsaBiuU++sWqsQcQnzBvkS+yHm1lVdySZaCwkYzNLNrDBmP
aHHHoVoz+lhnsGdM/5Oy3Yw7hBhgLPRKOHbHQxMWAmGF0xQ82IhJArKK4gJ7i/Ns
6LMVI+OrR2t2RZF6n5wua9zMEMDFA3AP0/HGu/uetPZkhyKPMGY+mlRI97opZwpp
cOCRHhZWstxraDekmwYmp+ZGt2oyrFOZ0raWJJ1WF5s3D5K96NS1BzQFwZBirXLF
5CNEfGsQ5UOSycNNxqGU6YFXvXmcCBw0MnCPkJAdMeF/xUAitJZtqmo05ppTYTwu
KuyN7RikOoMJzcbBrHaJB/eyD+U87vkKwgoXhgjG8ANo5HnHpQnDemN4AOBfvNtv
Fmp+48Kq9VOH+wZGDodo5lhJ5jmAlYJLgUZE6Llybzo9doRUzbo/6LOKqpNasavO
TBeeizOfTKCBMpSxMTUO3Ty03vIRJkkljCStKxmdXPsi3oWnNjGjx+fgSABmboc9
uZdRl1kplDQLGrtIwejRml8pYhWApHsB0knPTOCgzXgqduoZYH0dguOBmpGs+9rvJ
IMSoG3EKbpdItCi5nCUo+MaJj+qF30gbVgSRzWBwqUZbblkwfjxcM8UoCiYx1fme
77Q3MAu46QFFzeEfvyAM0RS1tNAw8mHBVrxUy2FpePZcQQgTMeqxZQYLv/AlKVB
JkXSZAc2tdftedydcQ1/VdPtbjqi+XKfQ/Z8AAB+1L8ZN9wG8dT3UXdBaGZj80ZoT
JhjElvJ5kvu+PlGJgEmxy2P8KmhHXXKNXWJborqg2wRBqPyPsTRwGRlhu6LMkl7MT
pHf6HSOAsG+WlKzzhPjQRqsnGn+zEBUpnbwiwpEGGBsIAAAAMgWCAQzb+QUJAA0v
AAKbBCIhBh9Ba4E0J3asXFTAWdKEMz/GipiW86Jrqq+4vmz2NLeAAAAAHTeEGkO
XCrhUMNivX5qs3If1nKljpoppsxuzC4pkM0wGGvKKGRfabFhlMmh9lrqpQdXaV+D
FXIqbDbf10yT0zuaV4VS4qWWztNpbMPGJw9/EEUL
-----END PGP PRIVATE KEY BLOCK-----
```

C.2.3. Merged Certificate

A peer that had received the original certificate and then the subsequent certificate would have merged the two into this object:

-----BEGIN PGP PUBLIC KEY BLOCK-----

```
xioGaQZEeRsAAAAGcRJVRUdqGTNMnwwTeQZv+o8XiaDdVTTPXnjPNH2FR4zCkgYf
GwgAAAAzBYJpBkR5ApsDAh4IIiEGH0FrgTQndqxcVMDB0oQzP8aKmI9bzomuqD7i
+bPY0t4DJwkCAAAAAGXjENl3uzg/tNojW2+khGm+tpsbjJOiUZ0J9I7TZsmLWtcH
lk6y4YJfsZTJl6A4Xdf7n+t3bz8im8Z5aPRfp1SL3OrBK+2DY3GX7xBa1EAgljoN
zsQKBmkGRHkjAAAEwMnQyYCYyPLlqXKM8d+oKEe+almaw5IdqzQRl3fdQA8Wl0lA
/Ywqa+MA9Ni7ZMBE9pBRUKefJGZOu2NuNIJSzsB4cltYPDqrMSR29Jqqn0xXpISJ
VomIxc1j5VAKqn04AuJ8zEa3imhDxUr4JORP8eri5gwkgu8BMQJfDA9kBgeYzQz
x5JMj5Mh/dW8/WOLDzIE77I8L6dX2NB0v6uAV0A6eFy+LCuivxV/ADRH/9SW1tDG
