PQC Modules: Requirement Specifications, Integration, and Testing

Think openly, build securely

ICMC 2021

September 2, 2021

Dr. Markku-Juhani O. Saarinen

Senior Cryptography Engineer, PQShield Ltd.

NIST Post-Quantum Cryptography

New Public-Key Cryptography Standards

(SP 800-208) Hash-Based Signature: ***LMS / HSS, XMSS / XMSS^{MT}.***

(NIST Finalists) Key Establishment: ***KYBER, NTRU, SABER, McEliece.***

(NIST Finalists) Digital Signature: ***DILITHIUM, FALCON, Rainbow.***

Timeline situation (August 2021):

- ➔ October 2020: NIST SP 800-208 "**Recommendation for Stateful Hash-Based Signature Schemes.**"
- ➔ 2020 NSA: Indicated choosing from the NIST PQC (HBS and Lattice PQC) into CNSA/NSS.
- ➔ 2021 NIST: PQC algorithms chosen for standardization **by the end of the year 2021**.

Outline: Notes for PQC Reqspecs

What can I ask for.. before FIPS 140 & NIAP covers PQC ?

1. **Hash-Based Signatures (HBS) for Firmware Updates.**
2. PQC KEMs: Basic low-level API and Functional Testing.
3. PQC Signatures: Basic low-level API and Functional Testing.
4. Formal Property Verification in PQC engineering.
5. TVLA: “Industry standard” for Side-Channel testing.

NIST SP 800-208 (October 2020)

Stateful Signature Algorithms LMS, HSS, XMSS, XMSS^{MT}

- Refers to RFC 8554 (LMS/HSS) and RFC 8391 (XMSS), some parameter changes.

- Entirely based on SHA-2 or SHA-3 hash functions. Post-quantum secure (~Grover).

- Stateful: Private key supports a limited number (2^{10} ,2^{16}, .. , 2^{40}) of one-time signatures. (*In many implementations the "state" can be just a non-secret signature index 1,2,3.. Then you just need to have a design guarantee that no OTS index is used twice.*)

- Verification is *unlimited*; just needs an approx. 64-byte public key, 2 kB+ signature.

Typical use case: Integrity checking, future-proofed firmware updates.

NIST SP 800-208, LMS SHA-256/192

A "Preferred Choice" for National Security Systems Firmware

NSA Quantum Computing and Post-Quantum Cryptography FAQs in August 2021:

Q: Can I use stateful hash-based signatures?
A: NSA recommends the use of SP 800-208 hash-based signatures, when implemented on properly validated cryptographic modules, to protect NSS in the specialized scenarios outlined in the standard; e.g., for firmware signing. Our preferred parameter set is Section 4.2, LMS with SHA-256/192.

My understanding of the testing situation: Just check against the reference code

- LMS (Leighton-Micali Signature) is not available in FIPS Automated Testing (ACVTS), but SHA-256 of course is. The standard refers to RFC 8554 for algorithm specification.
- RFC 8554 does not discuss the 192-bit truncated version, but has test vectors for LM_SHA256_M32_H5, LMOTS_SHA256_N32_W4, and LMOTS_SHA256_N32_W8.
- SP 800-208, Section 8.1: FIPS 140-3 at Level 3+ requires that an HSM is used for key generation and signature, "No secret key import or export possible!"

LMS and Hash Based Signatures

Current Practical Approach

- If you need more than 2^{25} signatures total, **HSS is just a hierarchical way of using LMS** parameters more than once, so should be fine. HSS also has faster key generation and signing (LMS_SHA256_M24_H25 keygen or first sign is about 2^{37} ops; many hours.)

- Add a "**Winternitz mode**" to a hardware SHA module to do LMS/HSS more efficiently by streamlining padding & iteration (no need to move data to back and forth to CPU).

- **Business continuity and FW Updates:** Consider having mitigating risk controls against Sect. 8.1 physical FIPS requirements*; It may be better to implement layered physical security and document operational procedures for key backups and disaster recovery.*

- (We're aiming to FIPS 140-3 validate LMS/HSS signature verification module only. LMS verification is simple and robust: The control firmware is only few hundred lines.)

