Side-Channel Leakage tests for Post-Quantum Modules

18th CryptArchi Workshop

May 30, 2022 - Porquerolles

Dr. Markku-Juhani O. Saarinen

Senior Cryptography Architect, PQShield LTD

Motivation: NIST PQC Means FIPS 140-3 PQC

Post-Quantum Crypto transition is driven by NIST/FIPS

NIST/FIPS Post-Quantum Crypto: Selection 2022, Standards 2024.

NIST Selection: *"Any Day Now."* (Has been since the start of 2022 🤷)

Replaces ECC, RSA, ECDSA key establishment and signatures in many applications.

Especially for U.S. Government Entities:

- Relatively fast transition expected (presidential directives NSM-08, NSM-10).
- Regulations mandate FIPS 140-3 cryptography -> including PQC modules.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Motivation: FIPS 140-3 Non-Invasive Security

Also Known as Side-Channel Testing

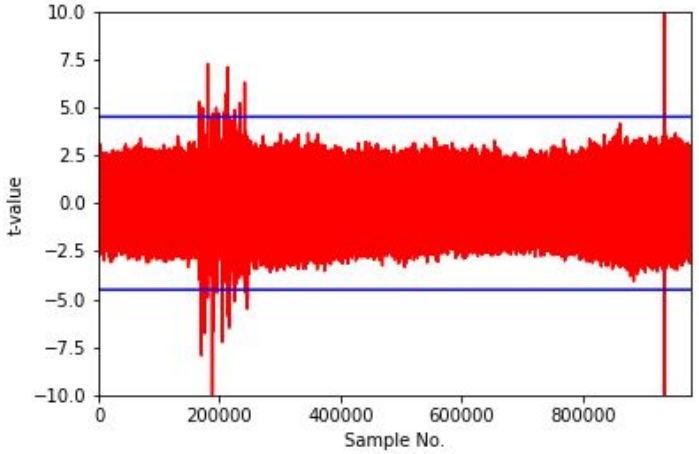
Introduced as a major change in FIPS 140-3 in relation to FIPS 140-2:

- Side-Channel Attacks (Power, Emissions, Timing) are in 140-3 scope!

- Documentation required for Levels 1 and 2. Mitigation Testing at Levels 3 and 4.

But how?

- Initially not tested (only “if claimed by a vendor”.)
- Annex F of ISO 19790:2012 defined no test metrics.
- Annex F of ISO 19790:2022 will have ISO 17825.



