MTG

Kryptoagilität in der Standardisierung der IETF

Dr. Falko Strenzke, MTG AG

CAST Workshop Kryptoagilität, 2025-05-15, Darmstadt

SecurITy made in Germany
Trust Seal
www.teletrust.de/itsmig

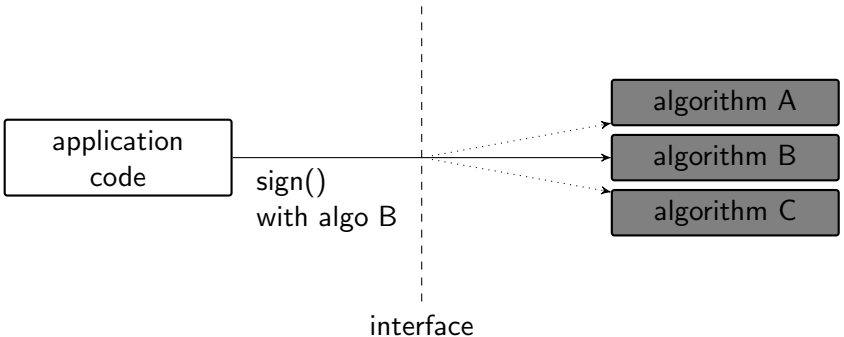
MTG

The new PQC Algorithms

- KEM (Verschlüsselung)
  - ML-KEM
- Signaturen
  - ML-DSA
  - SLH-DSA #
  - XMSS / LMS #

Cryptographic Agility relies on stable interfaces

- To easily exchange an algorithm, the interface must not change
- Constant interface definition: treat algorithm as black box



Price question

MTG

Which interface gets “more complicated” through PQC? Encryption or signatures?

KEM Interfaces

MTG

- KEM
  - Interface change compared to encryption scheme
  - KEM encrypts a random key
  - random key used to encrypt session key or data directly

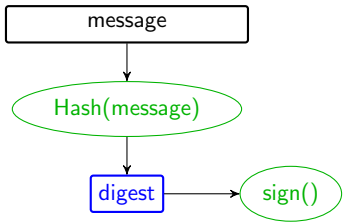
Complexity of KEM interface

MTG

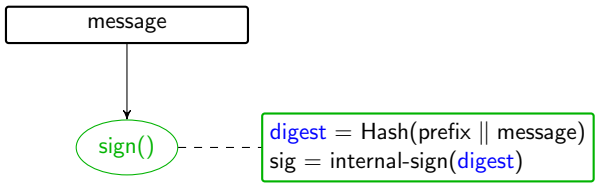
- currently only one KEM: ML-KEM
- no interface changes to be expected for future KEMs

Signatures: From “hash-then-sign” to “pure signing”

- traditional algorithms RSA, ECDSA:
  - build by Hash-then-sign paradigm
    1. hash the message stream
    2. sign the hash



- modern / PQC design paradigm is not hash-then-sign
  - advantage: increased collision resistance of signature scheme
  - PQC signature algorithms receive the message stream directly



Signature Interface

	NIST FIPS std.	**IETF (RFCs)**
schemes	ML-DSA, SLH-DSA	XMSS, LMS
context parameter	✔	✖
pre-hash variant[1]	✔	✖

[1] pre-hash variant ≈ hash-then-sign

Signatures: Online Operations

MTG

- Online property (a.k.a stream processing, incremental processing):
  - process the message during signing or verification as a stream, requires only **constant** (small) amount of **memory**
- if not possible, need to store full message
  - sometimes impossible for resource-constrained devices

MTG

- $\mu \leftarrow H(H(\text{public-key}) \| M)$ // *hedges against multi-user attacks*
- then sign μ
- short message representative μ
- FIPS: computing μ needs to be done in a cryptographic module

ML-DSA

MTG

- $\mu \leftarrow H(H(\text{public-key}) \| M)$ // *hedges against multi-user attacks*
- then sign μ
- short message representative μ
- FIPS: computing μ needs to be done in a cryptographic module

SLH-DSA

MTG

- $R \leftarrow \mathrm{PRF}_{\mathrm{msg}}(SK.prf, \mathrm{opt\_rand}, \mathrm{PK.root}, M)$ *// M to hedge against RNG failure*
- $\mathrm{digest} \leftarrow H_{\mathrm{msg}}(R, \mathrm{PK.seed}, \mathrm{PK.root}, M)$ *// improved collision resistance*
- compute signature based on *digest* ...