Outline: Notes for PQC Reqspecs

What can I ask for.. before FIPS 140 & NIAP covers PQC ?

1. Hash-Based Signatures (HBS) for Firmware Updates.
2. **PQC KEMs: Basic low-level API and Functional Testing.**
3. PQC Signatures: Basic low-level API and Functional Testing.
4. Formal Property Verification in PQC engineering.
5. TVLA: “Industry standard” for Side-Channel testing.

Lattices: Random Samplers

Decryption or Sign. verify testing won't catch these bugs

- **A random sampler** picks a random number *from given distribution*.

  Testing options: χ^2 test statistics or similar, or a fully defined, deterministic sampler.

- Uniform distribution $0 \leq x < 2^n$ is easiest. Binomial (Hamming Weight) and other direct mappings from fixed n bits are almost as easy. **SABER**, **NTRU** work with just these.

- **Dilithium** and **Kyber** also need uniform random $0 \leq x < q$, where q is a small prime. This is done with *rejection sampler* that picks an uniform n-bit x's ($q < 2^n$) until $x < q$. A variable number of x's are required, but the method is still, usually *leakage-free*.

- **Falcon** signature requires random numbers from the *Discrete Gaussian distribution*. Designers define a deterministic sampling method, which relies on IEEE 754 doubles.

PQC KEM Low-Level Interface

KEMs are KEMs - Hope you can make them *deterministic*

- (CCA2) PQC KEMs can be used to for public-key encryption and decryption (e.g. by pairing them with AEADs like AES-GCM), but this is not their natural testing interface.

- PQC KEMs can also be used for ephemeral key exchange, but do not have the commutativity of Diffie-Hellman. KEMs natively use a keygen/encaps/decaps API.

Keypair generation: Initialization.	(PK, SK) ← **KeyGen**(Seed_{KG})
Encapsulation: Public key operation.	(CT, SS) ← **Encaps**(PK, Seed_{ENC})
Decapsulation: Private key operation.	SS ← **Decaps**(CT, SK)

PK = Public Key, **SK** = Secret Key, **CT** = Ciphertext, **SS** = Shared Secret, **Seed** = Random.

PQC KEM With Random Seeds

Hoping to retain KeyGen/Encaps KAT Determinism

Finalist PQC KEM	**$Seed_{KG}$**	**$Seed_{ENC}$**
KYBER (all variants)	64	32
SABER (all variants)	96	32
Classic McEliece	32	(Large)
NTRU-hrss701	1432	1400
NTRU-hps2048509	2445	2413
NTRU-hrss1373	2776	2744
NTRU-hps2048677	3243	3211
NTRU-hps4096821	3927	3895
NTRU-hps40961229	5865	5833

- KYBER and SABER only take 32-96 bytes from the RBG and initialize a (SHAKE) XOF with this. (Originally specified for performance reasons.)

- With **$Seed_{KG}$** and **$Seed_{ENC}$** inputs these KEMs are fully deterministic. KATs (Known Answer Tests) can be used *without a dummy RNG*.

- Not all algorithms have this, but is easy to add.

We hope NIST retains such simple KAT Testability!
(vs. the pain of *e.g.* validating RSA key gen now..)

PQC KEM Functional Testing

Current Practical Approach

- **Run binary KATs:** PQC design teams have specified fairly efficient and secure, de facto serialization methods for public keys, secret keys, and ciphertexts. **Each submission comes with a set of KeyGen and Encaps KATs that use those**. We use them to test our hardware modules against public optimized and reference implementations.

- **Add coverage:** We have **added KAT tests for invalid, corrupted, and mismatching public keys and ciphertexts**. PQC KEM Decapsulation should fail in an "implicit" manner with a specific SS' ≠ SS result (no failure oracle). This must be tested.

- **Be smart and avoid ASN.1 (beyond algorithm OIDs.)** The designer's bit encodings can be improved, but not much! ASN.1 or other "abstract" low-level encodings can just make things worse; they'll likely introduce complex input validation and security problems (e.g. timing attack vectors.) Modern Elliptic Curve Crypto avoids them too.