“Non-Invasive” in FIPS 140-3 Specifications

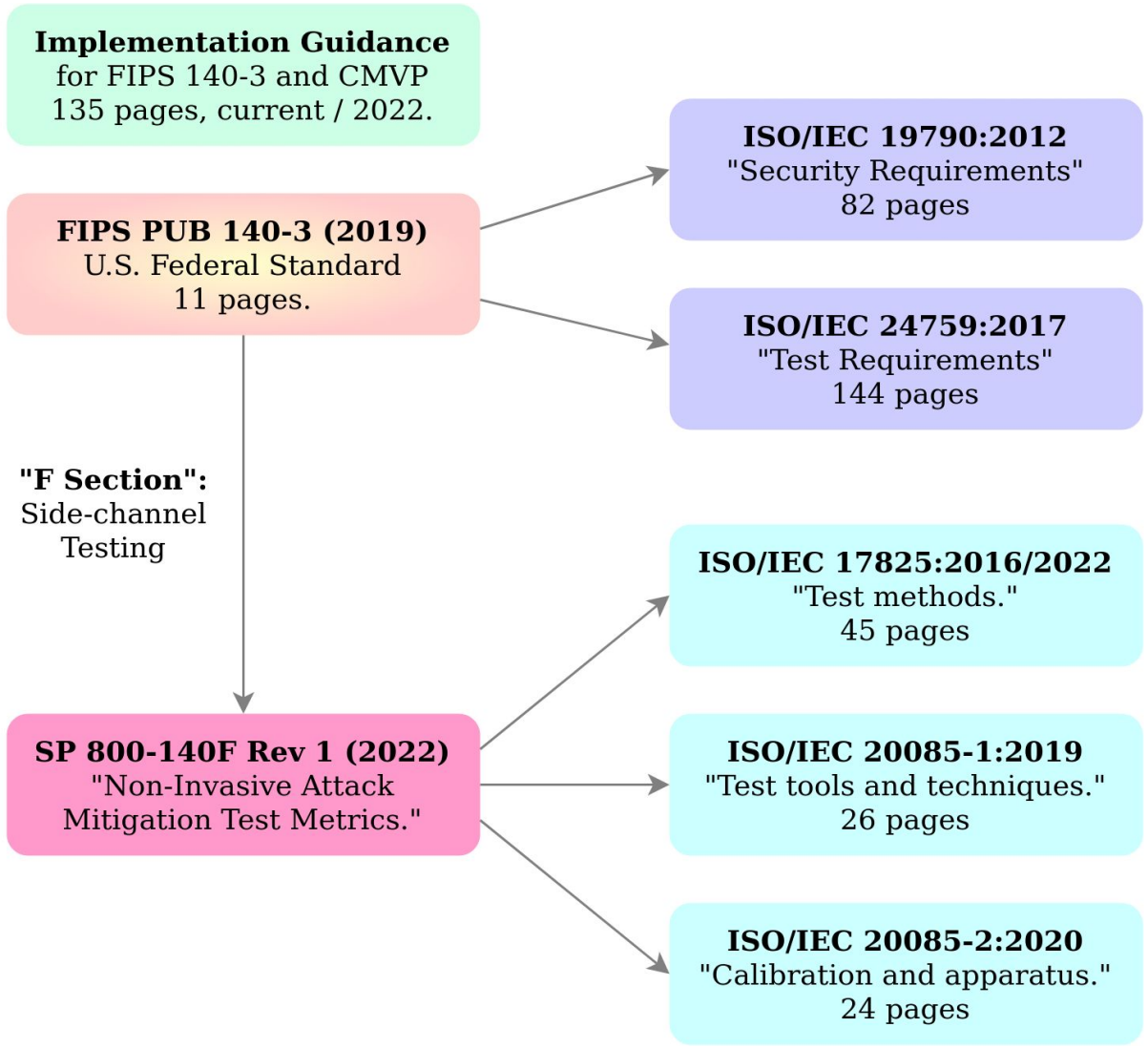
NIST Special Publication		ISO/IEC 19790:2012(E)	ISO/IEC 24759:2017(E)
SP 800-140	modifies	--	§6.1 through §6.12
SP 800-140A		Annex A	§6.13
SP 800-140B		Annex B	§6.14
SP 800-140C		Annex C	§6.15
SP 800-140D		Annex D	§6.16
SP 800-140E		Annex E	§6.17
SP 800-140F		Annex F	§6.18

NIST SP 800-140F Rev. 1 (Draft)

CMVP Approved Non-Invasive Attack Mitigation Test Metrics

Document Revisions

Edition	Date	Change
Revision 1	[date]	**§ 6.2 Approved non-invasive attack mitigation test metrics** Added: ISO/IEC 17825 and associated ISO/IEC 20085-1 and -2



Applying the Standards to PQC

"Testing methods for the mitigation of non-invasive attack classes"

ISO/IEC WD 17825:2021(E) is based on Test Vector Leakage Assessment ("TVLA.")
It does not try measure the difficulty of attack (like CC AVA_VAN); just detect leakage.

The standard text starts out with:

The test approach employed in this International Standard is an efficient "push-button" approach: the tests are technically sound, repeatable and have moderate costs. [!]

Reality:

- That's for testing labs ~2025. A well-defined "push-button" does really exist yet.
- But vendors already need to be able to assure customers about side-channel security.

ISO 17825 Leakage Analysis Scenario

DPA and DEMA: Power and Electromagnetic Emission Traces

- **Standard attack setting**: Tester can set inputs to the module at the IO boundary (API). Can choose inputs and synchronize to the start of the operation.