WocNMYqQ75JkFSpe4RVcYvVBeli06XBInjGAm/hSzwE6RglwbpQdy3WajfYsVuqL
QgRqpZhuKmul8vaij0JGHRhAEDNJuvmbtdgnCneezGtbOPF4zcax5Da+EekANUKh
aClxZjSAFpAcpii0GEleQIPYxaWas6C2fsNxmsh57d/U0p0SAAiOrmVLrdCC6kp
CnuqRiQ8jAC0+/tmZRe3YVi6lBiNhlVXtcoUCOCH4jWiMAMB04sTb1uz+3EtAJkt
ECDJ5muyIHhCBBQMKBKI9rMCujJfIQApZlIHq8qOl2G1NfcdGfsUQ9sbnKZRYik6
```

5FBMIOsco+Q7XppgxyehKSmEpxJfwCtfC6YIosud5GSDOZMzCVQViDUawRk58KII
S8YDRNq9SrdqY9uLx4gq23Jyvfmkald7EsbAaOKbPQNgPaRpJTO7W9QGmFdCL4Kk
O4hOoVZHltq5Pko+/zYQick7kro5pvPDUaVhxuEu/GilZWNOR9fO4TIItZwmIjwR
5YN0MVC7CyFEaBySPHhYZDdJJ7dj+EKncGts02cO7HmM8WUDPAAEllViZ/k10KjH
5oO+jWM6YBZrkLOPVngJSMB43EpJICEySsGc2qCNK5NMtLvNSGsvsuUdr/WDtANT
rWI2XOW9oVs9APpuiKrHT+OGQTKnK+1KhUlJMEV6Iqt/6jt2ereLCzgls8B1fceG
6disSDiz6jKn/1APsSinlcsQqyR8MwoaE5glTcx8SGFL3xK3VvMrSMED/8g4bsGa
yfeOM7lKrOIrv8nCyXrINODrW5Sgmmml012GR6310qdybFsqZ0BFmlWWlTRFKekRo
memB21yZXbc/qVhQILS41oYtUhSxFjBBbClKcAlHIzxNJeJaofALanq2VbKEbnBT
pCm+0HhH8Xu+/ACtZew8QliFrcoRSPM2zcy/nqPAN4QsvRFkykoDiDcId3cjwfAA
EKWKIshR5jNZoSsW+fcKcGklhoYbHVY0WhQiZTSwNZs0OqaujNNhLgKZ5bebExqz
AkIrEhozFSS0ZBtzjCs1IMRchHOHzgvp3HWl0ZuwtxKu9MY1h+dOjbqx Fai7IwOs
3xC0PMN03GIWkcNL8UACrLNEFPW6sVwElROXcTy5Lde63WsmxipfJEtDwFeqrvxZ
97yDKfRMGILApby2CFuAPkYLiQd+lFGwytNxYbqI7pJKOdLVZ5s/7kBhlCwSedZ9
luK/3JijNotlNrt/7atasasK2kKjFpRGpUvArJRJGdVJszm5sBZ0PTWn94Zxunaz
ICo15jhCENG++qdeJ6ZbGHwMX807LWNXHUC1YtRVTUR0mSFE2mwxLPFeerJlIOkY
FzqWfDCMwXTwllBLbyLQWhk/JW9MNw+ewkrxPD7CiwYYGwgAAAAAsBYJpBkR5ApsM
IIEGH0FrgTQndqxcVMDB0oQzP8aKmI9bzomuqD7i+bPY0t4AAAAAR6cQz511A0OR
vovCj4Bu3eC92BkTY3LUETvAr6HFyS3+Llv9HvuzceJ/i+cw9MKuoly0JXzHaZfk
waKna/SDe0nzYpe3CE+jgj3A4z2uDr4jiAD0xAoGaQZEESMAAATAmCRkirW5CYNd
ucEJs50OewXTIDpd+1BRD883x65KmmPbY8VABXdlc4unIE7ynMNXTJvWkV8jUE3H
URen4k9UULzRkF3uwKh46VCn94Shmw7rs5wU44QXc7ays0Ypa3XFVnjoMEo+C8N5
NqLQipIHpmJjgejVgh8/E10g5FCTc2sx8XAjdzxpBLRclKagpAUlabPjYIOowoOI
ZTs9Cx05wgy491pEcFuvWAWbdLbxVh9tq1MdV2FJdiU8OhvIg2LHdK2KAK7h8gBQ
