WiP: Applicability of ISO Standard Side-Channel Leakage Tests to NIST Post-Quantum Cryptography

Markku-Juhani O. Saarinen
PQShield Ltd.
Oxford, United Kingdom
mjos@pqshield.com

***Abstract*—FIPS 140-3 is the main standard defining security requirements for cryptographic modules in U.S. and Canada; commercially viable hardware modules generally need to be compliant with it. The scope of FIPS 140-3 will also expand to the new NIST Post-Quantum Cryptography (PQC) standards when migration from older RSA and Elliptic Curve cryptography begins. FIPS 140-3 mandates the testing of the effectiveness of "non-invasive attack mitigations", or side-channel attack countermeasures. At higher security levels 3 and 4, the FIPS 140-3 side-channel testing methods and metrics are expected to be those of ISO 17825, which is based on the older Test Vector Leakage Assessment (TVLA) methodology. We discuss how to apply ISO 17825 to hardware modules that implement lattice-based PQC standards for public-key cryptography – Key Encapsulation Mechanisms (KEMs) and Digital Signatures. We find that simple "random key" vs. "fixed key" tests are unsatisfactory due to the close linkage between public and private components of PQC keypairs. While the general statistical testing approach and requirements can remain consistent with older public-key algorithms, a non-trivial challenge in creating ISO 17825 testing procedures for PQC is the careful design of test vector inputs so that only relevant Critical Security Parameter (CSP) leakage is captured in power, electromagnetic, and timing measurements.**

***Index Terms*—FIPS 140-3, Post-Quantum Cryptography, Side-Channel Attacks, DPA, DEMA, TVLA, ISO 17825**

I. Transitions: FIPS 140-3, Side-Channels, and PQC

The U.S. & Canadian standard for the security of cryptographic modules is FIPS 140-3 [1]; after two decades, validation of implementations to the older FIPS 140-2 standard [2] ended in 2021. The new standard states that:

> *Major changes in FIPS 140-3 are limited to the introduction of non-invasive physical requirements.*[1]

Non-invasive physical attacks use external physical measurements to derive secret information but do not modify the module's state. They are commonly known as Side-Channel Attacks (SCA) and have been widely known since the 1990s; Timing Attacks (TA) [4] and Differential Power Attacks (DPA) [5] are the most common ones.

Almost concurrently with the start of FIPS-mandated side-channel testing, NIST is transitioning to Post-Quantum Cryptography (PQC) standards for digital signatures and key establishment [6]. Hence designers of new high-assurance cryptographic modules for Government use are expected to meet both SCA and PQC requirements.

[1]However, there have been other substantial changes in FIPS 140 testing, such as the much more comprehensive entropy source requirements [3].

II. ISO 17825 Side-Channel Testing

A. Limitations

ISO 17825 [7] presents its tests as repeatable, "push-button" tests with moderate costs. The test metrics determine conformance to ISO 19790 [8] (and, by extension, future versions FIPS 140-3 [1]) at Security Levels 3 and 4. The procedure primarily tests for statistical signs of first-order leakage of secret information (Critical Security Parameters, CSPs). Key recovery is not expected to be demonstrated.

Due to its push-button leak-detection nature (without consideration for specific attacks), it is useful to think of ISO 17825 as defining a "least common denominator" base level for side-channel security. It is not commensurable with an Evaluation Assurance Level (EAL) gained from Vulnerability Analysis (AVA_VAN) under the Common Criteria scheme [9].

The [7] non-invasive test metrics exclude attacks that modify the module's state, either destructively or through temporary faults. The test control interfaces are similar to the cryptographic module's external API. Since fault attacks are excluded, the main requirement for laboratory equipment is to issue commands synchronously and gather high-precision measurements of power consumption (for DPA), electromagnetic emissions (for DEMA), or for Timing Attacks (TA).

B. General Test Procedure

Our working assumption is that a PQC Implementation Under Test (IUT) will be required to meet a similar non-invasive security as RSA and Elliptic Curve (ECDSA) implementations. The statistical methodology, calibration, and other technical requirements are largely the same as for older public-key cryptography in ISO 17825 [7] and ISO 20085 [10], [11]. Note that hybrid IUTs can implement both types of algorithms.