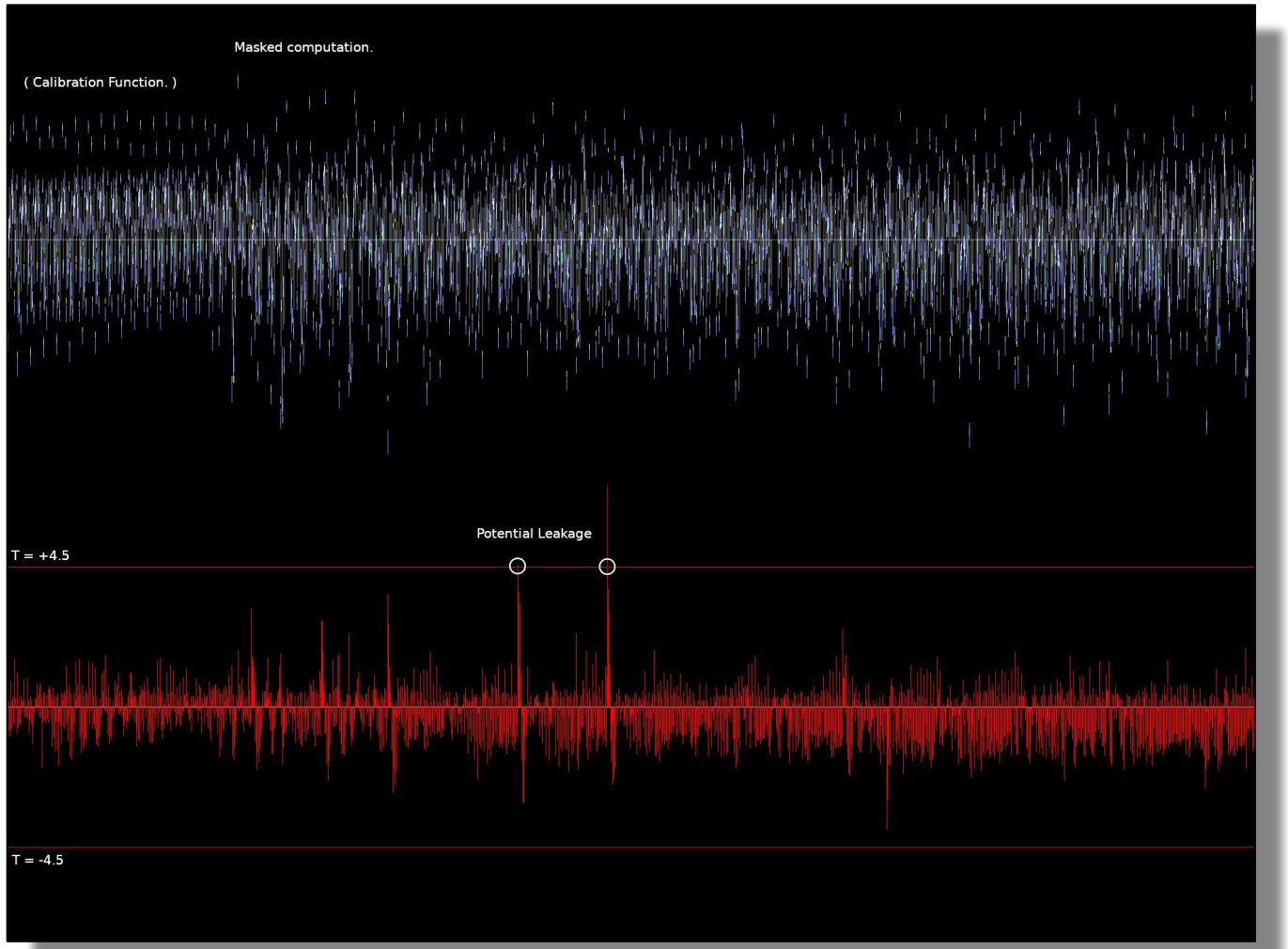
- **Oscilloscope measures power** (or electromagnetic emissions) at high precision, perhaps a couple samples per clock cycle. Measurement vectors are “traces”.

- **Traces are analyzed** to detect leakage. In leakage analysis the analyst can know or choose keys: Is looking for correlations between keys and and the traces.

- **Statistical analysis of significance.** PASS/FAIL metric (no key recovery).

