Post-Quantum

Cryptography Conference

Architecting PKI Hierarchies for Graceful PQ Migration

Public Key Infrastructures (PKIs) are intricate systems to design, deploy, and maintain. As post-quantum cryptography (PQC) becomes a reality, one of the most challenging decisions will be algorithm selection. Historically, this has been straightforward—for example, using RSA-2048-SHA256 consistently throughout the PKI hierarchy. In a PQC world, this approach becomes less viable due to tradeoffs that may require different algorithms or parameters at each layer. For instance, long-term secure algorithms might be optimal for root CAs, high-performance algorithms for issuing CAs, and bandwidth-efficient algorithms for end entities. Hybrid approaches further complicate these decisions, offering additional security or migration flexibility depending on the use case. This talk explores a "toolbox" of migration mechanisms for X.509 and presents example PKI hierarchies tailored to specific scenarios, illustrating how to navigate the complexity of algorithm choices for a graceful transition to post-quantum cryptography.

Mike Ounsworth
Software Security Architect at Entrust

January 15 and 16, 2025 - Austin, TX (US) | Online

PKI Consortium

Architecting PKI Hierarchies for Graceful PQ Migration

Mike Ounsworth

Software Security Architect

PKI Consortium

January 2025

Austin, Texas

ENTRUST

SECURING A WORLD IN MOTION

Agenda

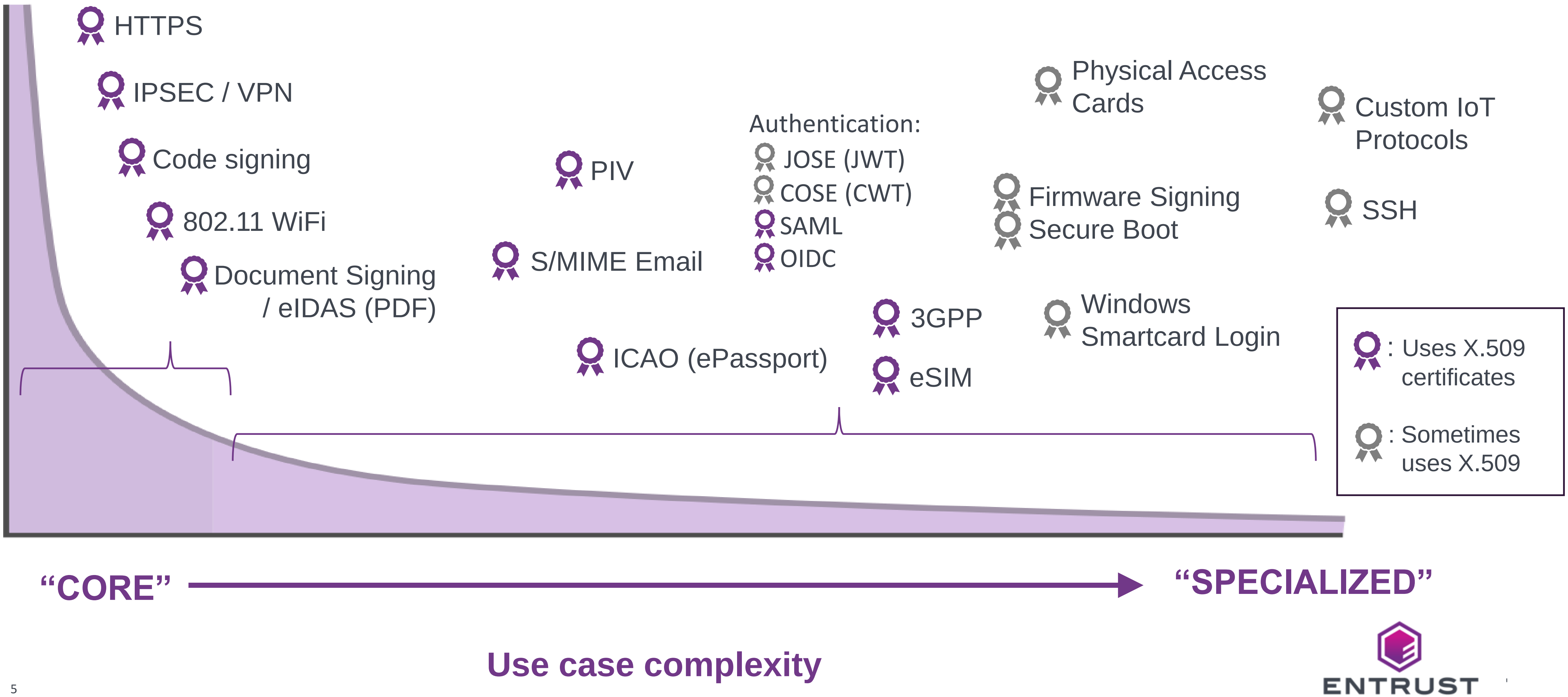
- The Migration Problem and the Long Tail of X.509
- Which algorithm(s)?
- Toolbox
  - Heterogeneous PKIs
  - Hybrid PKIs for Ease of Migration
  - Hybrid PKIs for Security

ENTRUST

The Migration Problem and the Long Tail of X.509

ENTRUST

The Long Tail of X.509 Usage



● HTTPS

● IPSEC / VPN

● Code signing

● 802.11 WiFi

● Document Signing / eIDAS (PDF)

● PIV

● S/MIME Email

● ICAO (ePassport)

Authentication:

○ JOSE (JWT)

○ COSE (CWT)