SLH-DSA

MTG

cannot process message hash as a stream;
need to store M for second computation

- $R \leftarrow \mathrm{PRF}_{\mathrm{msg}}(SK.prf, \mathrm{opt\_rand}, \mathrm{PK.root}, M)$ // *M to hedge against RNG failure*
- $\mathrm{digest} \leftarrow H_{\mathrm{msg}}(R, \mathrm{PK.seed}, \mathrm{PK.root}, M)$ // *improved collision resistance*
- compute signature based on *digest* ...

$\longrightarrow$ pure SLH-DSA: signing, verification not online, no streaming

SLH-DSA

cannot process message hash as a stream;
need to store M for second computation

- $R \leftarrow \mathrm{PRF}_{\mathrm{msg}}(SK.prf, \mathrm{opt\_rand}, \mathrm{PK.root}, M)$ // *M to hedge against RNG failure*
- $\mathrm{digest} \leftarrow H_{\mathrm{msg}}(R, \mathrm{PK.seed}, \mathrm{PK.root}, M)$ // *improved collision resistance*
- compute signature based on *digest* ...

⟶ pure SLH-DSA: signing, verification not online, no streaming
⟶ need to resort to pre-hash (hash-then-sign) variant for memory efficiency

SLH-DSA

MTG

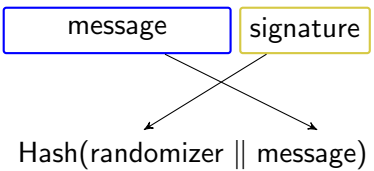
cannot process message hash as a stream;
need to store M for second computation

- $R \leftarrow \mathrm{PRF}_{\mathrm{msg}}(SK.prf, \mathrm{opt\_rand}, \mathrm{PK.root}, M)$ *// M to hedge against RNG failure*
- $\mathrm{digest} \leftarrow H_{\mathrm{msg}}(R, \mathrm{PK.seed}, \mathrm{PK.root}, M)$ *// improved collision resistance*
- compute signature based on *digest* ...

$\longrightarrow$ pure SLH-DSA: signing, verification not online, no streaming
$\longrightarrow$ need to resort to pre-hash (hash-then-sign) variant for memory efficiency
→ no increased collision resistance at all

Hash-based signature verification

- all hash-based signature schemes prefix a randomization string (“randomizer”) to the signed data
- randomizer is encoded in the signature



Properties of signatures schemes

	Algorithm specification (NIST, RFC) (final)				X.509 (cert, CRL) specification (LAMPS drafts, not final)			
	ext. msg. repr.	**pre-hash**	**sign-online**	**verify-online**	**ext. msg. repr.**	**pre-hash**	**sign-online**	**verify-online**
ML-DSA	✔(μ)	✔	✔[2]	✔[2]	✔(μ)	✖	✔[2]	✔[2]
SLH-DSA	✖	✔	(✔)[3]	(✔)[4]	✖	✔	(✔)[3]	(✔)[4]
XMSS/LMS	✖	✖	✔	✖[5]	✖	✖	✔	✖

[2] initialisation of ML-DSA stream-processing for message requires public-key

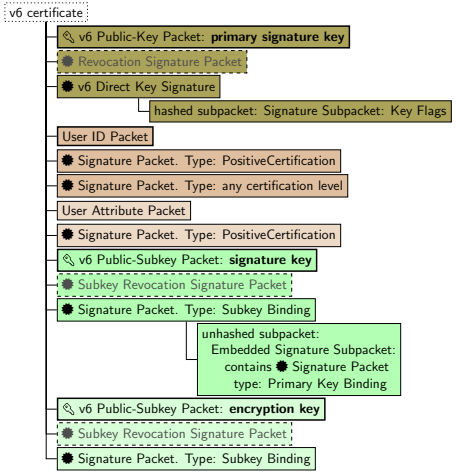
[3] online-signing with SLH-DSA only with pre-hash variant

