Post-Quantum

Cryptography Conference

Update on end-to-end PKI and HSM integrations with ML-DSA

At last years PQC Conference we benchmarked Hardware Securtiy Modules with Dilithium. Now that FIPS-204 is released, it is time to forget about Dilithium and do production level integrations using ML-DSA. This session shows PKI application integration for issuing certificates, with a number of HSMs that are ready for ML-DSA. We will highlight how easy, or hard, it is to integrate using PKCS#11 or REST APIs. Of course there will be benchmarks of certificate issuance comparing ML-DSA against classic algorithms. Let's see what else we are able to squeeze in until January.

Tomas Gustavsson
Chief PKI Officer at Keyfactor

SSL.com PQ SHIELD HID KEYFACTOR ENTRUST

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

PKI Consortium

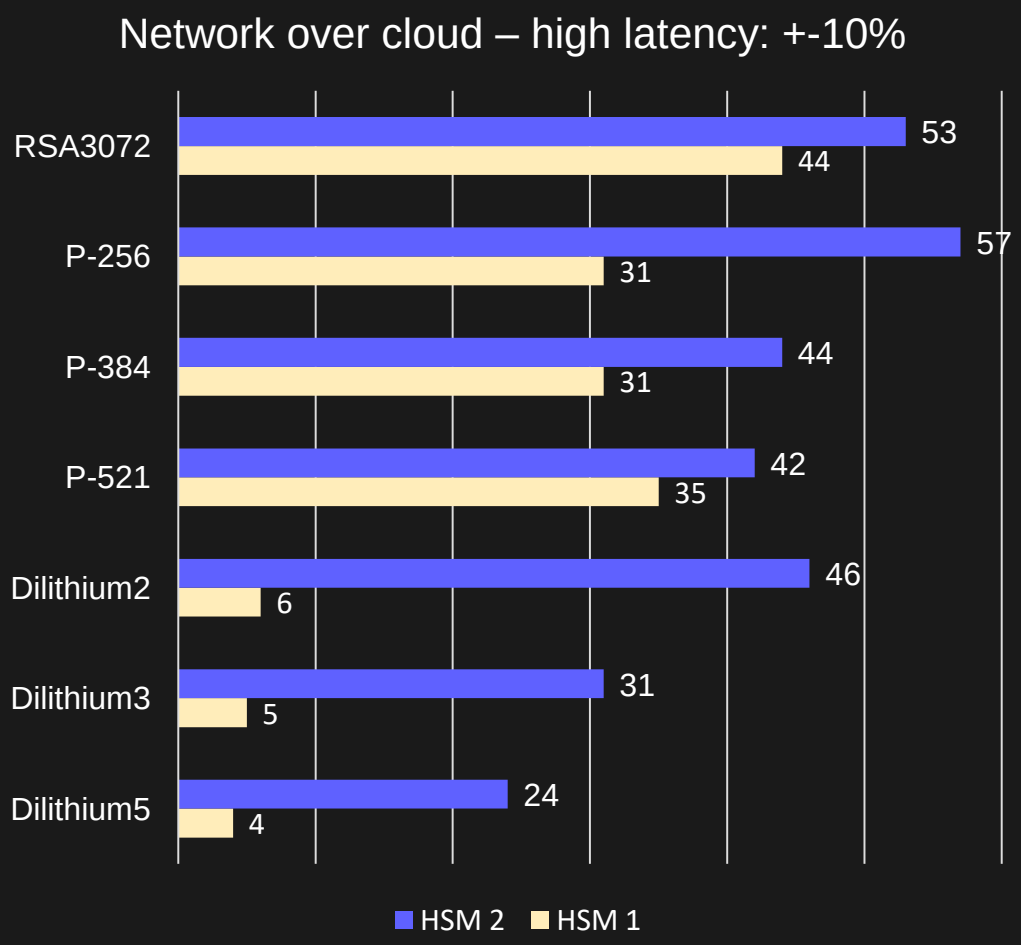
End-to-end PQC and HSM integrations (with PKI)