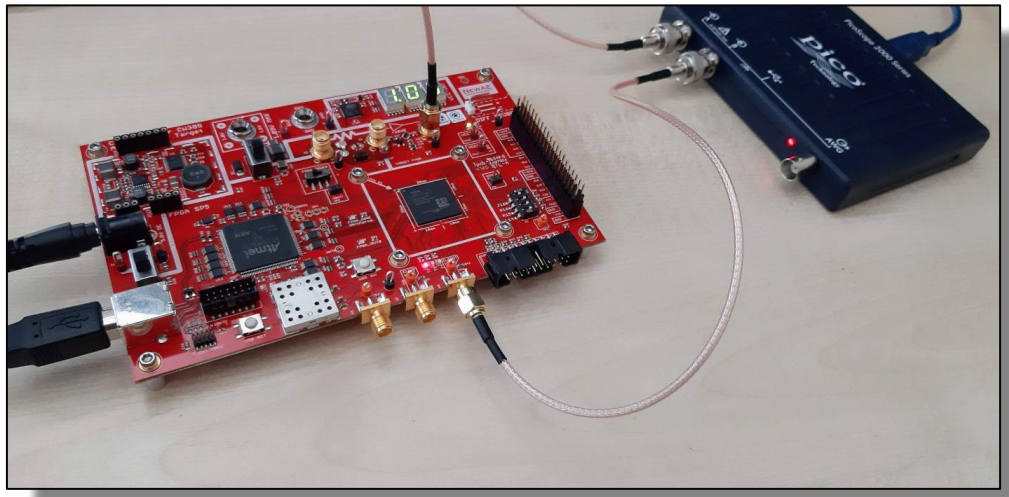
Side Channels: FPGA Leakage Emulation

ISO/IEC 17825 & 20085 - PQC Side-Channel Tests



👈 We use FPGA to emulate leakage of hardware post-quantum crypto modules. Try to apply ISO 17825.

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



What are the “non-invasive mitigations” like?

Expect masking + ad hoc countermeasures

- Masking splits secrets into “shares.” Successful measurement of an individual share does not leak the secret itself. “Masking Gadgets” used to perform arithmetic steps.

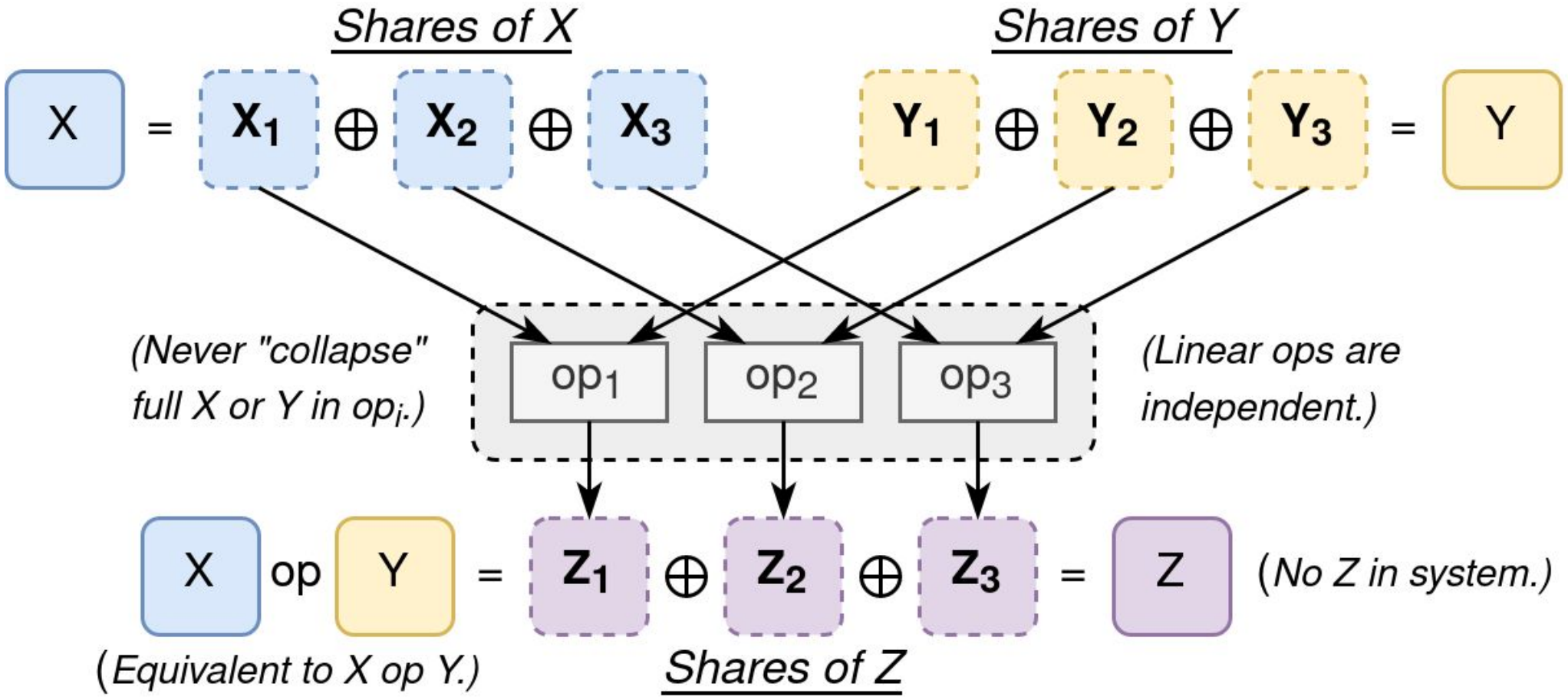
Type:	Relationship:	Algebraic object:
A/Q:	$X = X0 + X1 \pmod{q}$	Prime q is 3329 (Kyber) or 8380417 (Dilithium).
A/N:	$X = X0 + X1 \pmod{2^N}$	Size N is 16 or smaller (NTRU 11..14, Saber 14).
B:	$X = X0 \oplus X1$	Bit strings (managed e.g. as 64-bit words).