The most readily applicable test is the "general statistical test procedure" [7, Figure 7]. The current version of these tests create data subsets A and B of measurements (e.g., trace waveforms) with the IUT, where input test vectors to A and B differ the way input CSPs are selected.

Example: Subset A may use a fixed bit value in the 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 the leakage of the CSP and will result in a FAIL. If the confidence level is not reached, the test result is a PASS for the test vectors.

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.

The 2016 version of ISO 17825 [12] derived the sample size N (of waveforms) directly from the security level; $10,000$ for Level 3 and $100,000$ for Level 4, and the statistical threshold value was fixed at $C = \pm 4.5$. However, this leads to a statistically flawed experiment; for example, the *family-wise error rate* (FWER) was not considered [13], and hence the resulting confidence levels were not accurate.

Under the newer versions of ISO 17825 [7], the security level determines (Cohen's) standardized effect size; $d = 0.04$ for Level 3 and $d = 0.01$ for Level 4. This, together with the false-positive rate α, false-negative rate β are used to determine the number of traces N and threshold C. Bonferroni correction is used to address FWER.

III. PQC Experiment Classes

We will focus on NIST finalist lattice-based KEMs and Signature algorithms. At first, the tests may seem straightforward to set up; the private keys and shared secrets of NIST PQC algorithms are considered CSPs. But this creates a lot of false detections. NIST PQC private keys also contain components that can be derived from public information; those items are not CSPs, and detection of their "leakage" will create false positives. For purposes of our discussion, we exclude such public, shared items of information from the secret key, and both pk and sk are hence required for private-key operations.

Example: The public matrix or ring element in lattice-based cryptography (typically noted with the letter A) is not assumed to be secret. Many algorithms derive A on-the-fly from a much shorter seed value using an Extensible Output Function (XOF) – and have the seed appear in both public and secret keys. The role of A is analogous to public modulus n in the RSA or a generator in Elliptic Curve crypto; they are needed for private key operations but are also public parameters and hence need to be excluded from the distinguishing process.

There are additional, fundamental algorithmic differences between Post-Quantum Cryptography and older asymmetric algorithms such as RSA. For example, [7] states that *"Asymmetric cryptography repeatedly uses elementary operations."* This is less true for lattice-based algorithms than it is for Elliptic Curve or RSA cryptography. The structure of the newer algorithms is less homogenous and consists of many special processing steps in addition to repeated arithmetic iterations. Devastating leakage can occur at almost any stage of the algorithm, and hence full-algorithm testing is essential.

API Conventions: We define simple API abstractions for PQC algorithms at the "I/O Boundary" as required by the ISO 17825 standard. In practice, these APIs may need to be implemented as multiple discrete steps that set up or import keys securely for the operation. Export may also be in scope.

We use $(\mathsf{pk}, \mathsf{sk})$ to denote public and private components of the keypair, preferably refactored in a way that sk contains only secret (CSP) parameters whose leakage leads to a security compromise. If masking is used, it is assumed that a fixed sk is remasked ("refreshed") for each private key operation trace. Some APIs allow randomizers to be passed with the API; $[\mathsf{seed}]$ represents this. Such determinism is helpful for testing, but the seeds themselves are CSPs and must be handled appropriately in a side-channel context.

A. Classes CPA and CCA: Key Establishment Mechanisms

PQC Key Encapsulation Mechanisms (KEM) can be used in many different roles; for ephemeral key establishment (in interactive protocols such as TLS) and with long-term keys for public-key encryption (in applications such as e-mail), and for also for challenge-response authentication. All of these use cases have different security requirements.

For technical definitions of CPA and CCA security, see [14]. CCA KEMs are often built on top of CPA components using the Fujisaki-Okamoto (FO) transform [15], [16]. The general test procedure is not adaptive and hence excludes interactive forms of tests ("experiments") required for CPA and CCA security claims. The "CPA" and "CCA" terms are used here just to refer to the underlying components of KEMs. Security testing of CPA components is necessary to obtain assurances about the security of the CCA construct as well.