● SAML

● OIDC

● 3GPP

● eSIM

○ Physical Access Cards

○ Firmware Signing

○ Secure Boot

○ Windows Smartcard Login

○ Custom IoT Protocols

○ SSH

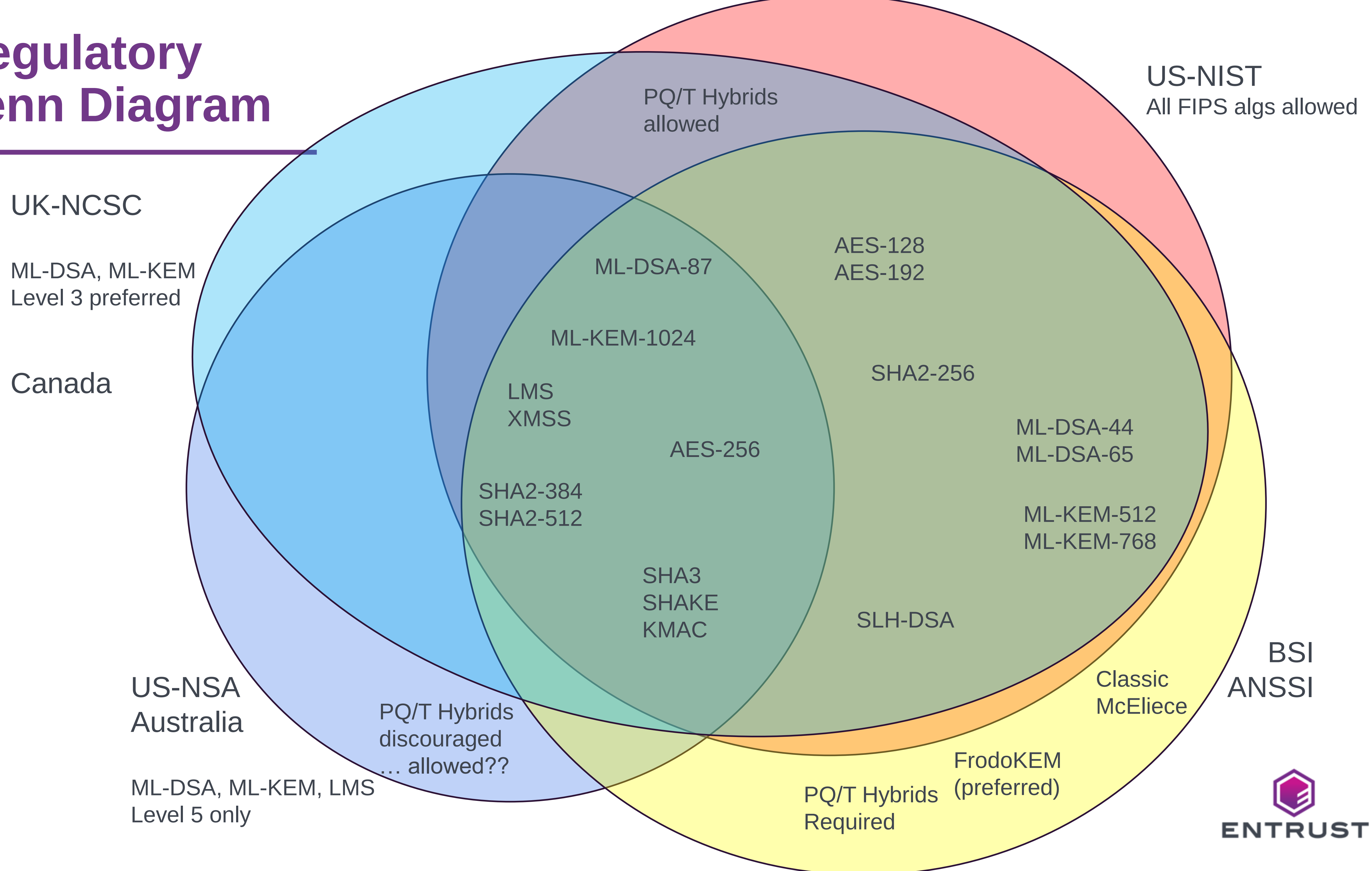
● : Uses X.509 certificates

○ : Sometimes uses X.509

“CORE” → “SPECIALIZED”

Use case complexity

Regulatory Venn Diagram



US-NIST
All FIPS algs allowed

UK-NCSC

ML-DSA, ML-KEM
Level 3 preferred

Canada

US-NSA
Australia

ML-DSA, ML-KEM, LMS
Level 5 only

BSI
ANSSI

PQ/T Hybrids
allowed

ML-DSA-87

ML-KEM-1024

LMS
XMSS

AES-256

SHA2-384
SHA2-512

SHA3
SHAKE
KMAC

AES-128
AES-192

SHA2-256

ML-DSA-44
ML-DSA-65

ML-KEM-512
ML-KEM-768

SLH-DSA

PQ/T Hybrids
discouraged
… allowed??

Classic
McEliece

FrodoKEM
(preferred)

PQ/T Hybrids
Required

Migration Considerations

Some questions you might ask yourself:

- Do you have compliance / regulatory requirements?
- Do you have long-term data security requirements (20+ years) ?
- Do you control your environment tightly enough to upgrade everything all at once, or do you need a staged migration?
  - If so, do you use only negotiated protocols (ex.: TLS) that will handle downgrade for legacy clients, or do you need to get backwards compatibility from the certificates themselves?
- Etc, etc, etc.

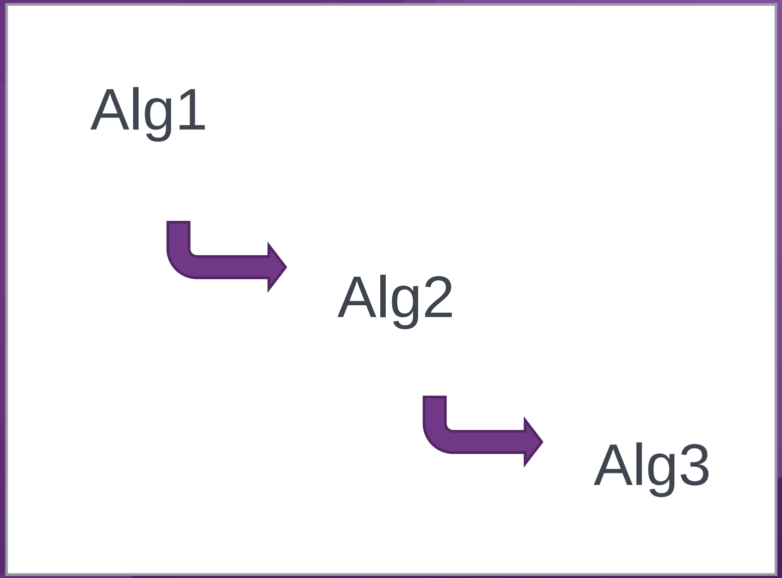
PQC Migration Toolbox

ENTRUST

PQC Migration Toolbox

Migration Tool	Description	Use Case
Flag Day	Change everything to PQC on Jan 1st.	You control everything, and can upgrade everything at the same time.
Mixed PKI / Heterogeneous PKI	Use different algorithms at different layers of the PKI	Optimize for long-term security at the top. Optimize for performance / bandwidth at the bottom.
Hybrid: Multiple Certificates / Parallel PKIs	Run independent Traditional and PQ PKIs. Issue everything with two certs.	Cases where clients can negotiate algorithms (TLS, VPN), or can gracefully ignore extra signatures (maybe S/MIME, code signing?).
Hybrid: Catalyst / Chimera / AltPubKey	Put a second pub key and sig into a non-critical X.509v3 extension.	Similar to Parallel PKIs, but where the certificate itself needs to be transparently backwards compatible.
Hybrid: Composite	It looks like a single key, but actually contains two algorithms inside.	Need strong dual algorithm security in protocols that do not natively support negotiation or multiple algorithms.