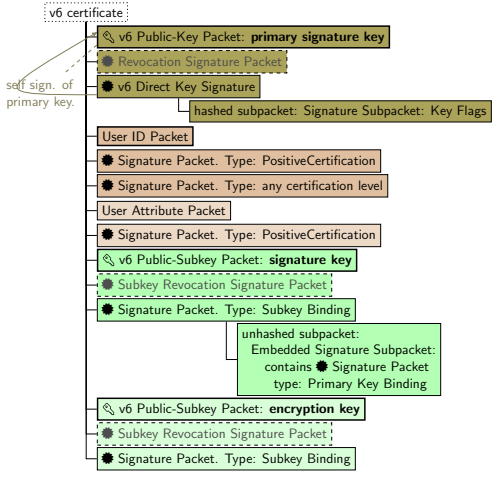
[4] online-verification with SLH-DSA only with pre-hash (or signature-first or randomizer-first)

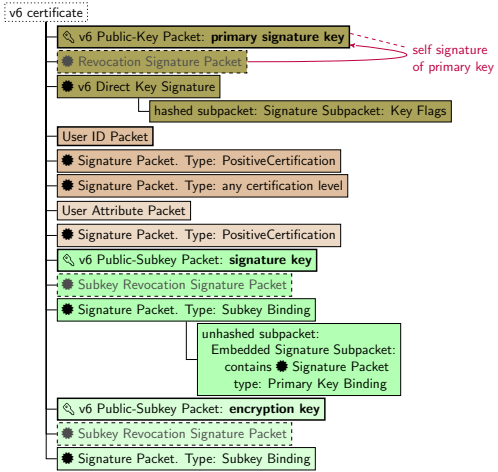
[5] XMSS/LMS online-verification only with pre-hashing on application level or with signature-first