- **Most cryptographers agree**: Masking and other attack mitigation techniques for PQC algorithms are technically more complex than for older cryptography.

- **Why?** The algorithms are not homogenous like RSA or ECC but contain a number of dissimilar steps. One may have to design a dozen different gadgets for one algorithm.

The main countermeasure: Masking

Limit leakage by breaking computation into randomized shares



First: Classify Internal and External Variables

Designer classifies all variables and wires into CSPs and PSPs

Critical Security Parameter (CSP) requires require both confidentiality (secrecy) and integrity (no modification) protection.

Examples: Secret and private keys, passwords, temporary tokens, and derived temporary quantities whose disclosure would compromise the security of the cryptographic system.

Public Security Parameters (PSPs) do not need confidentiality but need Integrity.

Examples: A public key needs to be handled in a way that prevents it from being changed or replaced. A digital signature or ciphertext is usually a PSP, not a CSP. Any component variable of a secret key that can be easily derived from the public key is a PSP.

Sensitive Security Parameters (SSPs): Together, CSPs and PSPs constitute SSPs.

Examples: Most inputs and outputs of a cryptographic module are SSPs. A public-private key pair is an SSP. FIPS Zeroization requirements apply to all SSPs, including PSPs.)

What needs to be protected?

Only CSPs are in Scope of non-invasive (and need masking)

Section 7.8 of FIPS 140-3 and ISO/IEC 19790:2012(E) defines non-invasive attacks:

``*Non-invasive attacks attempt to compromise a cryptographic module by acquiring knowledge of the module's CSPs without physically modifying or invading the module. Modules may implement various techniques to mitigate against these types of attacks.*''

- Only leakage of CSPs is relevant for FIPS 140-3. Public key leakage is a *false positive*.

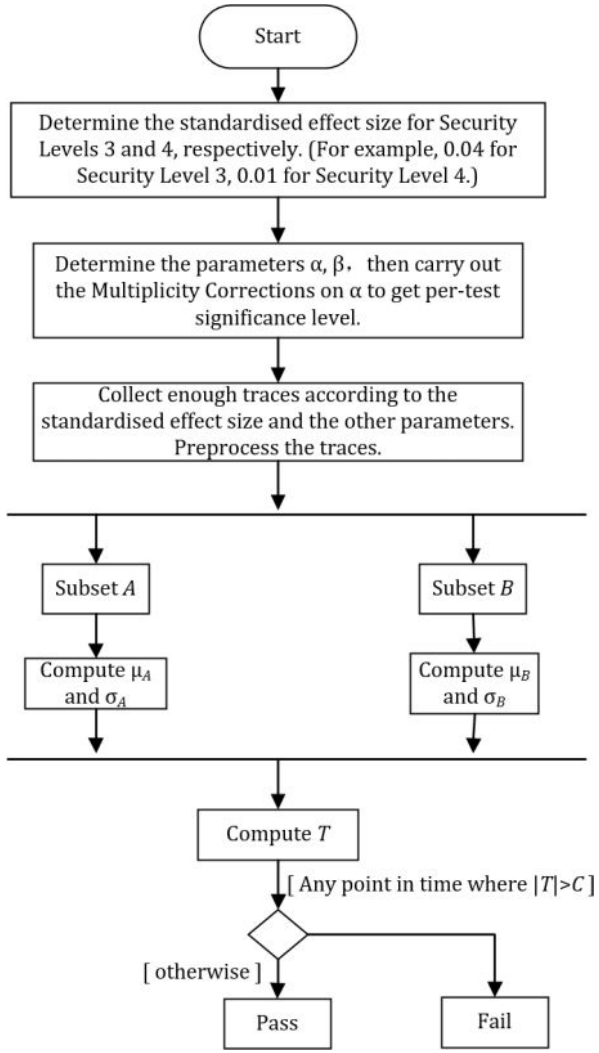
- For us, this CSP is primarily information that (A) can be used to determine a shared secret in a key establishment scheme or (B) forge a signature in a signature scheme.