Mixed PKI



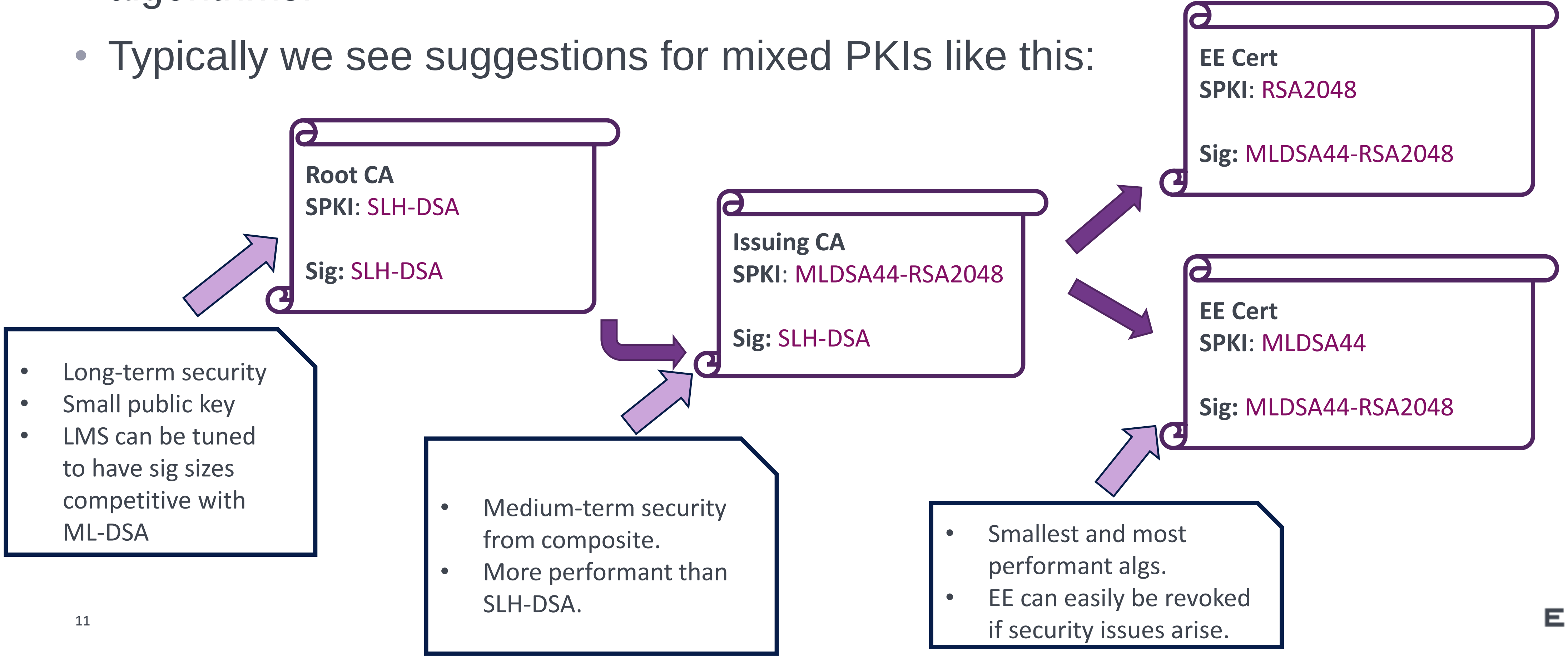
Alg1

Alg2

Alg3

Mixed PKIs

- Idea: Root CAs, Issuing CAs, and End Entity certificates have different requirements for long-term security, throughput, and bandwidth, so we can consider mixing & matching algorithms.
- Typically we see suggestions for mixed PKIs like this:



Root CA
SPKI: SLH-DSA
Sig: SLH-DSA

- Long-term security
- Small public key
- LMS can be tuned to have sig sizes competitive with ML-DSA

Issuing CA
SPKI: MLDSA44-RSA2048
Sig: SLH-DSA

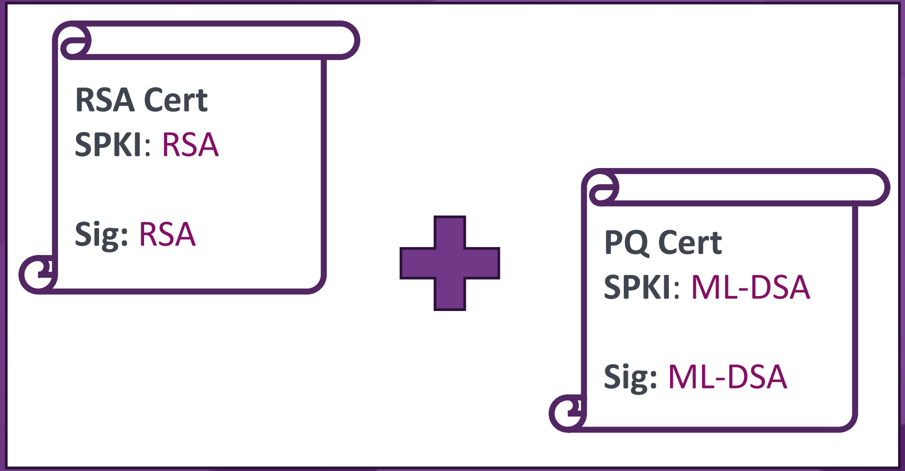
- Medium-term security from composite.
- More performant than SLH-DSA.