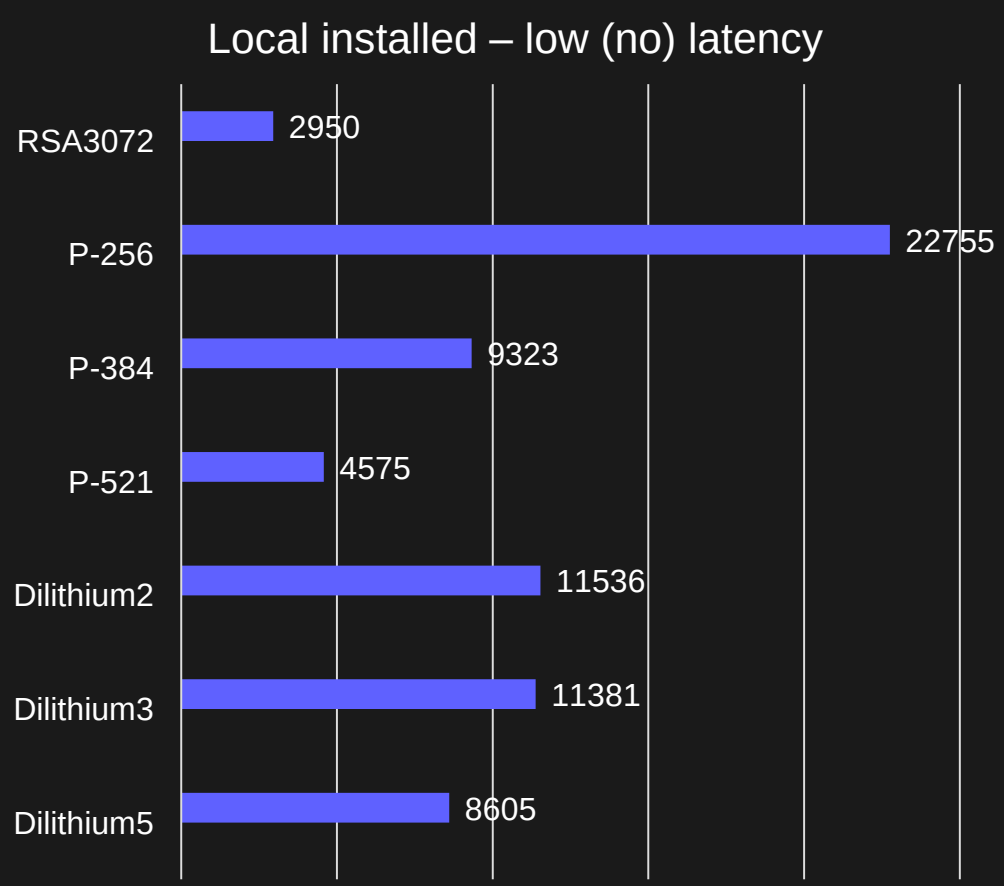
Tomas Gustavsson

Chief PKI Officer

Keyfactor

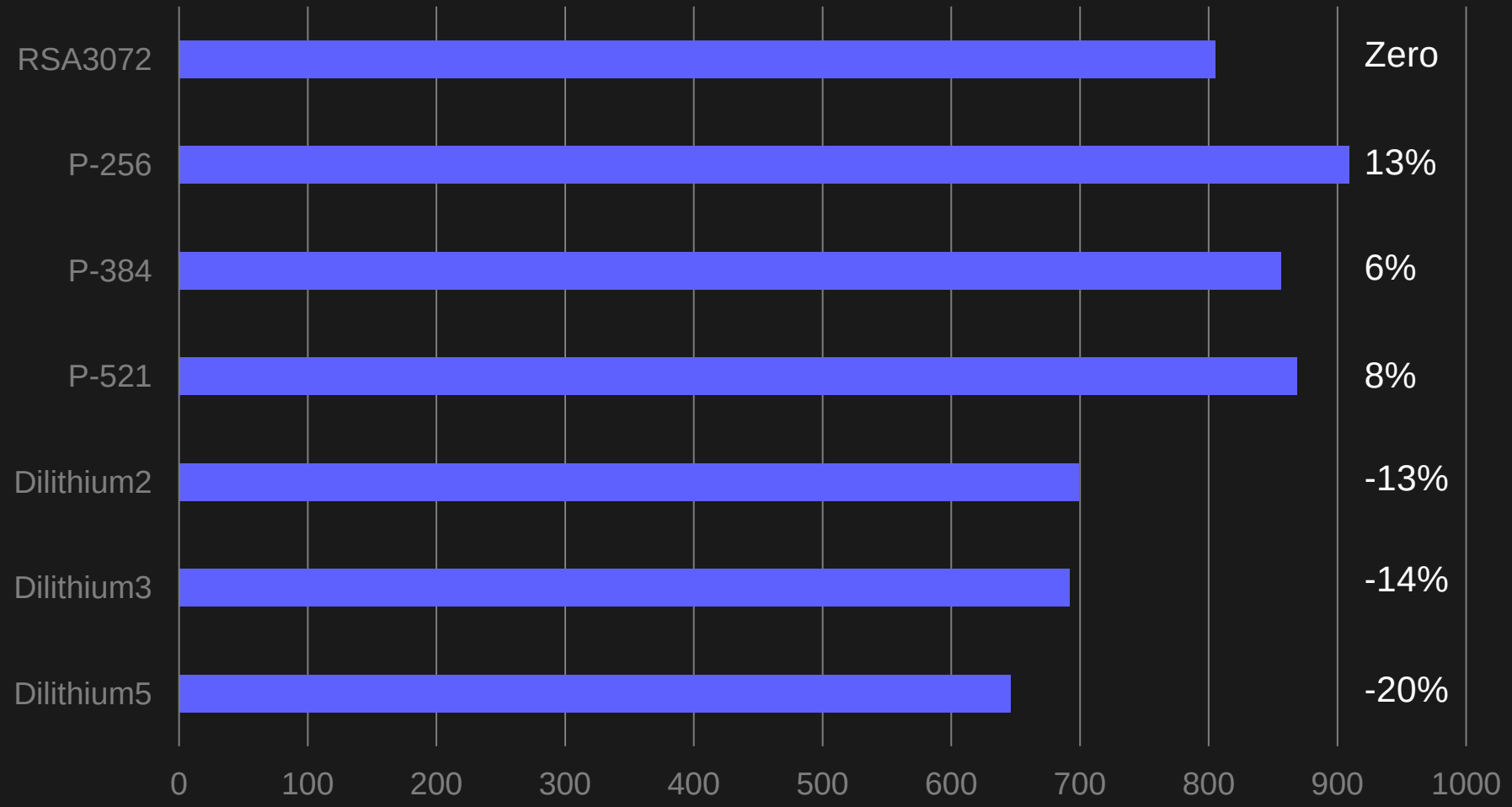
Signing Speed – Recap





KEYFACTOR

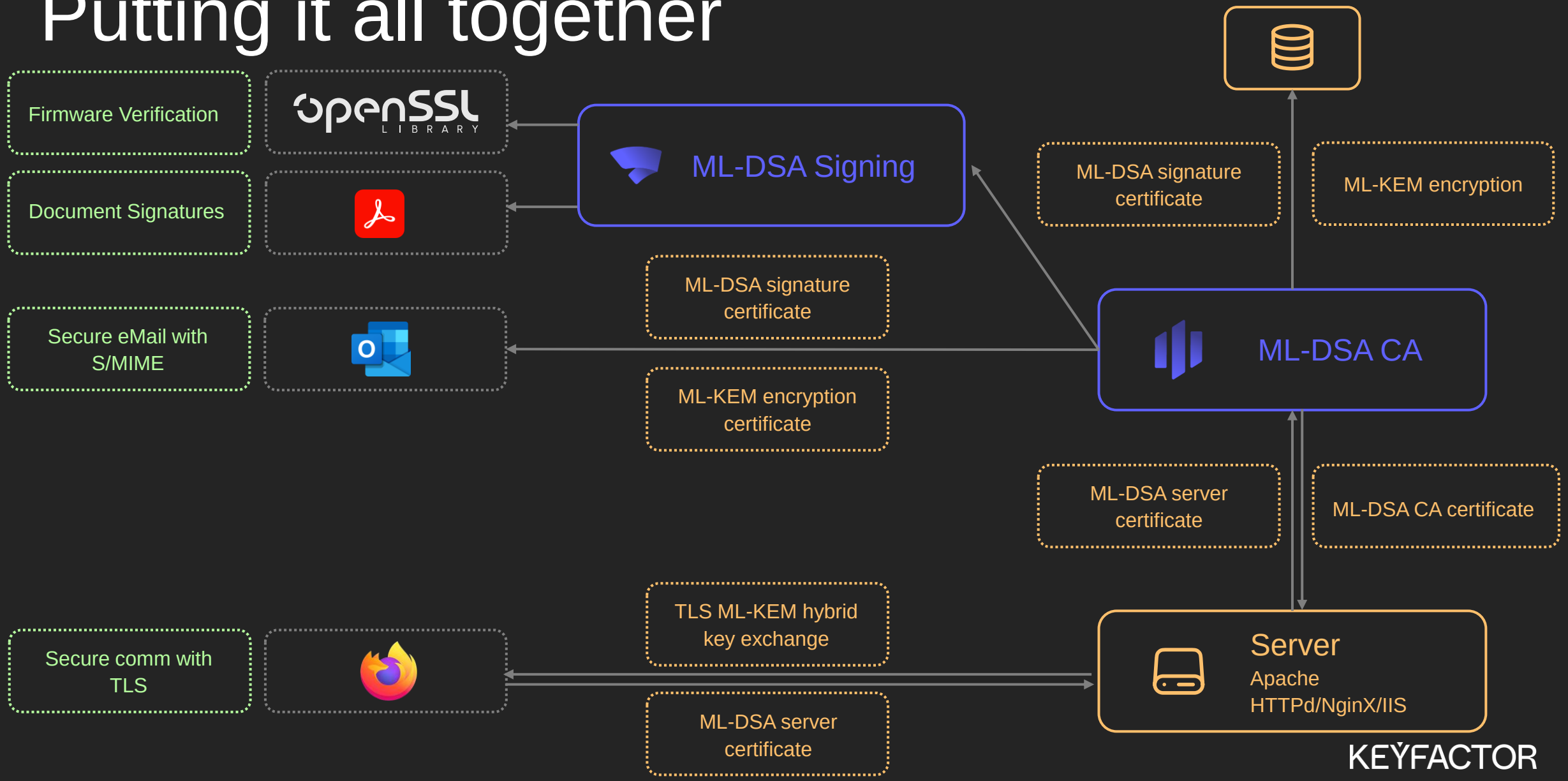
Certificate Issuance - Recap



KEYFACTOR


~~Dilithium~~
ML-DSA / FIPS 204
Quantum ready algorithms
~~SPHINCS+~~
SLH-DSA / FIPS 205
~~Kyber~~
ML-KEM / FIPS 203
KEYFACTOR

Putting it all together



What about PKI and Signing?