- Invasive physical attacks (that modify the state) seem to be out of scope for ISO 17825 and FIPS 140-3; e.g. fault attacks. These are a part of CC vulnerability assessments.

Basic SCA Tests for Post-Quantum Crypto

Detects “leakage” – no key recovery (easily False Positives)

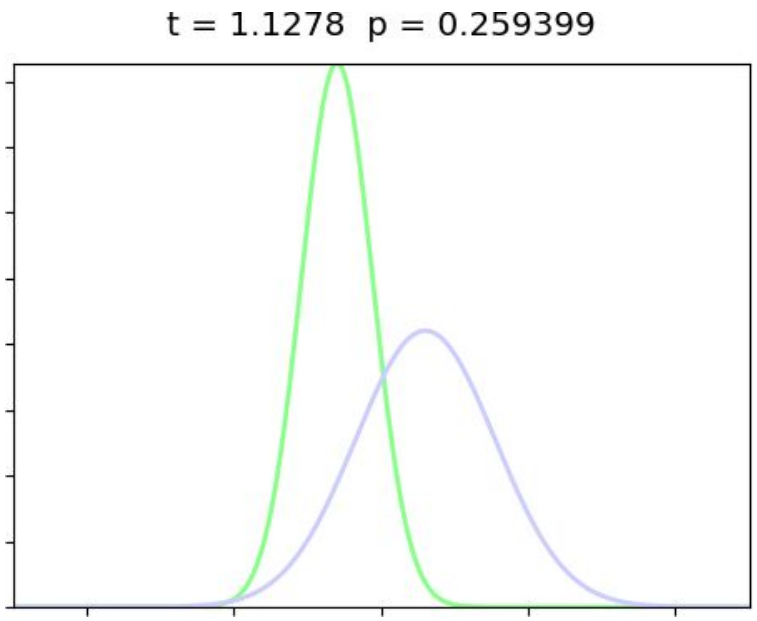
- ISO 17825 has a “general statistical test procedure.”
- The current version of these tests create data subsets A and B of measurements (e.g., trace waveforms) with the IUT.
- But the trace sets A and B need input test vectors!
- **Example**: Set A may use a fixed bit value in a CSP, while measurements in set B use random CSP values.
- If the A/B measurement sets can be distinguished from each other – with the Welch t-test with high enough statistical confidence – this is taken as evidence of CSP leakage.