EE Cert
SPKI: RSA2048
Sig: MLDSA44-RSA2048

EE Cert
SPKI: MLDSA44
Sig: MLDSA44-RSA2048

- Smallest and most performant algs.
- EE can easily be revoked if security issues arise.

Multiple Certificates



RSA Cert
SPKI: RSA
Sig: RSA

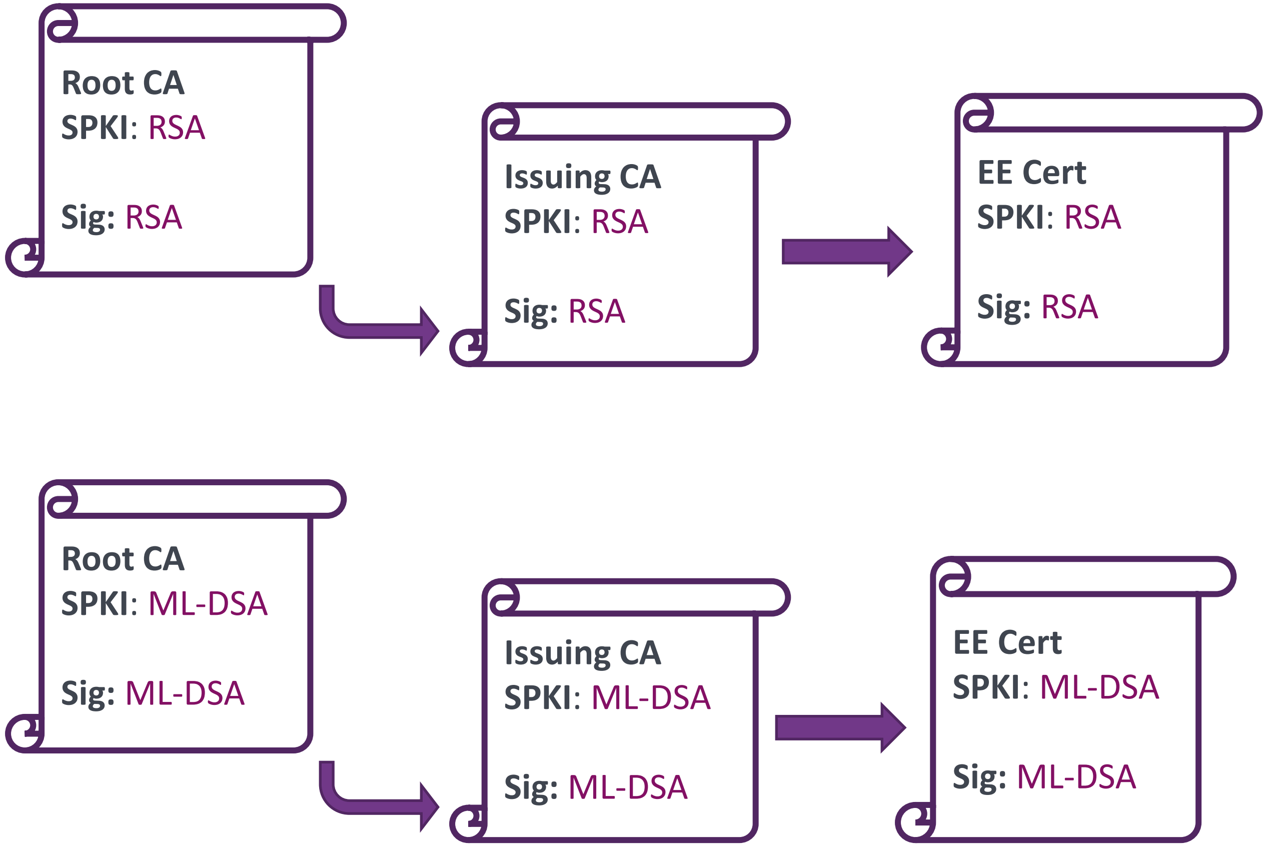
PQ Cert
SPKI: ML-DSA
Sig: ML-DSA

ENTRUST

Multiple Certificates

- Traditional and PQ PKIs operate in parallel.
- Each end entity gets two certificates.

- Pro: simple to deploy.
- Con: forwards all the complexity of hybrid / negotiation to the client.