Outline: Notes for PQC Reqspecs

What can I ask for.. before FIPS 140 & NIAP covers PQC ?

1. Hash-Based Signatures (HBS) for Firmware Updates.
2. PQC KEMs: Basic low-level API and Functional Testing.
3. **PQC Signatures: Basic low-level API and Functional Testing.**
4. Formal Property Verification in PQC engineering.
5. TVLA: “Industry standard” for Side-Channel testing.

PQC Signature Low-Level Interface

Message (not hash) padding is usually a part of the algorithm

- PQC Signature algorithms generally do not support the old "*hash-and-sign*" mode.

- The algorithms perform message pre-padding; this eases requirements on collision resistance and hash lengths. (A modern feature: Also XMSS & LMS/HSS, EdDSA.)

- The NIST "envelope" sign/open API is not super practical but can be used for KAT tests.

$(PK, SK) \leftarrow$ **KeyGen**($Seed_{KG}$)

$S \leftarrow$ **Signature**(M, SK, $Seed_{SIGN}$)

Ok / Fail $\leftarrow$ **Verify(** S, M, PK)

Concatenated "Envelope" KATs:

$SM \leftarrow$ **Sign**(M, SK, $Seed_{SIGN}$)

M / Fail $\leftarrow$ **Open**(SM, PK)

PK = Public Key, **SK** = Secret Key, **M** = Message, **S** = Signature, **SM** = Signed Message.

PQC Signature Functional Testing

My wish: Deterministic Key Generation and Signatures

- **Dilithium** uses an internal XOF in a similar fashion as Lattice KEMs. Current version 3.1 can be made deterministic with $Seed_{KG}$ = 32 bytes and $Seed_{SIGN}$ = 64 (or 0 random bytes as one can also use the message itself - and the secret key - to derive $Seed_{SIGN}$.)

- For modules we retain a compatible mode and hence can KAT test entire Dilithium KeyGen() and Signature() functions using the seeds - in a similar fashion as PQC KEMs.

- **Falcon** and **Rainbow** have somewhat under-specified internal seed expanders (that would have to be modified for NIST standardization), but could use the same principle.

- PQC Sign APIs are more like that of EdDSA than ECDSA. Also, specifying bit-level serialization is best done by the algorithm design teams rather than PKI integrators.

Outline: Notes for PQC Reqspecs

What can I ask for.. before FIPS 140 & NIAP covers PQC ?

1. Hash-Based Signatures (HBS) for Firmware Updates.
2. PQC KEMs: Basic low-level API and Functional Testing.
3. PQC Signatures: Basic low-level API and Functional Testing.
4. **Formal Property Verification in PQC engineering.**
5. TVLA: “Industry standard” for Side-Channel testing.

Formal Verification: Ask About It

Your PQC hardware vendor probably has formal models

- Formal verification is completely mainstream in the semiconductor industry, and tools are mature. It is just more effective than randomized testbenches. *Ask the vendor.*

- We mainly use SystemVerilog formal assertions & Bounded Model Checking (BMC). The tools can prove the assertions (or model equivalence) logically with a SAT solver.

- Can also cover Hardware/Software codesign (e.g. embedded C language with CBMC).

- Most of Dilithium, Kyber, Saber specification can be handled by modern formal tools. Creating & checking models for components such as (ring element) Rounding, Montgomery Reduction, Sampling, etc, is exactly what a *verification engineer* does.

To me it seems that the semiconductor industry is ahead of cryptographers in formal!

Outline: Notes for PQC Reqspecs

What can I ask for.. before FIPS 140 & NIAP covers PQC ?

1. Hash-Based Signatures (HBS) for Firmware Updates.
2. PQC KEMs: Basic low-level API and Functional Testing.
3. PQC Signatures: Basic low-level API and Functional Testing.
4. Formal Property Verification in PQC engineering.
5. **TVLA: “Industry standard” for Side-Channel testing.**

Side Channel (Timing, DPA) Testing

In the absence of approved tests in SP 800-140F

Situation as I understand it:

FIPS 140-3 Levels 3 and 4 "**Shall** be tested to meet the approved **non-invasive attack mitigation** test metrics."