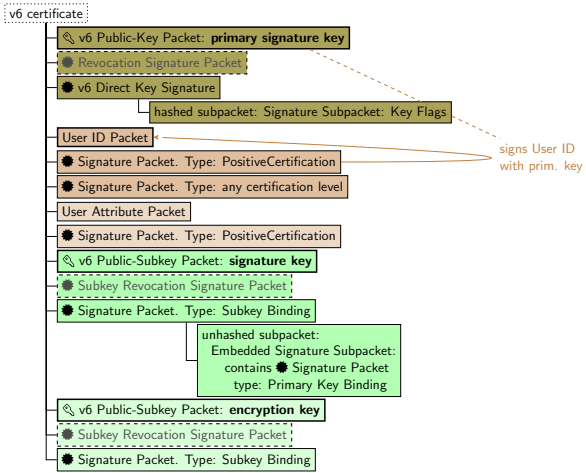
OpenPGP Protocol

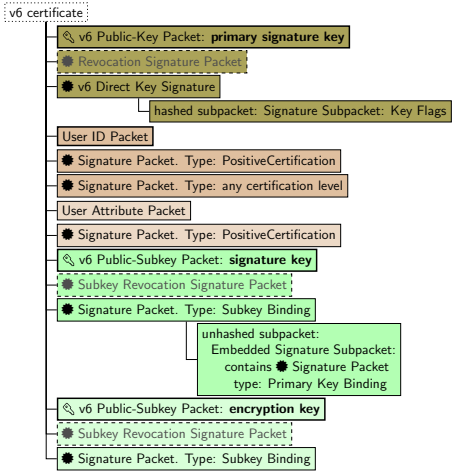
MTG

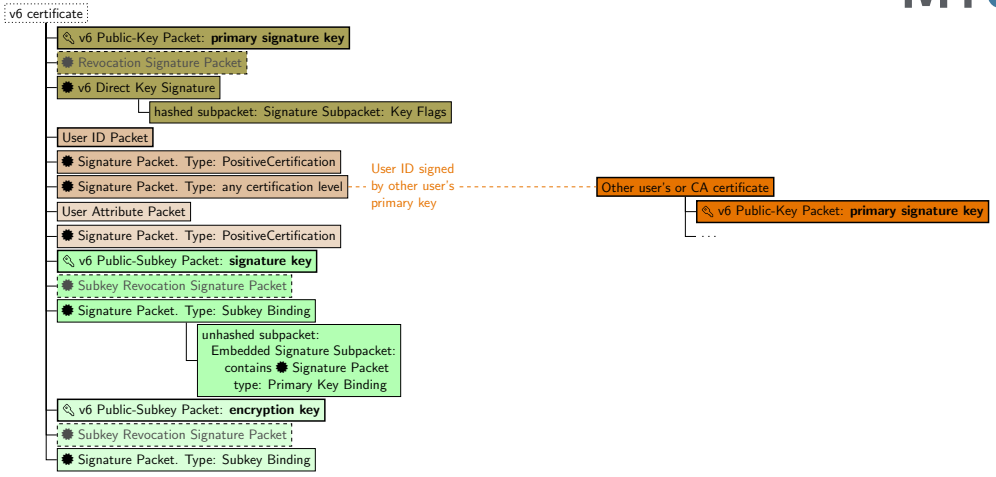

v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding

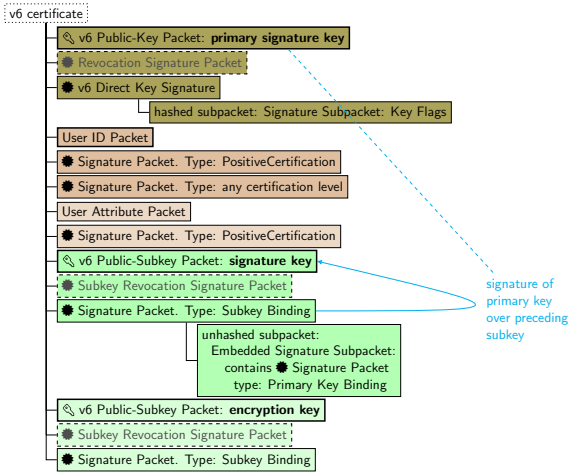

v6 certificate
self sign. of primary key.
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding

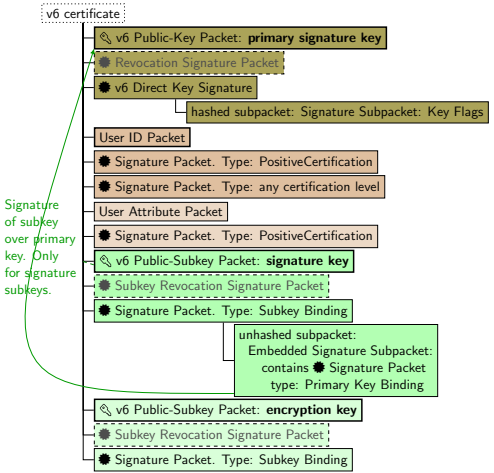

v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
self signature
of primary key
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding

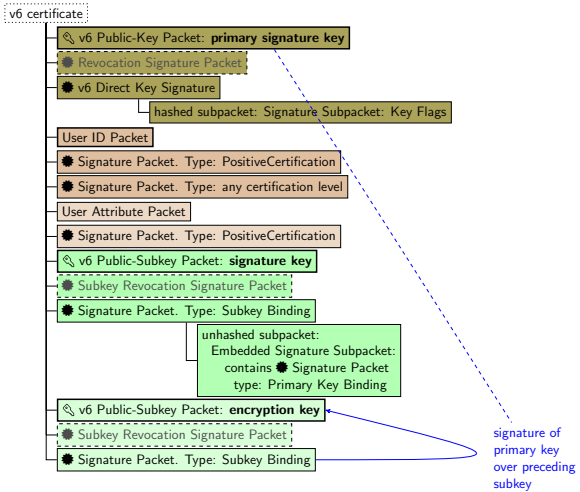

v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
signs User ID
with prim. key