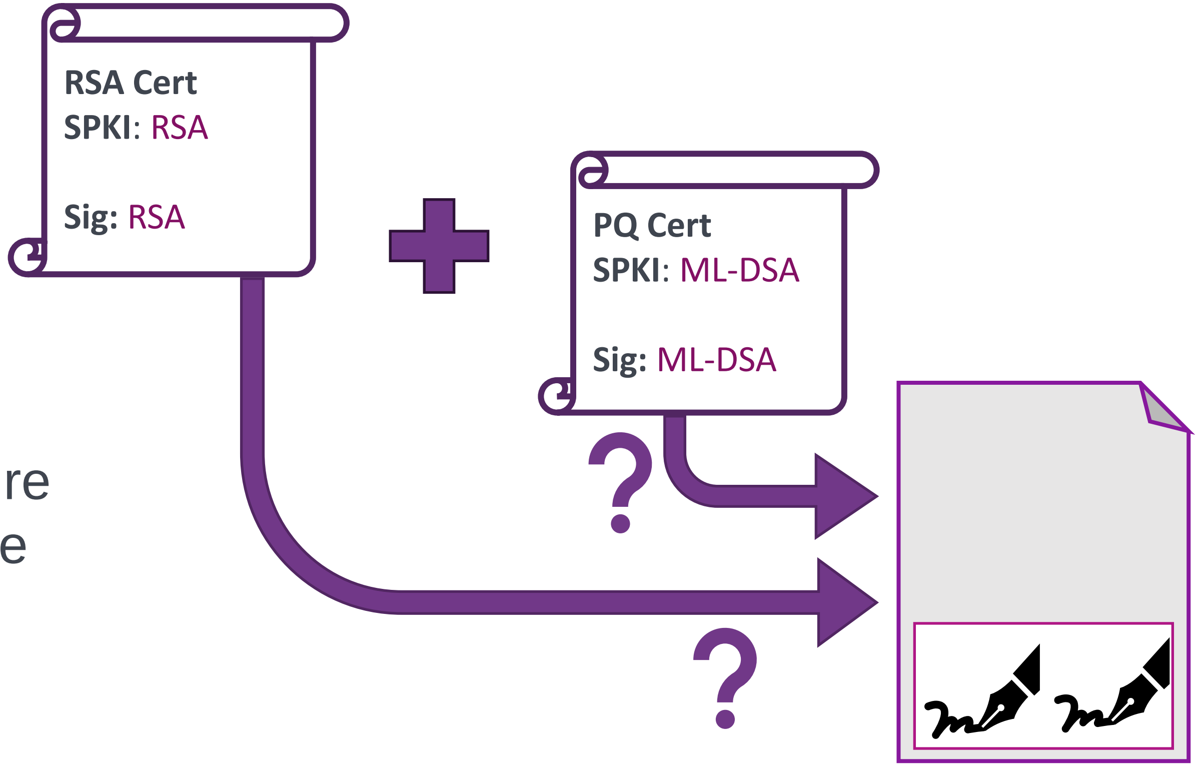
Multiple Certificates

Passive Backwards Compat	Hybrid Security
?	?

- Con: forwards all the complexity of hybrid / negotiation to the client.

- This works well for online negotiated protocols ex.: TLS, VPN, SSH.

- For offline / non-negotiated protocols such as S/MIME email, PDF signing, anything involving hardware smartcards, etc, it becomes tricky to figure out what kind of signature these two certificates are supposed to produce on any given document.



ENTRUST

Multiple Certificates
Passive or Active Backwards Compat

- CMS: **Yes -- passive**
  - Multiple SignerInfos allowed in OR-mode (but not all clients implement this correctly)
- OpenPGP: **Yes -- passive**
  - Multiple Signature packets allowed in OR-mode.
- XML D-Sig: **No?**
  - My quick reading of w3.org/TR/xmldsig-core1 is inconclusive about whether multiple <SignatureValue> elements are allowed, or what the processing rules would be in that case.
- TLS: **Yes -- active**
  - ClientHello advertises “signature_algorithms” and “signature_algorithms_cert” which allows the server to select the correct certificate.
- JWT / CWT: **Maybe?**
  - RFC 7515 (JWS) talks about a “signatures” array. Unclear how well-supported this is.

Multiple Certificates
Hybrid Security

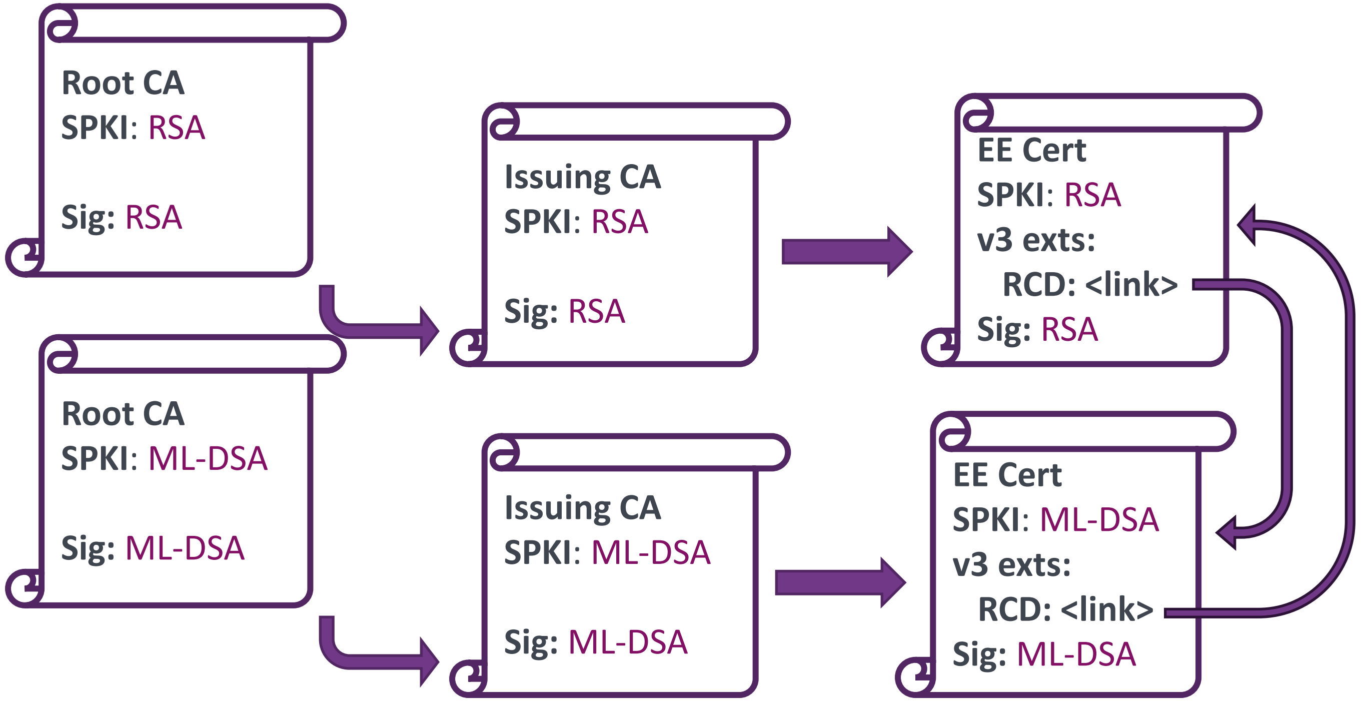
- CMS: Yes
  - RFC 5752 “Multiple Signatures in CMS”.
  - Each CMS SignerInfo is cross-linked to each other SignerInfo.
  - Unclear how well supported this is.
- OpenPGP: No?
- XML D-Sig: No?
- TLS: No?

Multiple Certificates

Discovery -- RelatedCertificateDescriptor

Passive Backwards Compat	Hybrid Security	Discovery
?	?	✓

- Allows for cross-linking End Entity (and potentially CA) certs in one or both directions.
- Can contain a URL so that the related cert is easy to fetch.
- Discovery: if I am holding one of your certificates, it's easy for me to discover the other one.