However.. NIST SP 800-140F is still empty (As are non-invasive test metric sections in ISO/IEC 24759 §6.18 and ISO/IEC 19790 Annex F.)

People are using: **ISO/IEC 17825:2016** (Side-channel terminology, Welch t-test / TVLA procedure), **ISO/IEC 20085-1:2019** (Test tools), and **ISO/IEC 20085-2:2020** (test calibration methods and apparatus).

Side Channels: TVLA and ISO/IEC 17825

Not perfect - but can be specified for PQC Side-Channel Tests

Common "non-specific" t-test:

Get Power/Emission traces from Signatures or Decryptions with random input and:

Set **A**: Constant Secret Key.
Set **B**: Varying (random) Secret Keys.

Compare pointwise distributions in Set A to Set B with Welch t-test to detect leakage.

$$T = \frac{\mu_A - \mu_B}{\sqrt{\frac{\sigma_A^2}{N_A} + \frac{\sigma_B^2}{N_B}}}$$

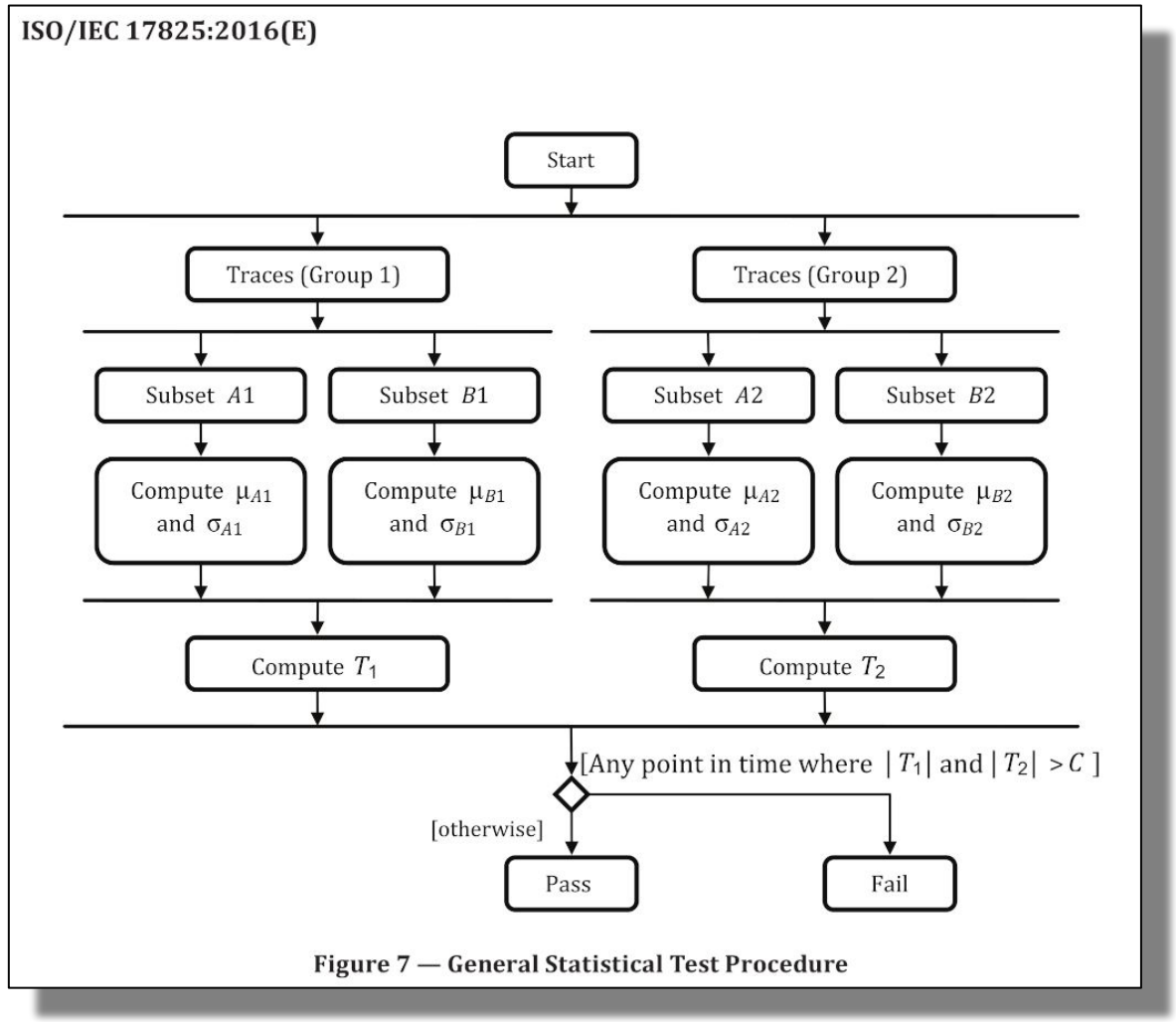


Figure 7 — General Statistical Test Procedure

Side Channels in PQC

Main things to verify in PQC Signatures and KEMs

- **PQC Signatures** are used for authentication similarly to ECDSA. Observation of repeated signatures must not help forgery.