waSiY0n+mhy/1HJ3gcf8FXTkfMQPAh2Ym0W5oVqc1HTYAlEucgKVZZ155TX1bDnd
9QDr52rNEcf5FFIySbxZbDc/ummER8T6oIXXmbLtURKdind4WStWV4XEBGEvoB2g
65igKI+vl8bH8z0V2XznScIfKM5qNEMle5JcugGaew9BxaNKuot6dLT+8n7IoTqQ
LHQzg5SxcccNwSLstWbQ2I2OXYPQCleRMwbOIdrUISFqQzXd4nZN6S0y3gFOYig
Qcmmljx5A24eaam4pJ0I4APVlYnRypCyKwRMyiOYrKQZy3cYoUpycMMKwXR2YS9q
QawPI57Wso67KsQAjYvWB397t0sRze3FF6erWlkIZly32rJ86MDJ4FXoQWke2YTg
Jrh3Co8K2cWWJYjzPDDAMJ6LkseB3LXYcGPhGT1mJaGY6wWLG083oJpIeDFMXBoe
CRfO5ouqRR8196wRySod5kRKKgKiQy4PIiLzIQCuSLcnCUBwqX47FLQBkbvG0M+c
sK6NlykFVjCed4CDMyf5Acho8nH96XbdunPWWXzyas2boiLDx8HoQDdpkXepucWN
SQ+uoLVizLyW9Ypa+2mMCHroyRnJUYwulyUNpIB+x7KW4Fx0YD8kShJq1WBEUEH8
Am19mmUkk7ljBsdLG5o5QwkxiASBRWu42IH6hcFlq5TxZqwBmDxj5AKqUxDoyBL1
3EEdlnz/+MzwcAPxZQU/+VeEdbTHfLUIlCXtJCpVSMjJ83poSxPtfHx0Anc4lwcT
+IRp8BJZtF2QCgZC8Q3I2rtse7nIt4pZRYV0RbGvwMxznH2Z97P1cMyccYJaXGW
kghwNb6dcInPtXr81WRxcIBx44ggWgFwkLdSho68pmg60cWcioEKxGwBBicaQq+p
JDDoqVC2AU8cATkHdZhzrI8aRXffWX4xsJ0/4a/1NI0F0mxvZrF01Qjg0ofYpkPS
OmnyjDlYKCMzRKvzlykv0LL0V6RhCw2uaKDhg3siZzgfBUocGwZ8kaTY7K/60qC4
x5oqIlncCN58chMOHU7YGNzaUCvgbUTUWShIGt7yLf+AikpmTq9IjkyjLkuak0b
VCQpEwHyBSiRVqo2CFmGdzU6wR8dAFiusQfLYCLF8X8514DkEYpwsYitUIS+PMB2
UoOVulZJhbB20rrmTMSZ92EGuBivy4rJos5tI3t7Z8gRkD9N2LymR3xT2xgv9p+R
NxglSM2WlxMzED/MqDq20apCd5pBQTdEq7ep3BTGTAtO5a2Y8sxrmcqd4TsPn7Cm
MVdSUXno9U1RKjKgOI+Xwn7MXAsNJ0df2Znm0zWT1I66ccOovlX5XmLghOsmTBD
iziJLJCICOCrBhgbCAAAADIFgmKGRHkFCQANLwACmwQiIQYfQWuBNcd2rFxUwMHS
hDM/xoqYjlvOia6oPuL5s9js3gAAAAB/KRDjNtQHkZBvJC02Qn1fsEFW8USzkJcd
C4BmSoNw16iB5zKz/85ZwOhFh+mJEnoZWggS9ZM9Y1jhiJFNyAei81RcO/j8PCMC

```

K3H73ZjWMHxBcC7ECgZpDNv5IwAABMDAVlr9XMVG3CSAHq3N2eWsvYqvMfor9LNd
o60y8JWKOWBFsMDBTp58JZdbaEFmN9DqQVjnwu4Syw7sNxKRpxeokyvid38TCwbI
MRS1uLvojBq5NnkHM9ASkBhLUCvBJPV2DXlMrm5bjSZ8BzOUyFO5nE43PRPnMTaM
LVHiK8y1L7yzZC8XdjM3RNfSZvTABYSFAci2x6MlymYwMafSF6JDzD5KfJHqYo0h