Root CA
SPKI: RSA
Sig: RSA

Issuing CA
SPKI: RSA
Sig: RSA

EE Cert
SPKI: RSA
v3 exts:
RCD: <link>
Sig: RSA

Root CA
SPKI: ML-DSA
Sig: ML-DSA

Issuing CA
SPKI: ML-DSA
Sig: ML-DSA

EE Cert
SPKI: ML-DSA
v3 exts:
RCD: <link>
Sig: ML-DSA

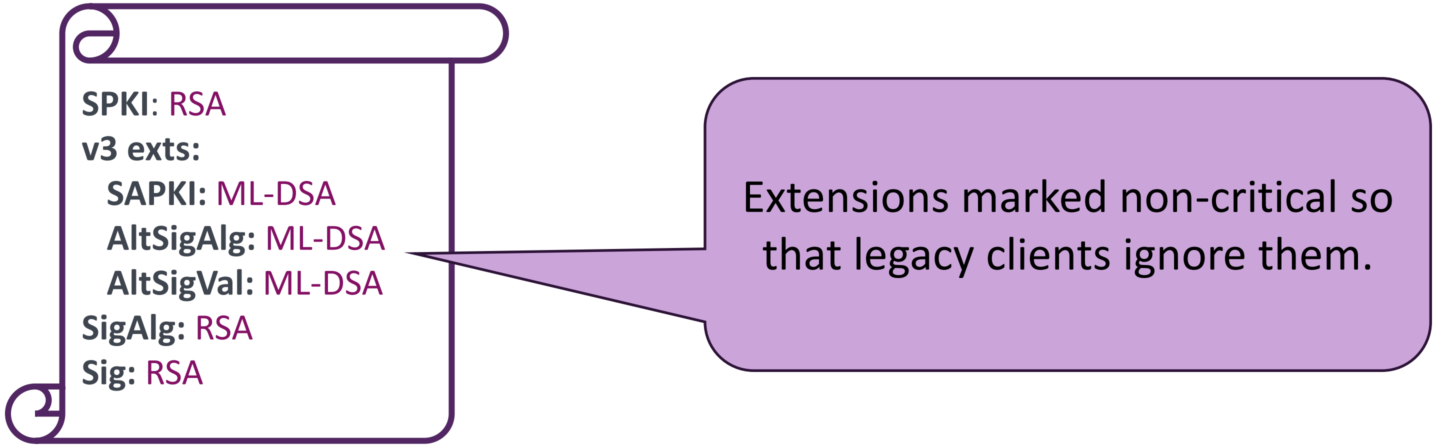
https://datatracker.ietf.org/doc/draft-lamps-okubo-certdiscovery/

subjectAltPublicKeyInfo / altSignatureAlgorithm

aka “ITU-T X.509 2019 Hybrid”
aka “ISARA CATALYST”

subjectAltPublicKeyInfo / altSignatureAlgorithm (aka "ISARA CATALYST" or "Chimera")

- (sometimes referred to as "hybrid certs", but this is confusing for obvious reasons)
- New X.509 v3 extensions: `subjectAltPublicKeyInfo`, `altSignatureAlgorithm`, `altSignatureValue` do exactly the same thing as their "primary" equivalents, but allow for a second alternate public key and signature to be carried in a certificate.



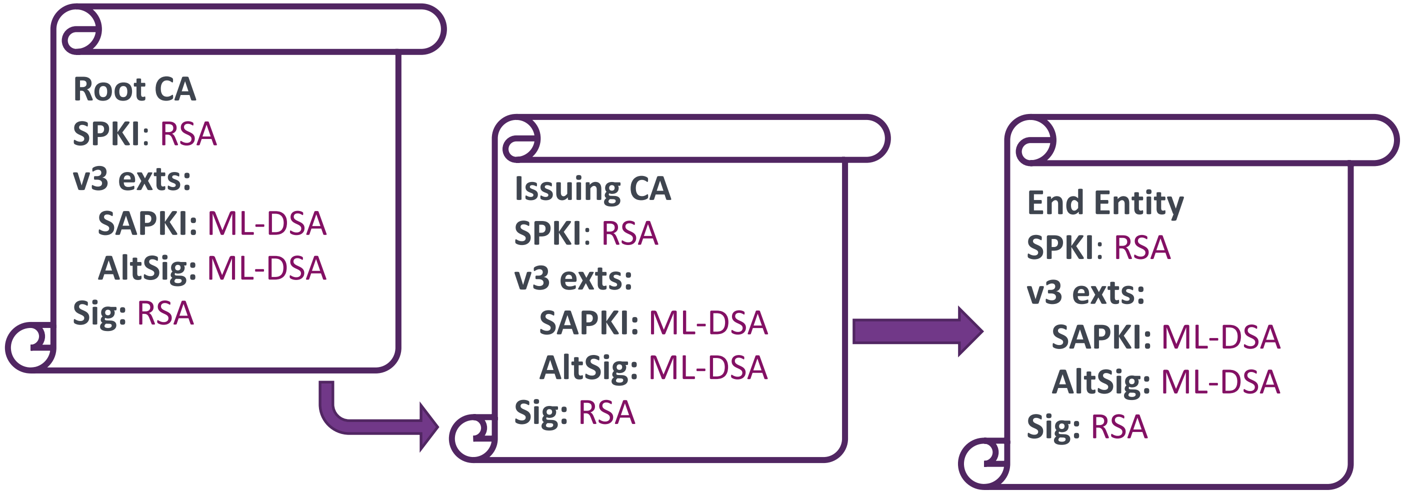
SPKI: RSA
v3 exts:
  SAPKI: ML-DSA
  AltSigAlg: ML-DSA
  AltSigVal: ML-DSA
SigAlg: RSA
Sig: RSA

Extensions marked non-critical so that legacy clients ignore them.