- "**KEMTLS**" is likely to be adopted for authentication so CCA KEM Decapsulation is used with static keys, which must not leak. https://www.ietf.org/id/draft-celi-wiggers-tls-authkem-00.html

- Also for CCA KEMs, **Decapsulation failure oracles** (malformed or mismatched ciphertexts) must not be detectable via side channels.

  (Payload-dependent latency seems unlikely as signatures always use hashes and KEMs do not deal with plaintext at all. Key generation: one trace?)

During module development, check **all components that "touch" SSPs**.

Side Channels: PQC Timing Attacks

Secure implementations are available: Trust but Verify

- Most PQC Finalists have implementations that are resistant to timing attacks, assuming that certain CPU instructions have *data-independent latency*.

- Verification: On RISC-V, the new **Zkt extension** defines that set. We use a special ISA simulator to verify that SSP passes only through safe instructions in compiled code.

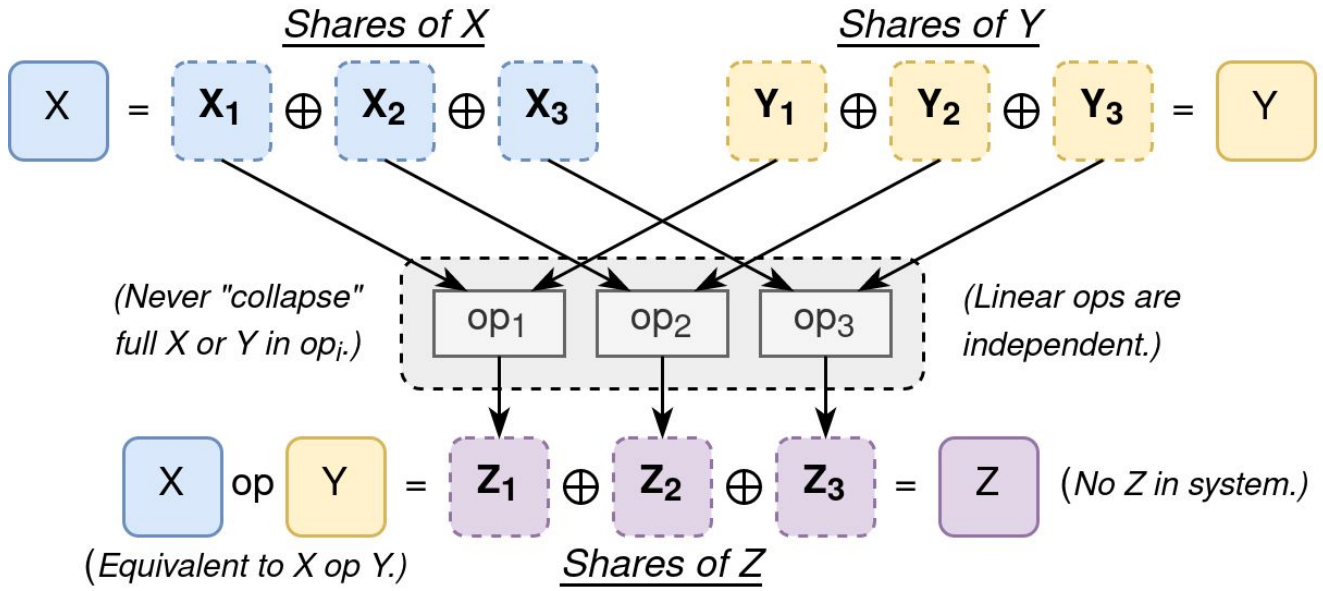
- Outside RISC-V one can use tools such as the memory sanitizer to do similar checks; https://www.amongbytes.com/post/20210709-testing-constant-time/