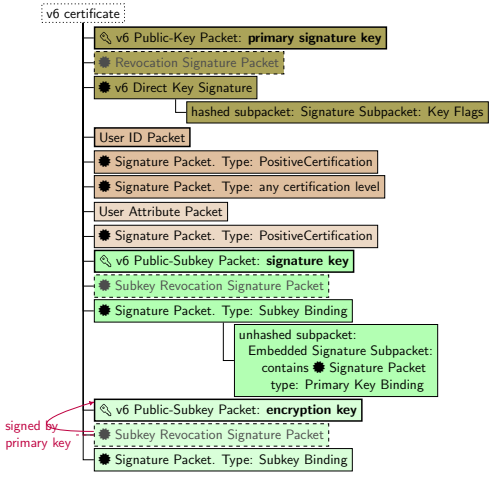

v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding

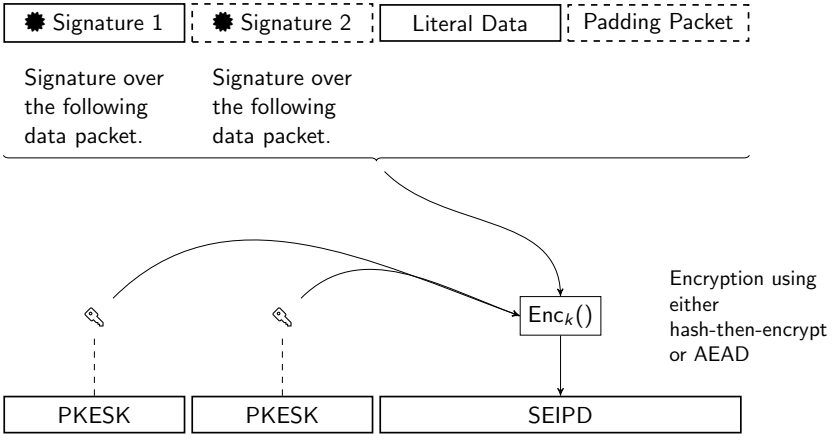

v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
User ID signed
by other user's
primary key
Other user's or CA certificate
v6 Public-Key Packet: primary signature key
...


v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
signature of
primary key
over preceding
subkey


v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
Signature of subkey over primary key. Only for signature subkeys.


v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
signature of primary key over preceding subkey


v6 certificate
v6 Public-Key Packet: primary signature key
Revocation Signature Packet
v6 Direct Key Signature
hashed subpacket: Signature Subpacket: Key Flags
User ID Packet
Signature Packet. Type: PositiveCertification
Signature Packet. Type: any certification level
User Attribute Packet
Signature Packet. Type: PositiveCertification
v6 Public-Subkey Packet: signature key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding
unhashed subpacket:
Embedded Signature Subpacket:
contains Signature Packet
type: Primary Key Binding
v6 Public-Subkey Packet: encryption key
signed by
primary key
Subkey Revocation Signature Packet
Signature Packet. Type: Subkey Binding


Signature 1
Signature 2
Literal Data
Padding Packet
Signature over the following data packet.
Signature over the following data packet.
$Enc_k()$
Encryption using either hash-then-encrypt or AEAD
PKESK
PKESK
SEIPD
Public Key Encrypted Session Key. session key encrypted to public key of user A
session key encrypted to public key of user B
Symmetrically Encrypted Integrity Protected Data Packet. (message encrypted with session key)

PQC@Thunderbird

MTG

- BSI Projekt 480 “PQC@Thunderbird”
- Standardisierung von PQC in OpenPGP
- Implementierung in Thunderbird
- MTG mit TU Eindhoven
- Dez. 2021 - Nov. 2025

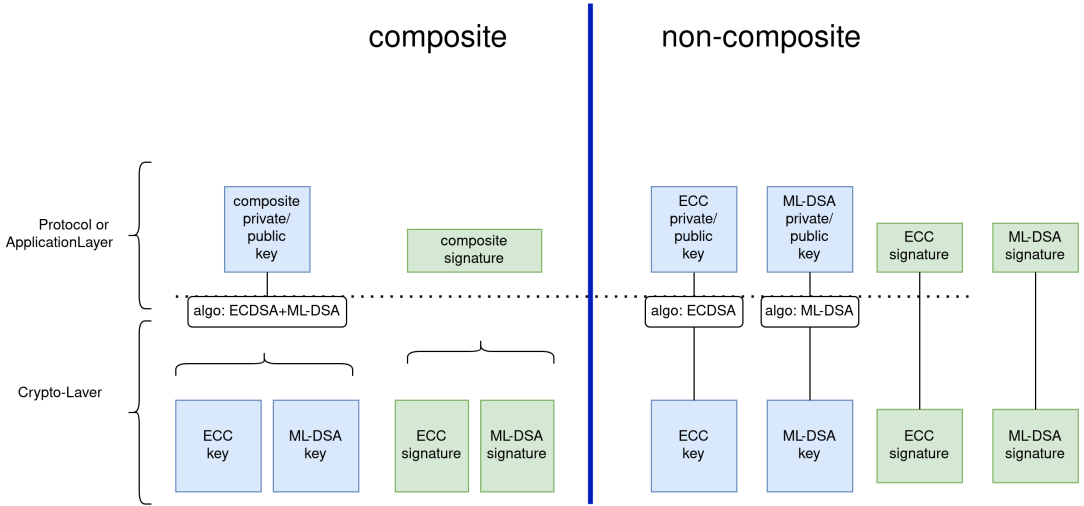
PQ/T hybrid schemes

MTG

- European governments recommend/require pairing PQC schemes with traditional algorithms
  - Exception: Hash-based algorithms **#**
- NIST: "new stuff sometimes gets broken"[6]