ITU-T X.509 2019, section 7.2.2 "Multiple cryptographic algorithms for public-key certificates"

subjectAltPublicKeyInfo -- AltPubKey

- Simple to deploy:



Root CA
SPKI: RSA
v3 exts:
  SAPKI: ML-DSA
  AltSig: ML-DSA
Sig: RSA

Issuing CA
SPKI: RSA
v3 exts:
  SAPKI: ML-DSA
  AltSig: ML-DSA
Sig: RSA

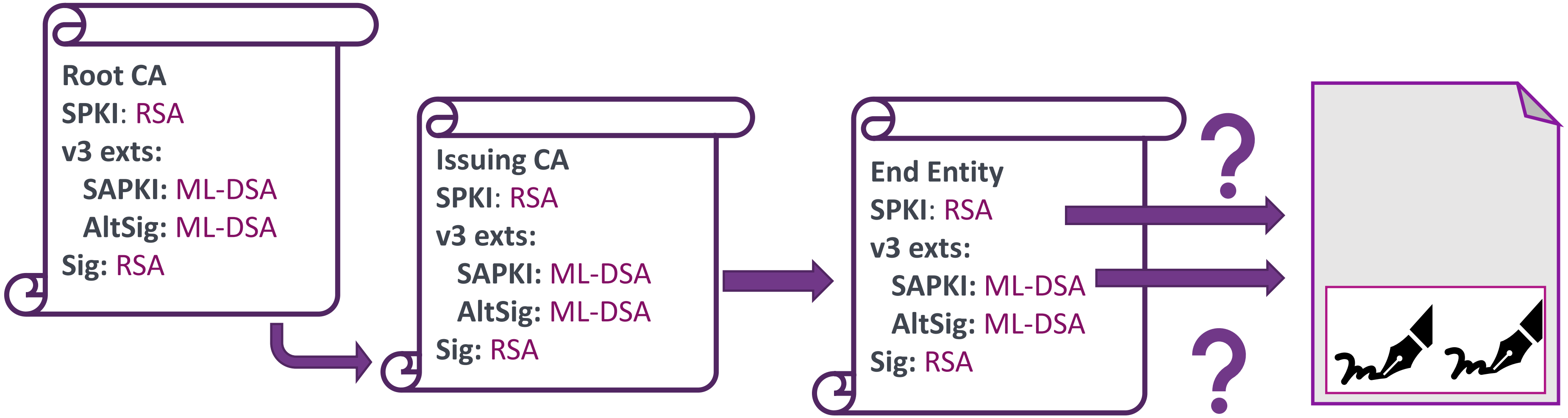
End Entity
SPKI: RSA
v3 exts:
  SAPKI: ML-DSA
  AltSig: ML-DSA
Sig: RSA

ITU-T X.509 2019, section 7.2.2 “Multiple cryptographic algorithms for public-key certificates”

subjectAltPublicKeyInfo -- AltPubKey

Passive Backwards Compat	Hybrid Security
?	?

- Simple to deploy:



Root CA
SPKI: RSA
v3 exts:
SAPKI: ML-DSA
AltSig: ML-DSA
Sig: RSA

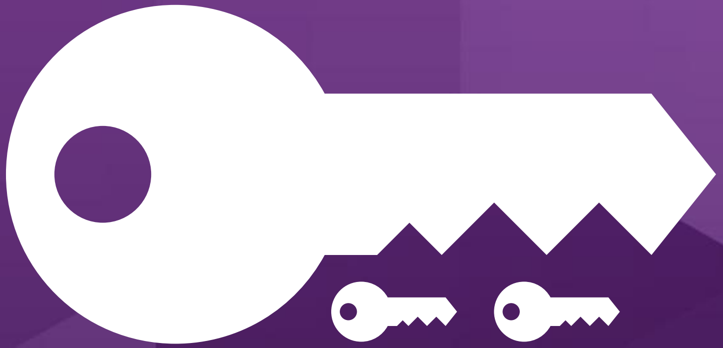
Issuing CA
SPKI: RSA
v3 exts:
SAPKI: ML-DSA
AltSig: ML-DSA
Sig: RSA

End Entity
SPKI: RSA
v3 exts:
SAPKI: ML-DSA
AltSig: ML-DSA
Sig: RSA

- But it’s not clear how you actually sign with it.
- Conclusion: possibly useful in some niche scenarios, but is not straightforward to use.

ITU-T X.509 2019, section 7.2.2 “Multiple cryptographic algorithms for public-key certificates”

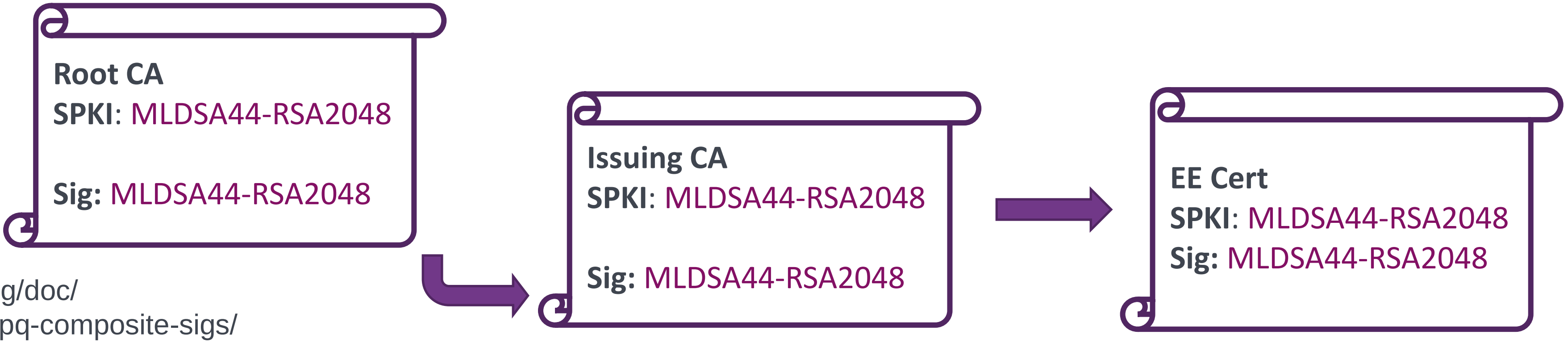
Composite



Composite Signatures

Passive Backwards Compat	Hybrid Security
✗	✓

- Idea: rather than use `id-MLDSA44`, use `id-MLDSA44-RSA2048-PSS-SHA256`.
- If you have a FIPS certified RSA-PSS implementation, then you can have a FIPS certified `id-MLDSA44-RSA2048-PSS-SHA256` without re-certifying anything.
- Needs to be coupled with one of the mechanisms above to achieve **Backwards Compatibility.**
- But, has **Protocol Backwards Compatibility** in that all of the hybridization is handled within the cryptographic algorithm; it still produces one public key and one signature; so no protocol modifications are needed except for supporting the new AlgorithmID OID.