r0dXDhmBedDwcN5zGsFpVkWcOSNaXmK2hXZ4dEZl jzIRFtoHf9ZcZoNMczejaLpb
QWoYpaSGUmbJbDWDTOo6dKCEIntGL3n5NLN3iQC8JNykHavYNXxhaCm3fJLYm9nU
JLOYsWcMw/OjBWQ1kQKFkiSsIjtkb934PQR4Ic8BvQBRnANlaZ0iQBrarVwXBVWa
GCrFDs1iNVbBH13KBbl7WlD4OlGnhqsWaC5XYfCGLrsQRh7Iyd04UZgSA1+HJs3R
ibM5uRCZLGowDhGSKldGEQmqgZrHoPQRSMsGHpmovkM2Gmr5At/wtv4Eoynmm/eR
IJrytRiBk7I5LfgwN33wLlanKaBzuDgUDNF1w8R0Cnfrg44gwpzccVOzAM+kHleY
Z/IoFxp6XuXYnW9ThbqAAP6EasDmRalS1BoyXGG3EPMHIJ9QPqBWTd5AiIhbVTI8
FCvbyW7ZGxFhJCFCLXm2ulvgaXL7Vx8Hy4wlKp7Qvq7ZjRCqLBUQZIAosGZha4n
UOPrc3LMj7jrTRHQgljEx1IHrAbWAaCzmhXGrgakClQRc3DDgz7CF9/ZkjbcVoC0
jNbqbq/GkT8IfUS4IyUlnAZkaDAYo0dAsoxUj65yrHojPFYyXZrnegfqG5/DNOU1
MciiqgBii+HA0XlHmdAzCJDlp5Ljzw8tdUbGgYr1PvrFqrEHEJ8wb5Evsh5otVX
ckmWgsJGMzSzawwZj2hxx6FaM/pYZ0oHTP+Tst2MO4QYYCz0Sjh2x0MTFgJhhdMU
PNIISQKyiuICE4vzb0izFSPjq0drdkWRep+cLmvczBDAXQNwD9Pxxrv7nrT2Zici
j5hmPppUSPe6KWcKaXDgkR4WVrLca2g3pJsGJqfmRrdqMqxTmdK2liSSFhebNw+S
veJutQc0BcQQYqlyxeQjRHxrEOVDksnDTcahlOmBV715nAgcNDJwj5CQHTHhf8VA
IrsWbapqNOaaU2E8Lirsp+0YpDqDCC3Gwax2iQf3sg/lPO75CsIKF4YIxxvADaOR5
x6UJw3pjeADgX7zbbxZqfuPCqvVTh/sGRg6HaOdYSeY5gJciS4FGR0i5cm86PXaE
VM26P+iziqqTWGrzkwXnoszhbSggTKUstE1Dt08qN7yESZJJY3ErSsZnVz7It6F
pzYxo8fnxkgAZm6HPbmXUdZKS0Cxq7SMHo0ZpfKWIVgKR7ATpJ6UzgoM14KnbgG
WBznYLjgZqRrPva7ySDEqBtxCm3aSLQouZwlKPjGiY/qhd9IG1YEKclgcKlGW25Z
MH48XJvFKAomMdX5nu+0NzALuOkBRc3hH72ADNEUtbtQMPJhwVa8VMthaXj2XEEE
IEzHqsWUGC7/wNSlQSZF0mQHNrXX7XsnXEJf1XT7W46ovlyn0P2fwpeGGBsIAAAA
MgWCaQzb+QUJAA0vAAKbBCIhBh9Ba4E0J3asXFTAWdKEMz/GipiPW86Jrqq+4vmz
2NLeAAAAAHTeEGkOXCrhUMNivX5qs3If1NkLjpoppsxuzC4pkM0wGGvKKgRfabFh
lMmh9lrrpQdXaV+DFXIqbDbf10yT0zuaV4VS4qWWztNpbMPGJw9/EEUL
-----END PGP PUBLIC KEY BLOCK-----

```

After 2025-11-11T17:33:45Z (when the first subkey expires and is no longer usable), the peer can prune the merged certificate, which should bring it back to the certificate found in Appendix C.2.1.