- The term “constant-time” should not be taken literally. The algorithms have variable timing. It is sufficient that the timing does not correlate with SSPs (e.g. secret keys).

- **TVLA test**: “non-specific t-test” on time (e.g. cycle counts), fixed keys vs random keys.

Side Channels: PQC and DPA, Emissions

Common mitigation: Masking & Threshold Implementations



Masking requires special, often PQC algorithm-specific implementation techniques.

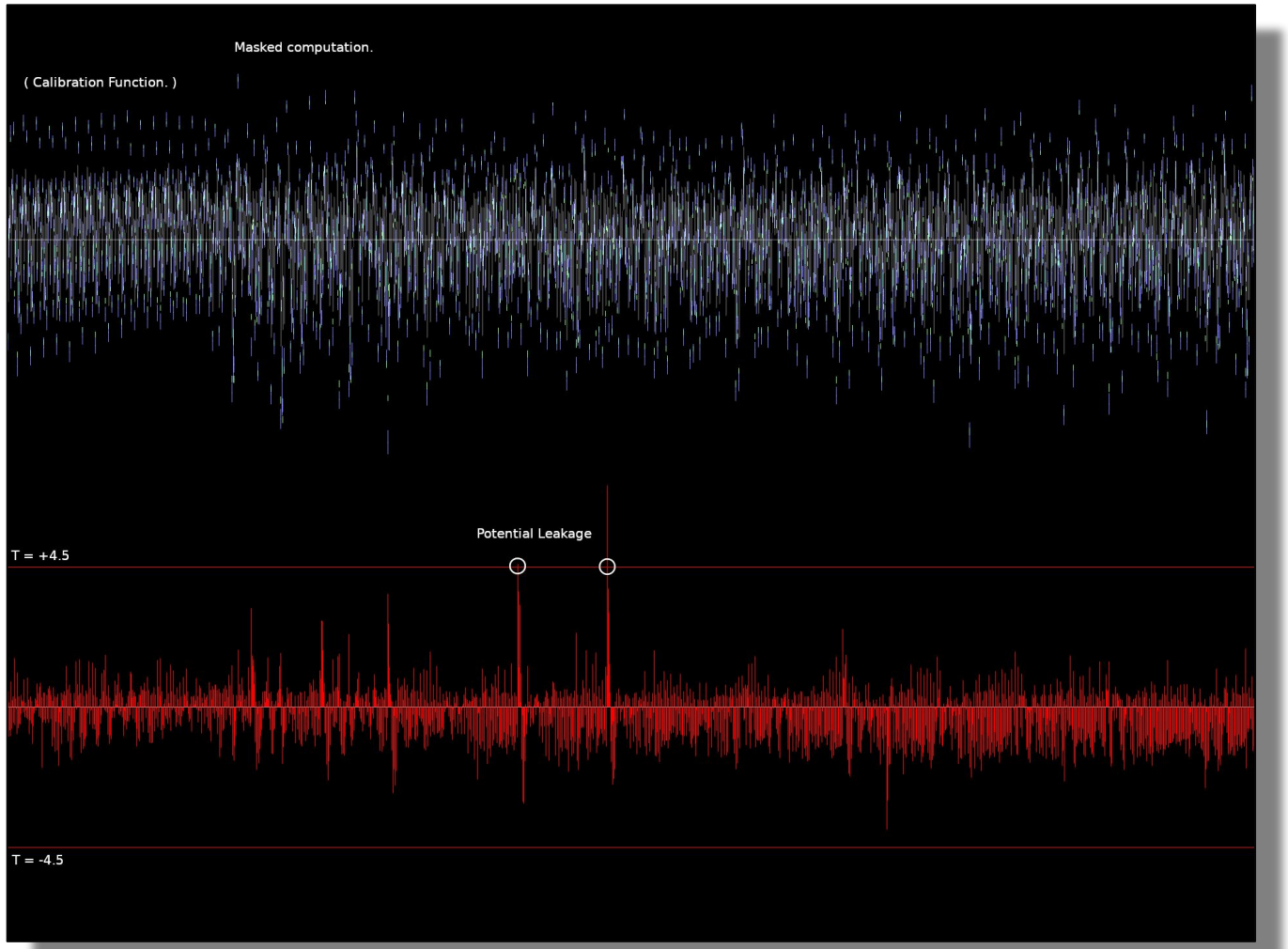
***Note**: XOFs and Seeds can be masked too (SHA3 is much simpler to mask than SHA2.)*