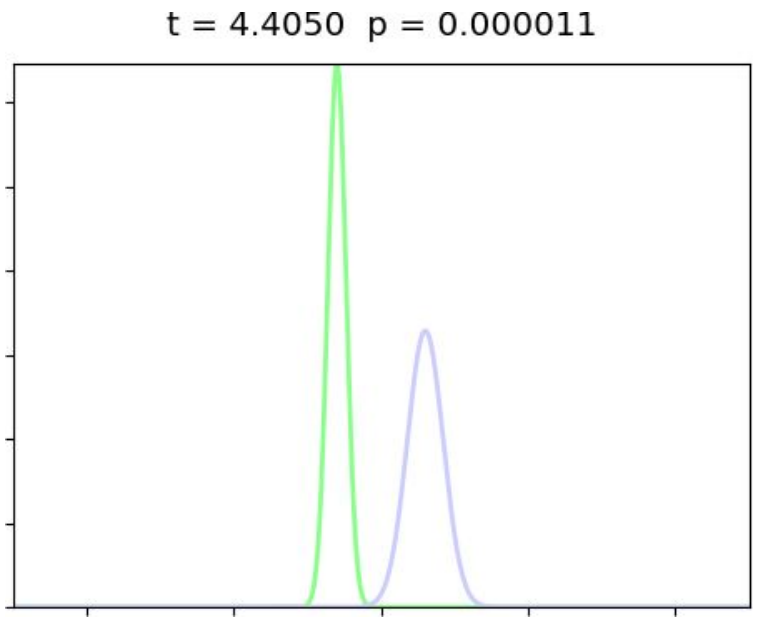


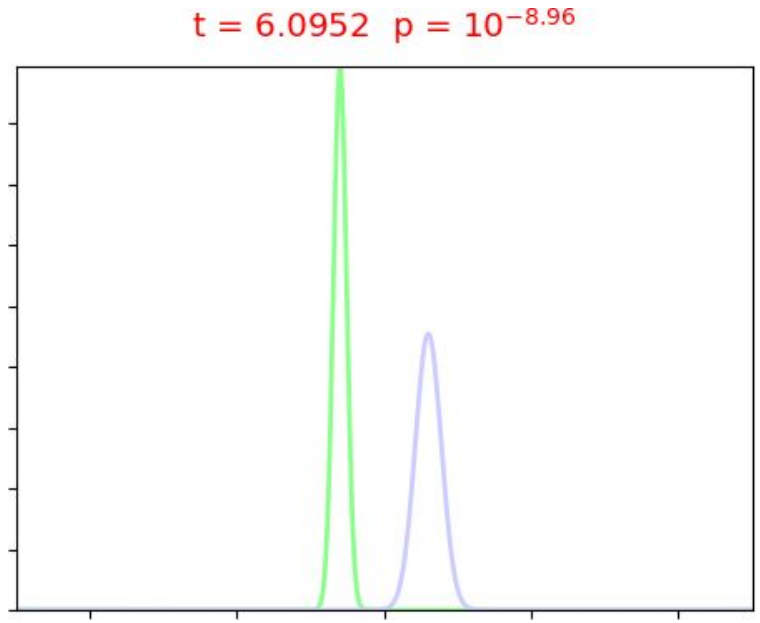
Simple math: Non-specific (Welch) t-test

Leakage is assumed when A and B don't have the same mean

- Subsets A and B are trigger - synchronized. Has cycle precision – some nanoseconds.
- For each time sample, compute averages (μ_A, μ_B) and standard deviations (σ_A, σ_B).
- t-statistic relates to the certainty that the two sets are distinguishable.
- Confidence "probability" assumes Gaussians distribution (here normalized by $1/\sqrt{N}$).







ISO 17825 “General Statistical Test Procedure”

Outline of the General Statistical Test Procedure

0. Determine the required sample size $N = N_A + N_B$ and t-test threshold C from the experiment parameters.
1. Collect Subsets A and B and compute their pointwise averages (μ_A, μ_B) and standard deviations (σ_A, σ_B).
2. Compute the pointwise Welch t-test statistic vector

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

3. If at any point $|T| > C$, the test results in a FAIL. If the threshold was is not crossed, the test is a PASS.

External API Interfaces for SCA testing

Using handles: Testing just the core private key operation

Tester: Create inputs (load test vectors or compute them).

↓

key_handle = **key_import():** Deserializes CSPs into module's internal memory layout.

↓

——————— *Trigger Measurement Start.* ———————

ss_handle = **decaps(** ct**,** key_handle **)**

——————— *Trigger Measurement End.* ———————

↓

ss_tv = **key_export(** ss_handle **):** Collapse shares and extract results from memory.

↓

Tester: Verify results, store measurement.

Also test secure CSP import and export

ISO 17825 Requires testing at “Module I/O Boundary.”

Using secure import (and export for keygen, encaps, decaps)

Tester: Create inputs (load test vectors or compute them).

↓

——————— *Trigger Measurement Start.* ———————

signed = **sign(** wrapped_key, msg **):**

handle = **key_import(** wrapped_key **)**: Deserialize CSPs.
signed = **sign(** handle, msg **):** Private key operation.

——————— *Trigger Measurement End.* ———————

↓

Tester: Verify results, store measurement.

Goals of Automatic TVLA "Sign-Off"

Leakage tests should aim for widest possible coverage

1. **Try to have specific testing coverage over all CSPs in all relevant sub-algorithms.**

   (Key Generation, Key Export, Key Import, Encapsulation, Decapsulation, Signature.)

2. **Design the experiments and test vectors (input data) in a way that eliminates false positives to greatest extent possible.**

   (Hopefully no need to specify "areas of interest" in resulting traces.)

Industry will need to agree on a standardized set of test vectors in order to have consistent results. These are dependant on details of each algorithm.

Two basic types of test vectors will get you far

Fixed vs Random (FIX) and A/B Classification (ABC)

1. **Fixed vs Random** (non-specific t-test) can be used in “live” testing:
   - Trace set A: Fixed CSP for every trace.
   - Trace set B: New random CSP secret for each trace.

2. **A/B Categorization** works with capture-then-analyze flow:
   - Records traces with detailed test vector metadata; CSPs are known in analysis.
   - Traces are categorized *after capture* to A and B sets based on CSP selection criteria, Examples: a specific internal CSP variable or secret key bit, “plaintext checking” bit.
   - The same trace data can be categorized to A and B in a number of different ways.