C.3. Old Subkey Removed

When the key and certificate are no longer useful, they can be removed.

C.3.1. TSK With Old Subkey Removed

max_rd time after the old rotating subkey expires (that is, any time after 2025-11-21T17:33:45Z, the keyholder's device checks for all incoming messages and processes them.

Once they have all been received and processed, the keyholder's device destroys the expired rotating secret subkey, leaving this TSK.

-----BEGIN PGP PRIVATE KEY BLOCK-----

xUsGaQZEeRsAAAAGcRJVRUdqGTNMnwwTeQZv+o8XiaDdVTTPXnjPNH2FR4wAET8I
Q3fQeNm7ldOZSkB/260JmbLlQIX/q4ugYzBOFKLCkgYfGwgAAAAzBYJpBkR5ApsD
Ah4IIiEGH0FrGTQndqxcVMDB0oQzP8aKmI9bzomuqD7i+bPY0t4DJwkCAAAAAGXj
ENl3uzg/tNOjW2+khGm+tpsbjJOiUZOJ9I7TZsmLWtcHlk6y4YJfsZTJl6A4Xdf7
n+t3bz8im8Z5aPRfp1SL3OrBK+2DY3GX7xBa1EAg1joNx8RrBmkGRHkjAAAEWmNq
yYCYyPLlqXKM8d+oKEe+almaw5IdqzQRl3fDQA8Wl0lA/Ywqa+MA9Ni7ZMBE9pBR
UkefJGZOU2NuNIJSzsb4cltYPDqrMSR29JqqnOxXpISJVOmIhxClj5VAKqnO4AuJ
8zEa3imhDxUr4JORP8eRi5gwkgu8BMQJfDA9kBgEYzQzx5JMJ5Mh/dW8/WOLdzIE
77I8L6dX2NB0v6uAV0A6eFy+LCuivxV/ADRH/9SWltdGWocNMYqQ75JkFSpE4RVc
YvVBE1iO6XBInjGAm/hSzwE6RglwbpQdy3WajfYsVuqLQgRppqZhuKmul8vaij0JG
HRhAEDNJuvmbtdgncneezGtbOPF4zcax5Da+EekANUKhaClxZjSAFpAcp1i0GE1E
QIPYxaWas6C2fsNxmsh57d/U0p0SAaiOrmVLrdCC6kpCnuqRiQ8jAC0+/tmZRe3
YVi6lBiNhlVXtcoUCOCH4jWiMAMB04stBluz+3EtAJktECDJ5muyIHhCBBQMKBKl
9rMCujJfIQApZlIHq8q0l2G1NfcdGfsUQ9sbnKZRyik65FBMI0sco+Q7Xppgxyeh
KSmEpxJfwCtfc6YIosud5GSD0ZMzCVQViDUawRk58KIIS8YDRNq9SrdqY9uLx4gq
23Jyvmfka1d7EsbAaOKbPQNgPaRpJTO7W9QGmFdCL4Kk04hOoVZHLtq5Pko+/zYQ
ick7kro5pvPDUaVhxuEu/GilZWNOR9f04TIIItZwmIjwR5YN0MVC7CyFEaBySPHhY
ZDdJJ7dj+EKncGts02c07HmM8WUDPAAEllViZ/k10KjH5oO+jWM6YBzrkLOPVngJ