PKI and Digital Signatures always use HSMs in production.

You will notice the difference between different tokens...

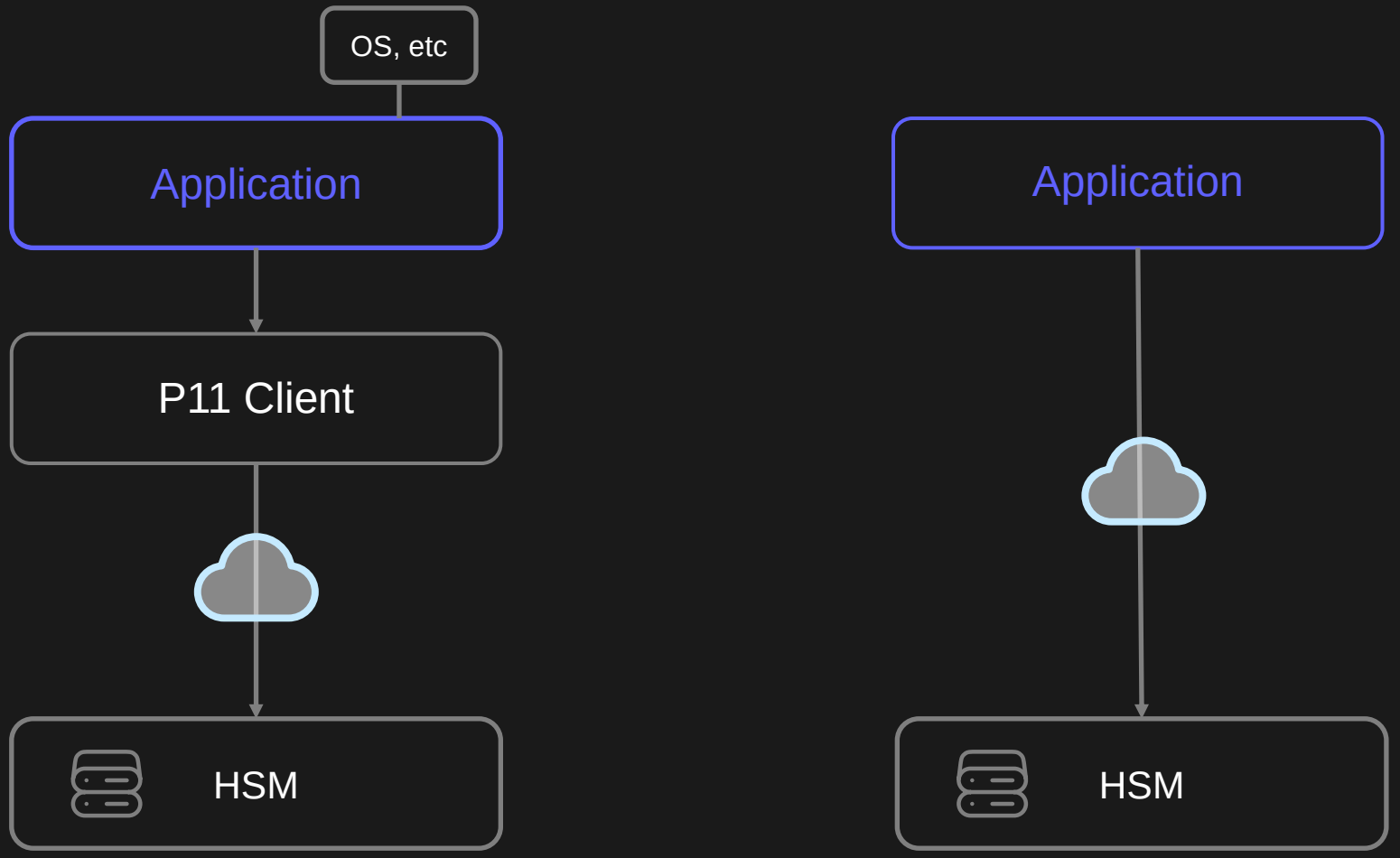
- Using SoftHSM is in-memory
- PCI HSMs are in-memory
- Network based HSMs pass the network stack and wire (latency)
- Cloud HSMs pass the Internet