[6] https://csrc.nist.gov/csrc/media/Presentations/2025/draft-sp-800-227-overview/images-media/sp-800-227-galagic.pdf#page=11

PQ/T hybrid signatures



PQC-Integration: Ursprüngliche Idee aus dem Projekt

MTG

- in jedem Fall: hybrid = “multi-algorithm” (außer SLH-DSA)
- erster Ansatz: vollständig generisch
  - Neue Algorithmen ID für ML-KEM, ML-DSA, SLH-DSA
  - Non-composite für generische Kombinationen von Algorithmen
    - Mehrere Signaturen in OpenPGP schon vorgesehen
    - z.B. RSA + ML-DSA unter Angabe der jeweiligen Algorithm IDs
  - und unter Angabe der Parameter:
    - RSA **4096**
    - ML-DSA **192 bit**

Umentscheidungen bei PQC Integration

MTG

- Umentscheidung 1
  - Composite
  - Algorithm ID: Feste Kombinationen aus PQ / T
- Umentscheidung 2
  - Sicherheitsparameter auch mit Algorithm ID festschreiben
  - Algorithm ID = 30 → “ML-DSA-65+Ed25519”

Two PQC Drafts

draft-ietf-openpgp-pqc
(adopted, WGLC)
BSI, MTG, Proton

algorithms	requ.	security
ML-KEM-768+X25519	**MUST**	192 / 128
ML-KEM-1024+X448	SHOULD	256 / 224
ML-DSA-65+Ed25519	**MUST**	192 / 128
ML-DSA-87+Ed448	SHOULD	256 / 224
SLH-DSA-SHAKE-128s	MAY	128
SLH-DSA-SHAKE-128f	MAY	128
SLH-DSA-SHAKE-256s	MAY	256

draft-ehlen-openpgp-nist-bp-comp
(**not** adopted)
BSI, MTG, NIST

algorithms	security
ML-KEM-512+ECDH-**NIST-P-256**	128 / 128
ML-KEM-768+ECDH-**NIST-P-384**	192 / 192
ML-KEM-1024+ECDH-**NIST-P-384**	256 / 192
ML-KEM-768+ECDH-**brpP256r1**	192 / 128
ML-KEM-1024+ECDH-**brpP384r1**	256 / 192
ML-DSA-44+ECDSA-**NIST-P-256**	128 / 128
ML-DSA-65+ECDSA-**NIST-P-384**	192 / 192
ML-DSA-87+ECDSA-**NIST-P-384**	256 / 192
ML-DSA-65+ECDSA-**brpP256r1**	192 / 128
ML-DSA-87+ECDSA-**brpP384r1**	256 / 192

all “MAY”

LAMPS WG: X.509, CMS, etc.

MTG

- Ähnlichkeiten:
  - Auch nur feste Algorithmenkombinationen
  - Auch Festlegung der Sicherheitsparameter durch Algorithm-ID
- Unterschiede:
  - XMSS und LMS
  - Alle 12 SLH-DSA Parameter
  - ML-DSA und ML-KEM als Standalone
  - Composite
    - Kombinationen mit RSA und ECDSA (NIST, Brainpool)
    - Berücksichtigung des Falcon Signaturverfahrens (noch nicht finalisiert von NIST)

Kryptoagilität in OpenPGP und X.509/CMS

MTG

- “niemand (in der IETF) will freie Kombinierbarkeit der Algorithmen”
  - composite multi-algorithm
- OpenPGP: starke Einschränkung der neuen Algorithmen gefordert
  - Interoperabilität steht im Vordergrund
- X.509/CMS: Größere Kryptoagilität (?) durch größere Algorithmen- und Parameterauswahl
  - Große Zahl von abgeleiteten Protokollen
  - Vielfalt steht im Fordergrund