SMB43EpJICeYSsGc2qCNK5NMtLvNSGsvsuUdr/WDtANTrWI2XOW9oVs9AppuiKrH
T+OGQTKnK+lKhULJMEV6Iqt/6jt2ereLCzglS8B1fceG6disSDiz6jKn/1APsSIn
lcsQqyR8MwoaE5glTkx8SGFL3xK3VvMrSMED/8g4bsGayfEOM7lKr0Irv8nCyXrI
NODrW5Sgmmml02GR63l0qdybfsqZ0BFmlWWlTRFKekRomeMB2lyZXbc/qVhqILS4
loYtUhSxJfBBbClKcAlHIzxNJeJaofALanq2VbKEbnBTpCm+0HhH8Xu+/ActZew8
QliFrcoRSPM2zcy/nqPAN4QsvRFkykoDiDcId3cjwfAAEKWKIshR5jNZoSsW+fcK
cGklhoYBhVY0WhQiZTSwNZs0OqaujNNhLgKZ5bebExqzAkIrEhozfsS0ZBtzjCs1
IMRchHOHzgvp3HWl0ZuwtxKu9MY1h+d0jbxqFai7IwOs3xC0PMN03GIWkcNL8UAC
rLNEFPW6sVwElROXcTy5Lde63WsmxipfJEtDwFegrvxZ97yDKfRMGILApbY2CFuA
PkYLiQd+lFGwyTNxYbqi7pJKOd1VZ5s/7kBhlCwSedZ9luK/3JijNotlNrt/7ata
saSk2kKjFpRGpUvArJRJGdVJszm5sBZ0PTWn94ZxunazICo15jhCENG++qdEJ6Zb
GHwMX807LWNXHUC1YtRVTUR0mSFE2mwxLPFeerJlIOkYFzqWfDCMwXTwllBLbyLQ
Whk/JW9MNw+ewkrxPD4AaGZh9fcInOX2yYpJXNKEnHCNh08uWEosBWRDzpeOVGja
b3dBAAafhaFGEhx4WqVYOan5ZDw/WNqlTVIRdLWvDuQjHvRMLCoqgQ/E+04G6uS6
arNh3q8BziJa4mELayyhwsGGBsIAAAALAWCaQZEeQKbDCIhBh9Ba4E0J3asXFTA
wdKEMz/GipiPW86Jrqq+4vmz2NLeAAAAAEenEM+ZdQDjkb6Lwo+Abt3gvdgZE2Ny
1BE7wK+hxckt/i5b/R77s3BCf4vnMPTCrqJctCV8x2mX5MGipwP0g3tJ82KXtwhP
oxo9wOM9rg6+IyAAx8RrBmkM2/kjAAAEWMBWVl1cxUbcJIAerc3Z5ay9iq8x+hH0
sl2jrTLwlYo7AEWwwMFOnnwl1ltoQWY30OpBWOfC7hLLDuw3EpGnF6iTK+J3fxML
BsgxFLW4u+iMGrk2eQcz0BKQGETQK8Ek9XYNeUyubluNjnwHM5TIU7mctjc9E+cx
NowtUeIrrLUvvLNkLxd2Mzde19Jm9MBthIUBylbHoyXKZjAxp9IXokPMPkp8kepi
jSGvRlCOGYF50PBw3nMawWlWRZw5ilpcyTaFdnh0RmWPMhEW2gd/1lxmg0xzN6No
ultBahilpIZSZslsNYNM6jp0oIQie0Yvefk0s3eJALwk3KSFq9glfGFoKbd8ktib
2dQks5ixZwzD86MFZDWRaOwsJKwi00pv3fg9BHghzwG9AFGcA3VpnSJAGtqtXBcF
VZoYKsUOzWl1VsEeXcoFuXtbUPg6UaeGqxZoLldh8IYuuxBGHsjJ3ThRmBIDx4cm
zdGJszm5EJksajAOEZIrV0YRCaqD0seg9CtIywYemai+QzYaavkC3/C2/gSjKeab