There is a reason why hash-and-sign is efficient

There is also a reason why hash-and-sign is legacy

KEYFACTOR

REST vs PKCS#11

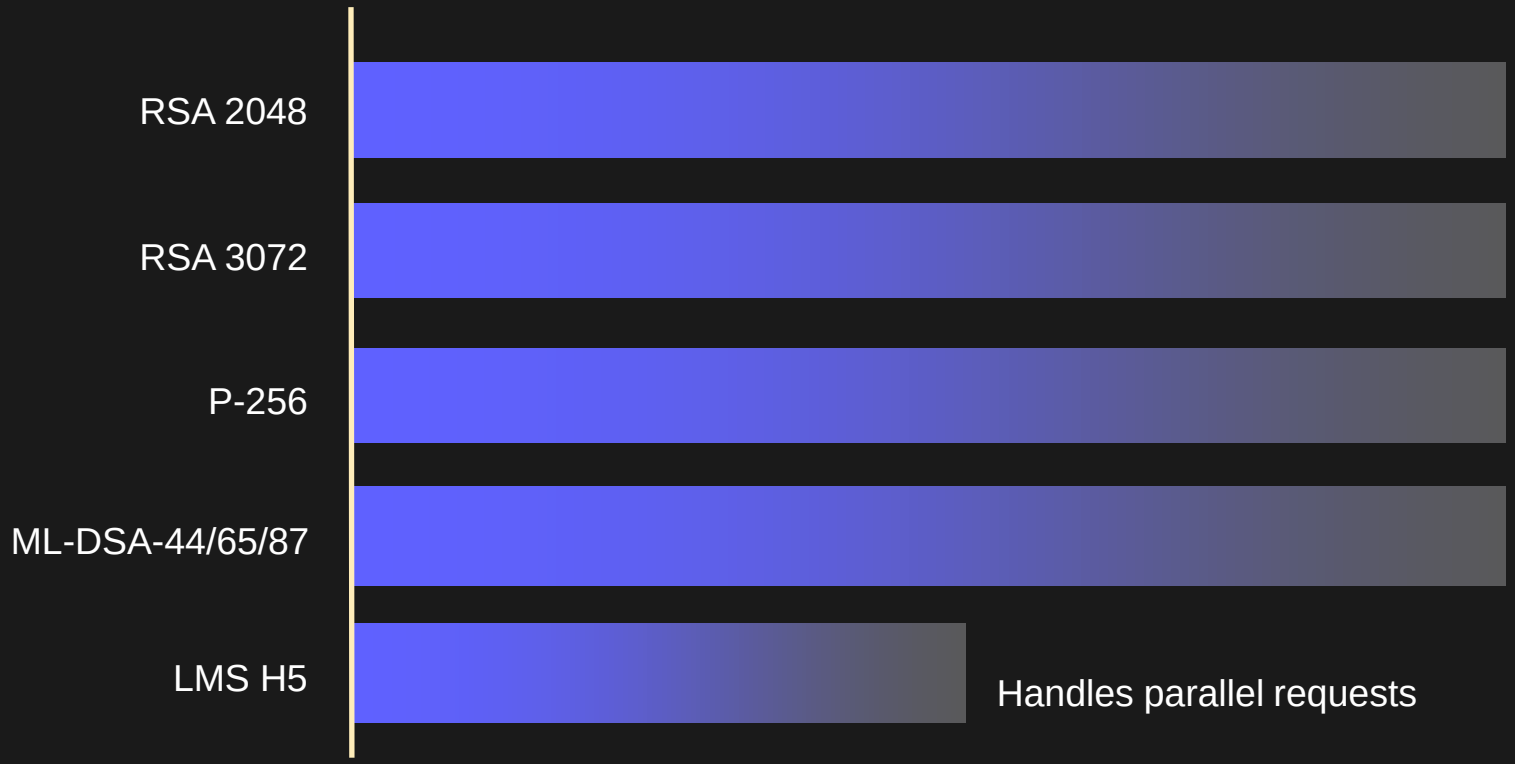


KEYFACTOR


PQC
in Practice
KEYFACTOR

Certificate Issuance HSM 1

REST remote Internet



KEYFACTOR

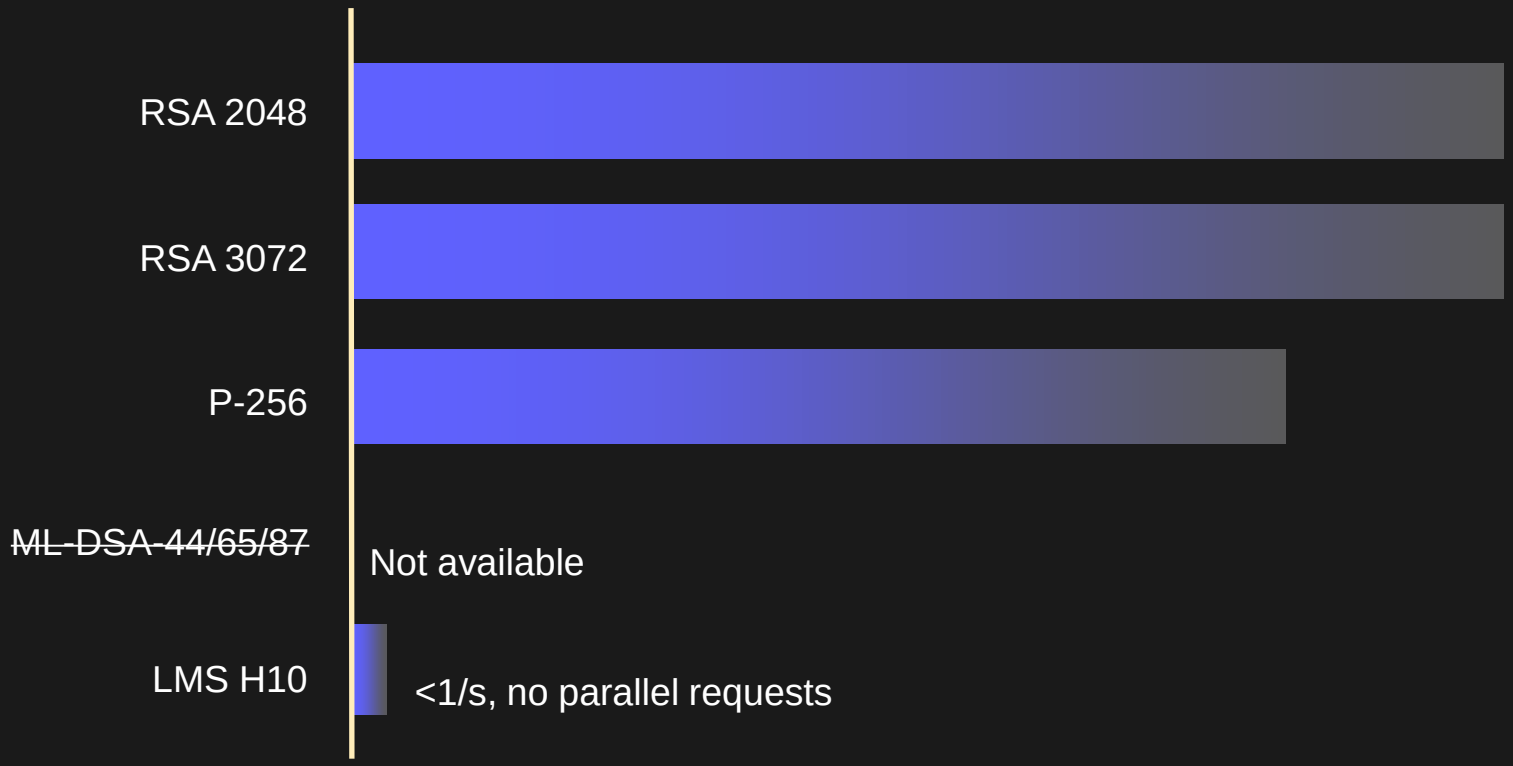
Certificate Issuance HSM 2

REST remote Internet

KEYFACTOR

Certificate Issuance HSM 3

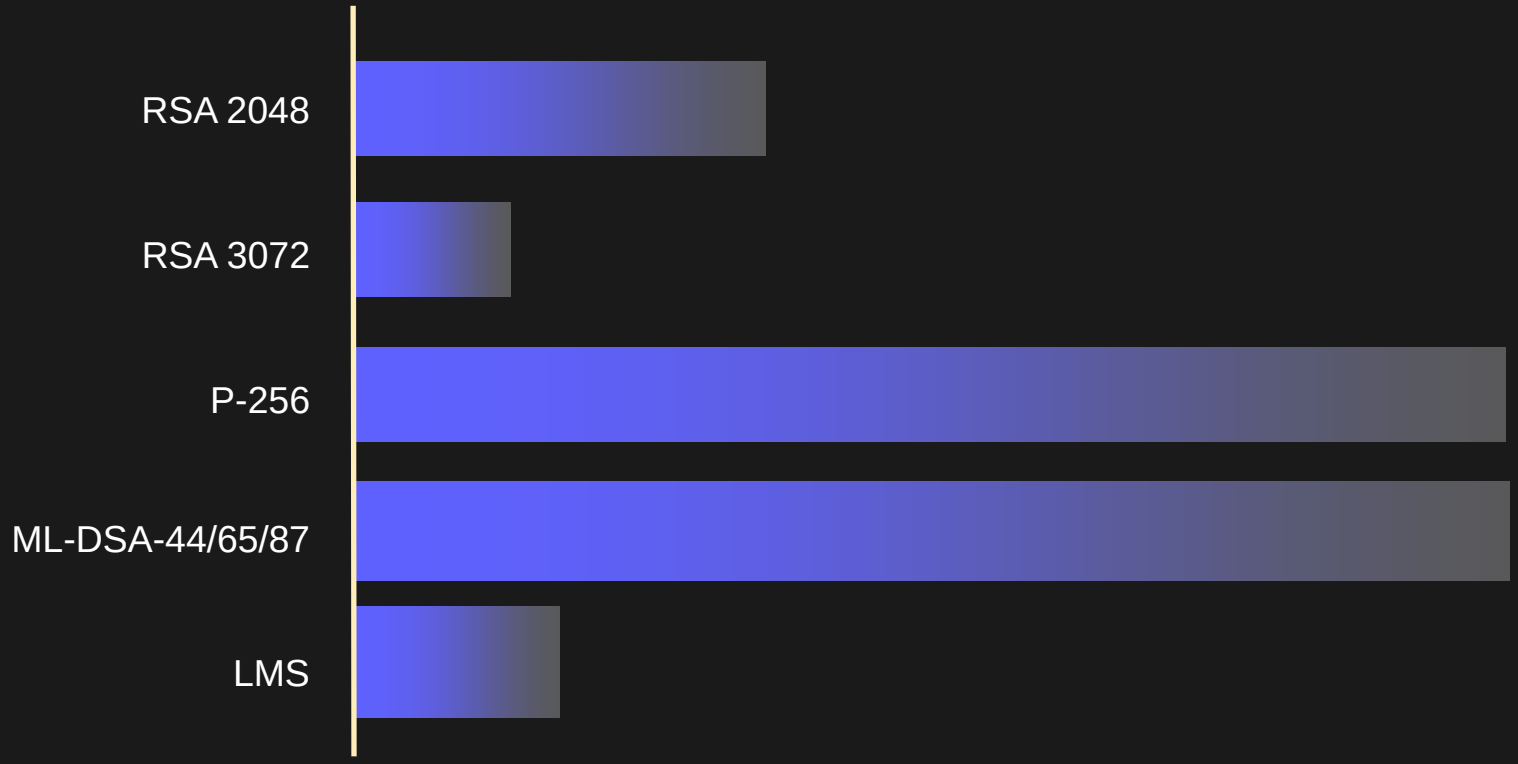
PKCS#11 remote Internet



KEYFACTOR

Certificate Issuance HSM 4

PKCS#11 local container



KEYFACTOR

Certificate Issuance HSM 5

PKCS#11 – local SW

KEYFACTOR

Signing what? How much?

Sizes vary

- Certificates are small
- CRLs are small to large
- Bank transactions are small
- Documents are small to medium
- Firmware is large to huge

There is a limit on the data we can stream to a network connected HSM and expect it to work.