Root CA
SPKI: MLDSA44-RSA2048
Sig: MLDSA44-RSA2048

Issuing CA
SPKI: MLDSA44-RSA2048
Sig: MLDSA44-RSA2048

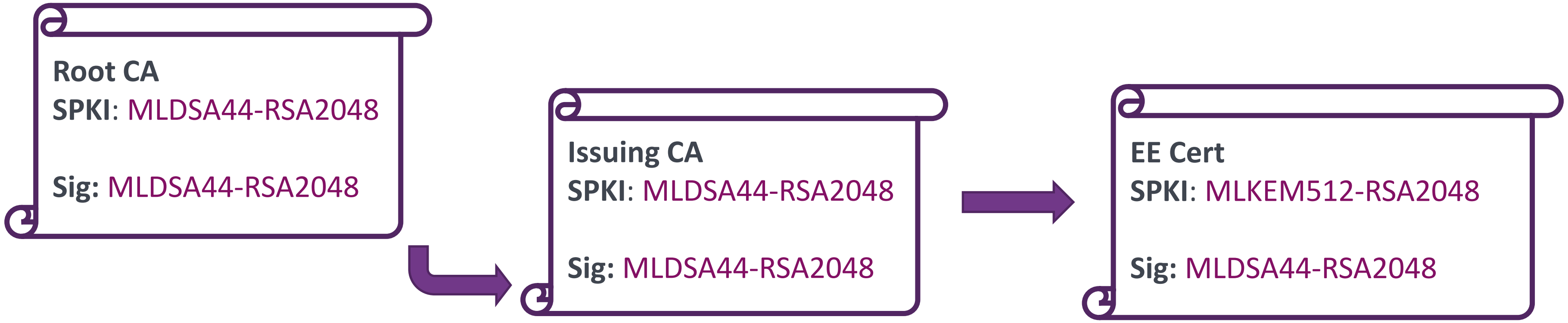
EE Cert
SPKI: MLDSA44-RSA2048
Sig: MLDSA44-RSA2048

https://datatracker.ietf.org/doc/draft-ietf-lamps-pq-composite-sigs/

Composite Encryption

Passive Backwards Compat	Hybrid Security
✗	✓

- Bonus! Composites are the only mechanism that allow for multiple algorithms to be used to perform a single AND-mode encryption.
  - The recipient must have both private keys in order to decrypt the data.
  - An attacker must break both algorithms in order to decrypt the data.
- Idea: rather than use `id-MLKEM512`, use `id-MLKEM512-RSA2048-KMAC128`.



Root CA
SPKI: MLDSA44-RSA2048
Sig: MLDSA44-RSA2048

Issuing CA
SPKI: MLDSA44-RSA2048
Sig: MLDSA44-RSA2048

EE Cert
SPKI: MLKEM512-RSA2048
Sig: MLDSA44-RSA2048

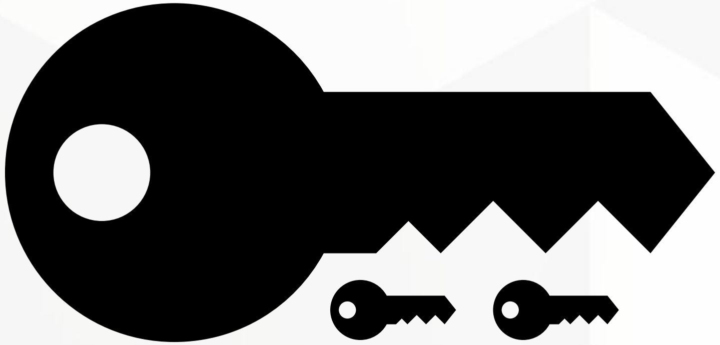
https://datatracker.ietf.org/doc/draft-ietf-lamps-pq-composite-kem/

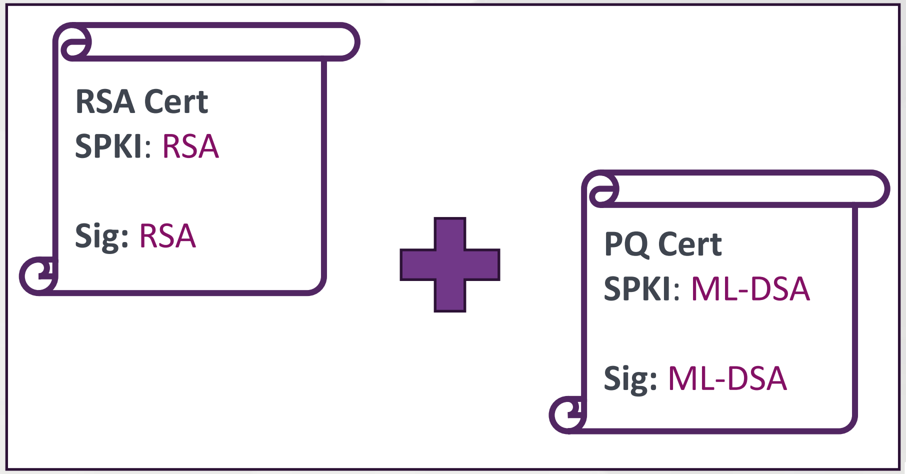
PQC Migration Toolbox

Migration Tool	Description	Use Case
Flag Day	Change everything to PQC on Jan 1st.	You control everything, and can upgrade everything at the same time.
Mixed PKI / Heterogeneous PKI	Use different algorithms at different layers of the PKI	Optimize for long-term security at the top. Optimize for performance / bandwidth at the bottom.
Hybrid: Multiple Certificates / Parallel PKIs	Run independent Traditional and PQ PKIs. Issue everything with two certs.	Cases where clients can negotiate algorithms (TLS, VPN), or can gracefully ignore extra signatures (maybe S/MIME, code signing?).
Hybrid: Catalyst / Chimera / AltPubKey	Put a second pub key and sig into a non-critical X.509v3 extension.	Similar to Parallel PKIs, but where the certificate itself needs to be transparently backwards compatible.
Hybrid: Composite	It looks like a single key, but actually contains two algorithms inside.	Need strong dual algorithm security in protocols that do not natively support negotiation or multiple algorithms.

Thank You!

mike.ounsworth@entrust.com





RSA Cert
SPKI: RSA

Sig: RSA

+

PQ Cert
SPKI: ML-DSA

Sig: ML-DSA

entrust.com

ENTRUST
SECURING A WORLD IN MOTION