MTG

Die neuen PQC Algorithmen

Das OpenPGP Protokoll

PQC für OpenPGP

PQC für X.509 und CMS

Agilität bei Sicherheitseigenschaften bei OpenPGP und X.509 / CMS

OpenPGP und LibrePGP

Zusammenfassung

Protocol security features and agility

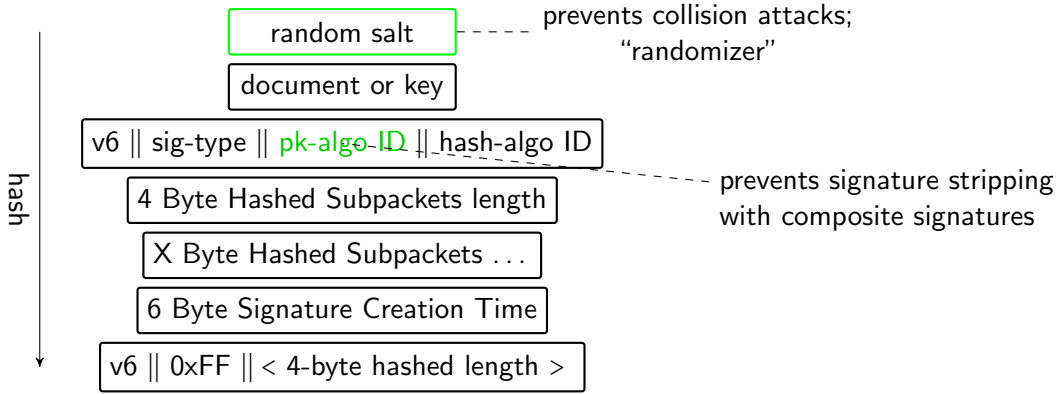
MTG

- Some protocol security features ...
  - generally relevant
  - relevant to PQC
- How to achieve security updates to protocols?
  - "the agile way" (?): pick new features explicitly
  - protocol version upgrade

RFC 9580 OpenPGP – signatures

MTG

v6 signatures — hashed data



OpenPGP uses hash-then-sign

MTG

- PQC algorithms used with hash-then-sign
- no disadvantage due to random salt (= randomizer)
- good cryptographic agility compared to X.509

Non-Separability in OpenPGP Signatures

MTG

v6 signature packet

v6 = 0x06
sig-type 0x00
1B pk-algo = hybrid-...
1B hash-algo
2B hashed subpacket length
hashed subpackets
2B unhashed subpacket length
unhashed subpackets
2B checksum for hash-value
algorithm-specific signature data

Non-Separability in OpenPGP Signatures

MTG

v6 signature packet

hashed and signed as meta data

- **v6 = 0x06**
- **sig-type 0x00**
- **1B pk-algo = hybrid-...**
- **1B hash-algo**
- **2B hashed subpacket length**
- **hashed subpackets**

- **2B unhashed subpacket length**
- **unhashed subpackets**
- **2B checksum for hash-value**
- **algorithm-specific signature data**

sig 1 | sig 2

Non-Separability in OpenPGP Signatures

MTG

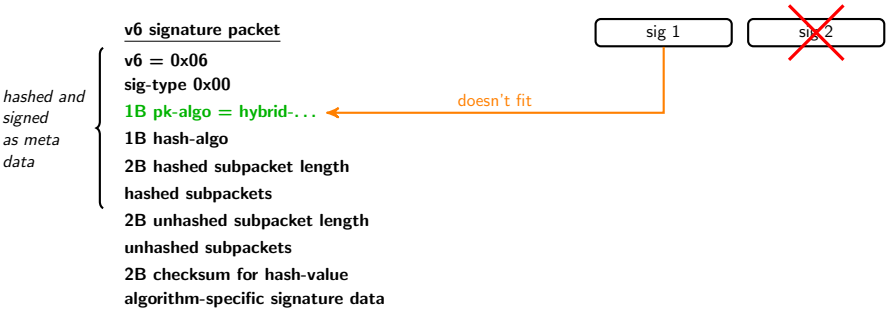
v6 signature packet

hashed and signed as meta data

- **v6 = 0x06**
- **sig-type 0x00**
- **1B pk-algo = hybrid-...**
- **1B hash-algo**
- **2B hashed subpacket length**
- **hashed subpackets**