Some popular options currently have 4KB limit of the signature payload.

The new algorithms ~~are~~ were all just-sign (not hash and sign)

KEYFACTOR

CRL size limits

ML-DSA-44 / LMS

KEYFACTOR

Lessons. Learned.

KEYFACTOR

What about PKI and Signing?

FIPS 204: ML-DSA + HashML-DSA

FIPS 205: SLH-DSA + HashSLH-DSA

PKI

- draft-ietf-lamps-dilithium-certificates stipulates ML-DSA and introduces ExternalMu-ML-DSA to cope with CRLs
- In theory, you could have different algorithms for certificates and CRLs
  - Not seen before
- Compare Ed25519 and phEd25519 (hint, it's not fun)

Signing

- CMS have always had a workaround – using signed attributes – signs a hash.
- Easy, always include some signed attributes – but debate is on-going.
- Other signature formats?

KEYFACTOR

FIPS 203, 204, 205

When can we start with ML-DSA and friends?

- Pre standard OIDs
  - 1.3.6.1.4.1.2.267.7.4.4 – Dilithium2 / ML-DSA-IPD-44
  - 1.3.6.1.4.1.2.267.7.6.5 - Dilithium3 / ML-DSA-IPD-65
  - 1.3.6.1.4.1.2.267.7.8.7 - Dilithium5 / ML-DSA-IPD-87

Confused yet?

- Must be changed to Standard OIDs (≠ML-DSA-IPD)
  - 2.16.840.1.101.3.4.3.17 ML-DSA-44
  - 2.16.840.1.101.3.4.3.18 ML-DSA-65
  - 2.16.840.1.101.3.4.3.19 ML-DSA-87
  - + HashML-DSA...2.16.840.1.101.3.4.3.32/33/34

KEYFACTOR

HSMs – PKCS#11 (v3.1) LMS

Same code tested with 2 HSMs so far.

```
final CKA ckaType = getAttribute(session, publicKeyRef, CKA.KEY_TYPE);
    if (CKK.HSS == ckaType.getValueLong()) {
        CKA ckaValue = getAttribute(session, publicKeyRef, CKA.VALUE);
        final HSSPublicKeyParameters params = HSSPublicKeyParameters.getInstance(keyBytes);
        final BCLMSPublicKey pub = new BCLMSPublicKey(params);
    }
```