- **Masking:** Computation on secrets is performed on randomized shares. **But!** Claim of a "masked implementation" alone does not guarantee even basic TVLA/17825 security.

- Masked hardware modules will offer non-invasive attack mitigation -- at least for most PQC Lattice Schemes. Software masking can also be done, but is not very portable.

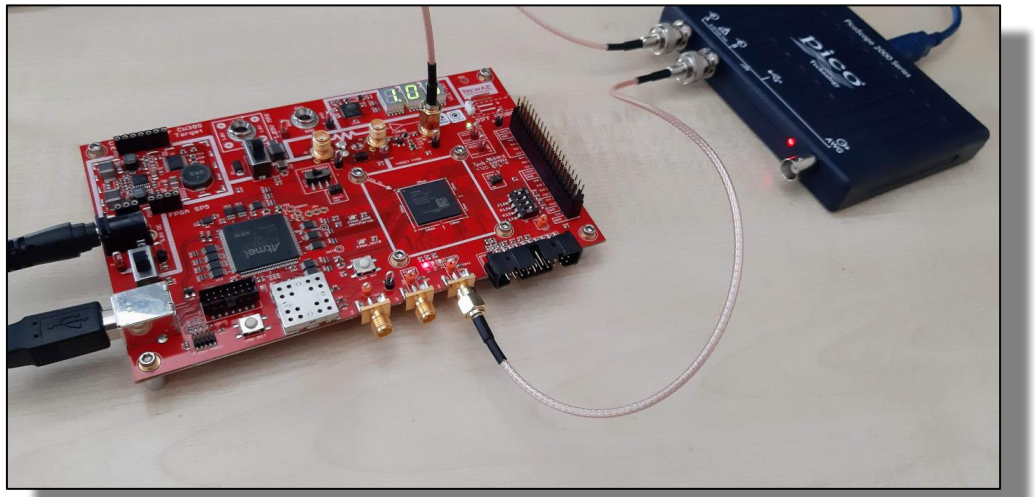
Side Channels: FPGA Leakage Emulation

ISO/IEC 20085 - Can be specified for PQC Side-Channel Tests



👈 We use FPGA to emulate leakage during HW module development. Helpful in finding “problem cycles.”

👇 CW305 *“artefact”* as discussed in Annex C of ISO/IEC 20085-2:2020(E).



Thank You!

Summary

- **Known Answer Testing of PQC** (keygen/encaps/decaps & keygen/sign/verify) is possible with many implementations, including hardware and even masked!
- .. **but need more coverage** for KEM decapsulation failures and other special cases.
- Testing hooks & APIs are little different than for pre-quantum asymmetric crypto.
- We hope NIST specifications will describe bit serialization and XOF “seed expanders” that currently makes these functions internally deterministic and high-level testable.
- At least individual low-level components are likely to be covered by formal tests.
- **TVLA & ISO 17825** has emerged as way to do basic side-channel testing (Timing, DPA, Emissions) of PQC modules. **Masking** is a robust, testable mitigation technique.