In both cases: Set A and Set B statistically differentiable with t-test = **FAIL**.

Example 1: SABER CPA Private Key Operation

Algorithm CPA.Decrypt: "SABER.PKE.Dec"

CPA.Decrypt(CT, SK):

0. $s = SK$ // *Load and remask the secret key.*
   $(c_m, b') = CT$ // *Deserialize the ciphertext.*
1. $v \leftarrow b'^T * s \pmod p$ // *Ciphertext b with private key s.*
2. $M' \leftarrow \text{round}(v - (p/T) \cdot c_m + h_2, p/2)$ // *Extract the m.s.b. from share.*
3. return M'

- Secret key masks must be refreshed before each private key operation.
- CPA.SD: Subset **A** has fixed (**CT**, **SK**) while subset **B** has all fixed CT, random **SK**.

- That was easy! .. however SABER.PKE.Dec is *not* the actual interface of SABER ..

Actual Saber (CCA KEM) Private Key

More complex because of Fujisaki-Okamoto Transform

Recall that ISO 17825 tests are done at the “Module I/O” boundary, i.e. with CCA:

CCA: PublicKey_cca = (seed_A ‖ Encode(b_p))
CipherText_cca = (Encode(c_m) ‖ Encode(b’))
SecretKey_cca = (**z** ‖ hash_pk ‖ seed_A ‖ Encode(b_p) ‖ Encode(**s**))

- The CCA secret key contains the public key too – because of re-encryption!

- If we just pick a random SecretKey_cca, then we’ will getting irrelevant leakage indications (false positives) from the public parameters, as those are not masked at all.

- False positives are similar to “leaking” public modulus n in RSA, or public point in ECC.

Example 2: Test Vector Creation

For all lattice schemes – CCA KEM and Signature

Standard format PQC secret keys are complex mixtures of secret and public information:

- Saber (CCA) **SK** = (**z** ‖ Hash(PK) ‖ seed_A ‖ Encode(b_p) ‖ Encode(**s**))
- Kyber (CCA) **SK** = (Encode(**s**) ‖ Encode(t) ‖ ρ ‖ Hash(PK) ‖ **z**)
- Dilithium **SK** = (ρ ‖ K ‖ Hash(ρ, t_1) ‖ Encode($\mathbf{s_1}$) ‖ Encode($\mathbf{s_2}$) ‖ t_0)

Avoiding false positives from non-CSPs; we'd want to keep the public (ρ or seed_A) values static and only manipulate private polynomial **s**. This is analogous to keeping the "curve" constant with elliptic curves and just looking for leakage in the scalars.

As with RSA and ECC, the procedure for high-level test vector generation depends on the algorithm structure. We're proposing test vectors that "activate" CSP components only.

Example 3: Plaintext Checking Oracle

Fujisaki-Okamoto is Extremely Fragile

- The Plaintext Checking (PC) oracle leaks the result of CT == **CT'** comparison (Step 5).
- PC oracle reduces security from CCA to CPA.
- Many CPA lattice schemes leak secret keys in an adaptive chosen ciphertext attack.

Even though test vectors are not adaptive, we test specifically for PC oracle in Decapsulation e.g. by mismatching secret key with ciphertext. (CT, **SK** are not repeated; we're looking for leakage of **Z/K.**)

Subset **A**: CT = CCA.Encaps(PK), matching **SK**

Subset **B**: CT = CCA.Encaps(PK'), mismatch **SK**

CCA.Decaps(CT, SK):

1. (H(PK), **Z**, **SK'**) ← **SK**
2. **M** ← CPA.Decrypt(CT, **SK'**)
3. (**R**, **K**) ← H(H(PK), **M**)
4. **CT'** ← CPA.Encrypt(PK, **M**, **R**)
5. if CT == **CT'** then:
6. | **SS** ← H(**R**, **K**)
7. else:
8. | **SS** ← H(**R**, **Z**)
9. return **SS**

Conclusions

- ISO 17825 / TVLA leakage tests are useful as a sign-off and positive assurance.
  No key recovery, attack potential grading – has different goals from AVA_VAN.

- ISO 17825 being adopted FIPS 140-3 and can be used on Post-Quantum Crypto.

- Such testing should focus on *coverage;* aim to test all CSPs, everywhere. But care must be taken to avoid false positives (e.g. detection of PSP variables).

Big caveat: Do not let such testing replace security analysis in the design process!

"When a measure becomes a target, it ceases to be a good measure".

– Goodhart's law (of unintended consequences.)