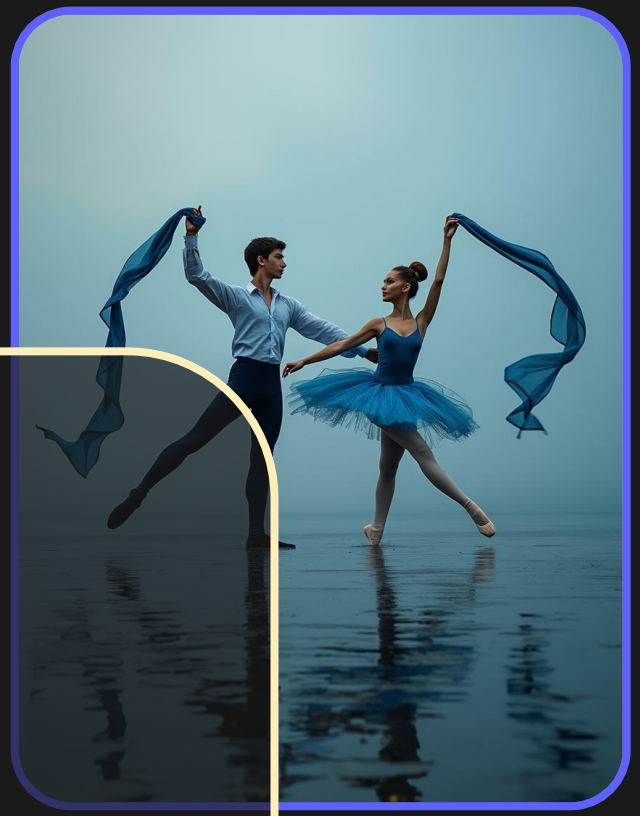
KEYFACTOR

HSMs – PKCS#11 (v3.2 - draft) ML-DSA

```
CKA ckaKeyType = getAttribute(session, publicKeyRef, CKA.KEY_TYPE);
if (ckaKeyType.getValueLong() = CKK.ML_DSA) {
    CKA ckaValue = getAttribute(session, publicKeyRef, CKA.VALUE);
    final byte[] keyBytes = ckaValue.getValue();
    final MLDSAPublicKeyParameters params;
    switch (keyBytes.length) {
    case 1312:
        params = new MLDSAPublicKeyParameters(MLDSAParameters.ml_dsa_44, keyBytes);
        break;
    <snip>
    default:
        throw new InvalidKeySpecException("Invalid length of ML-DSA public key");
    }
    final BCMLDSAPublicKey pub = new BCMLDSAPublicKey(params);
}
```

KEYFACTOR

HSMs – REST



```
Object jsonObjType = jsonKey.get("obj_type");
if (jsonObjType != null && jsonObjType.equals("RSA")) {
    <snip>
} else if (jsonObjType != null && jsonObjType.equals("MLDSA")) {
    Object jsonPublicKey = jsonKey.get("pub_key");
    JSONObject mldsa = (JSONObject)jsonKey.get("mldsa");
    Object parameterString = mldsa.get("param_set");
    if (parameterString == null) {
        log.info("ML-DSA key does not have parameters:" + mldsa);
    } else {
        MldsaParamSet parameterSet = MldsaParamSet.valueOf((String) parameterString);
        return Optional.of(new AsymmetricKey((String) jsonName, (String) jsonKeyId,
                Base64.getDecoder().decode(((String) jsonPublicKey)),
                parameterSet));
    }
}
return Optional.empty();
```

KEYFACTOR

Issues Encountered

LMS/HSS public key encoding

- SubjectPublicKeyInfo vs SubjectPublicKeyInfo-like
- ASN.1 OctetString vs BitString – Internet drafts changed

LMS vs HSS key and signature

- ACVP only have LMS signatures

LMS results are inconsistent

- Slow to very slow, multithreaded or not – not general use

KEYFACTOR

Issues Encountered

ML-DSA vs ML-DSA-IPD – things are not what they seem to be

- Context was added in the last minute, when verification fails you know it 'ipd'

ML-DSA vs HashML-DSA – last minute decision

- ExternalMu-ML-DSA
- SLH-DSA, LMS, …
- Different algorithm for certificates and CRLs?

Signing large data

- Careful testing needed – expect different limits on different HSMs
- How to document this so that users understand?

KEYFACTOR

Where are we now?

- ACVP testing is stable for FIPS standards
- BC 1.79 released with FIPS OIDs and 1.80 with interop fixes
- IETF interop going well – few releases yet
- Some HSMs with FIPS 204 IPD
- Few HSMs with FIPS 204
- Some HSMs with LMS - inconsistent
- Unwise with Production until RFCs
- Start testing early

Removed Dilithium completely – it is not ML-DSA

This is the starting point on the PQC migration journey.

- New algorithms will come in the future.
- Maintaining crypto agility is a must.

KEYFACTOR

Quantum-safe Cryptography, HSMs and Experiences

Tomas Gustavsson

Chief PKI Officer

Keyfactor