95EgmvK1Ehsrsjkt+DA3ffAuVqcpoH04OBQM0XXDxHQKd9GDjIDCnNxxU7MAZ6Qe
V5hn8igXE/pe5didb1OFuoAA/orQwOZFqVKUGjJcYbcQ8wcgn1A+oFZMPkCIiFtV

MjwUK9vJbtkbEWEkIUJeba6W+BpcvtXHwfljCUqntC+rtmNEKosG5RBkgCiwZmF
ridQ4+tzcsyPuOtNEdCCWMTHUgesBtYBoLoaFcauBqQKVBfzcMODPsIX39mSntxW
gLSMlupur8aRPwh9RLgjJSWcBmRoMBijR0CyjFSPrnKseiM8VjJdmud6B+obn8M0
5SUxyKLKqAEiL4cDRfUeZ0DMikOWnkuPPDy1lRsaBiuU++sWqsQcQnzBvkS+yHmi
1VdySZaCwkYzNLNrdBmPaHHHoVoz+lhnsGdM/5Oy3Yw7hBhgLPRKOHbHQxMWAmGF
0xQ82IhJArKK4gJ7i/Ns6LMVI+OrR2t2RZF6n5wua9zMEMDFA3AP0/HGu/uetPZk
hyKPmGY+mlRI97opZwppcOCRhhZWstxraDekmwYmp+ZGt2oyrFOZ0raWJJiWF5s3
D5K96NS1BzQFWzBirXLF5CNEfGsQ5UOSycNNxqGU6YFXvXmcCBw0MnCPkJAdMeF/
xUAitJZtqmo05pptYTWuKuyn7RikOoMJzcbBrHaJB/eyD+U87vkKwgoXhgjG8ANo
5HnHpQnDemN4AOBfvNtvFmp+48Kq9VOH+wZGDodo51hJ5jmAlyJLgUZE6Llybzo9
doRUzbo/6LOKqpNasavOTBeeizOFtKCBMpSxMTUO3TyO3vIRJkkljcStKxmdXPsi
3oWnNjGjx+fGSABmboc9uZdr1kpLDQLGrtIwejRml8pYhWApHsB0knpTOCgzXgqd
uoZYHodguOBmpGs+9rvJIMSoG3EKbdpItCi5nCUo+MaJj+qF30gbVgSRzWBwqUZb
blkwfjxcM8UoCiYx1fme77Q3MAu46QFFzeEfvyAM0RS1tNAw8mHBVrxUy2FpePZc
QQQgTMeqxZQYLv/AlKVBjKXSZAc2tdfteydcQl/VdPtbjqi+XKfQ/Z8AAB+1L8ZN
9wG8dT3UXdBaGZj8OZoTJhjElvJ5kvu+PlGJgEmxy2P8KmhHXXKNXWJborqg2wRBq
PyPsTRWGRlhu6LMkl7MTpHf6HSOasG+WlKzzhPjQRqsnGn+zEBUpnbwiwpEGGBsI
AAAAMgWCaQzb+QUJAA0vAAKbBCIhBh9Ba4E0J3asXFTAwdKEMz/GipiPW86Jrqqg+
4vmz2NLeAAAAAHTeEGkOXCrhUMNivX5qs3If1NkLjpoppsxuzC4pkM0wGGvKKgRf
abFhlMmh9lrgpQdXaV+DFXIqbDbf10yT0zuaV4VS4qWWztNpbMMPGJw9/EEUL
-----END PGP PRIVATE KEY BLOCK-----

Acknowledgements

Document History

Authors' Addresses

Daniel Kahn Gillmor
American Civil Liberties Union
Email: dkg@fifthhorseman.net

Holger Krekel
merlinux gmbh
Email: holger@merlinux.eu

Friedel Ziegelmayer
n0 Inc.
Email: me@dignifiedquire.com