- **2B unhashed subpacket length**
- **unhashed subpackets**
- **2B checksum for hash-value**
- **algorithm-specific signature data**

Non-Separability in OpenPGP Signatures

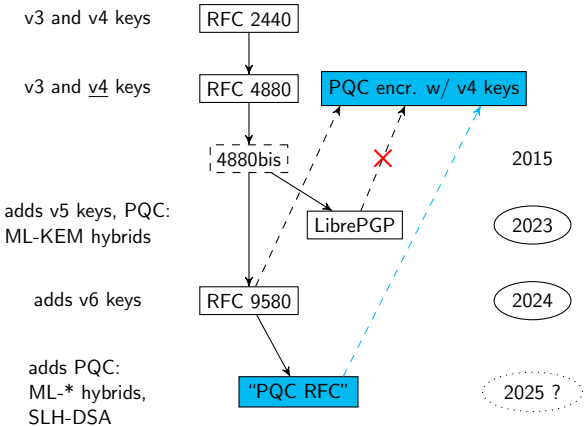


CMS – Signatures

- Cryptographic Message Syntax (e.g. S/MIME)
  - X.509 certificates
  - Signatures
  - Public-key encryption
- No random salt
  - Pre-hash signatures potentially vulnerable to hash-collision attacks
- No meta-data is hashed
  - Public-key algorithm not fixed by signature
  - → In case of hybrid: signature stripping attacks require extra countermeasure
    - Not fully sound if legacy signature remains
- CMS legacy problem
  - (theoretical) signature forgery problem with Signed Attributes[7]
    - Alternative views of what was exactly signed
    - Was not solved with context parameter of new PQC algorithms

[7] An EUF-CMA violation, see https://eprint.iacr.org/2023/1801

v6/v5/v4 PQC



- Main goal: fast adoption of PQC encryption
- GnuPG LibrePGP standard features incompatible ML-KEM hybrids
  - LibrePGP (outside IETF) is a recent fork of OpenPGP due to WG-internal unresolved technical "issues"

LibrePGP's approach to PQC

MTG

- So far, only ML-KEM is specified (no signatures)
- ML-KEM in combinations with ECDH
  - with any EC curve out 16
- greater parameter choice than
  - OpenPGP
  - X.509 / CMS

Protocol security features in comparison

	CMS	OpenPGP RFC 4880 v4	OpenPGP RFC 9580 v6	LibrePGP
AEAD	⚠ requires opt. mechanism from RFC 9709[8]	⚠ no AEAD	✔ sound AEAD	⚠ unmitigated weakness[8]
enhanced signature collision resistance	⚠ not for pre-hash variants or CMS with Signed Attributes[9]	⚠ no	✔ random salt	⚠ no

[8] https://cic.iacr.org/p/2/1/19

[9] Signed Attributes ≈ pre-hash

Zusammenfassung

- Die neuen PQC Signaturalgorithmen haben Variabilität im Interface
  - Betrifft v.a. die Speichereffizienz → hauptsächlich relevant für eingebettete Systeme
  - Hat Einfluss auf die Kryptoagilität (Austauschbarkeit)
  - Wechsel auf pre-hash kann notwendig werden
- IETF eher kritisch gegenüber generischer Kombinierbarkeit
  - bisher (fast) nur Vorschläge für composite multi-algorithm
  - OpenPGP: vergleichsweise restriktive Algorithmenauswahl, wenige Kombinationen
  - X.509 / CMS: deutlich größere Auswahl an Algorithmen und Kombinationen
- Protokolldesign und PQC
  - OpenPGP: hashing von Metadaten (salt, Algorithm ID)
    - Erlaubt u.a. Verwendung von hash-then-sign ohne Nachteile
  - X.509 / CMS
    - hash-then-sign nur mit schlechterer Sicherheitsaussage
- Sicherheitsfeatures in Protokollen
  - Zuverlässige Verbesserungen erfordern Sprung in der Protokollversion
  - "Agiles Nachrüsten" von Gegenmaßnahme eher problematisch

MTG

Thank you for your attention

Dr. Falko Strenzke
falko.strenzke@mtg.de
+49 6151 8000-24

MTG AG
www.mtg.de