Examples: Non-specific t-tests are commonly used in the literature to demonstrate basic security against side-channel attacks of PQC KEMs. Welch's t-test method was used in [17] and [18] to demonstrate the security of masked Kyber implementations. However, these tests directly invoked internal subcomponents – such interfaces may not be available for ISO 17825 testing. In [19] the t-test was also used in the non-specific "fix vs. random" A/B test mode to verify the security of the decapsulation component of SABER KEM against simple DPA (however, a neural network profiling/template attack was used later to attack the implementation [20]).

CPA.API – Encrypt/Decrypt API Abstraction.

In CPA testing, we use the CPA Encrypt/Decrypt subcomponent of KEM algorithms built using the FO transform.

Initialize: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \texttt{CPA.KeyGen}([\mathsf{seed}])$
Public: $\mathsf{ct} \leftarrow \texttt{CPA.Encrypt}(\mathsf{pk}, \mathsf{m}, [\mathsf{seed}])$
Private: $\mathsf{m} \leftarrow \texttt{CPA.Decrypt}(\mathsf{ct}, \mathsf{pk}, \mathsf{sk})$

CPA.PD class: Public key distinguisher.

The IUT is $\texttt{CPA.Decrypt}()$. The traces for set A use a fixed (but refreshed) keypair. Random, unique keypairs are used for each trace in set B. The ciphertexts ct are created (typically offline) using random messages m and matching keypairs to those used by IUT to decrypt.

CPA.PD.A: m = Random, $(\mathsf{pk}, \mathsf{sk})$ = Fixed
CPA.PD.B: m = Random, $(\mathsf{pk}', \mathsf{sk}')$ = Varying

Discussion: CPA.PD does not appear to be very important in practice as the main implication of a FAIL is is that the attacker can identify the fixed public key used, which is usually inconsequential. However, a PASS on CPA.PD implies that the secret key is also protected; this is a very positive feature but may require unnecessary performance compromises. Implementations are not expected to protect public parameters.

CPA.SD class: Secret key distinguisher.

The IUT is $\texttt{CPA.Decrypt}()$. All ciphertexts decrypted in set A are generated with a fixed (but refreshed) keypair $(\mathsf{pk}, \mathsf{sk})$. For set B a modified secret key sk' is used. Random messages m are encrypted in both cases with pk to create the ciphertexts.

Tweak: $\mathsf{sk}' \leftarrow \texttt{CPA.KeyMod}(\mathsf{pk}, \mathsf{sk})$

CPA.SD.A: m = Random, $(\mathsf{pk}, \mathsf{sk})$ = Fixed
CPA.SD.B: m = Random, pk = Fixed, sk' = Varying

Discussion: The CPA.SD test is applicable only if the raw private key operation $\texttt{CPA.Decrypt}()$ does not return a FAIL but will generally yield an invalid m' in case of an invalid secret key. This is an assumption that can be made with the underlying CPA decryption primitive of most Lattice-based KEMs. The test also assumes that the private and public key components have been refactored in a way that allows one to create a meaningfully modified secret key sk' with differences limited to CSP components. In classical cases, one could simply flip a bit in the CSP / secret key but for PQC the required synthesis is more complex. The resulting sk' is likely to be inconsistent with the public key, and hence a special test entry method may be needed. The modification must be meaningful also in the sense that it affects the result; $\texttt{CPA.Decrypt}(\mathsf{ct}, \mathsf{pk}, \mathsf{sk}) \neq \texttt{CPA.Decrypt}(\mathsf{ct}, \mathsf{pk}, \mathsf{sk}')$.

We recommend that the $(\mathsf{ct}, \mathsf{pk}, \mathsf{sk}, \mathsf{sk}')$ test vector datasets for CPA.SD – or code for generating them – are published and shared to guarantee the consistency of the tests.

CCA.API – Encapsulate/Decapsulate API Abstraction.

PQC algorithms generally provide the basic Key Encapsulation Mechanism (KEM) API for their CCA security versions. There is no message m to select, but a random shared secret ss is created in encapsulation. The algorithms are often designed for *implicit rejection*; decapsulation may fail, but a synthetic ss value is generated instead of a return code [16].

Initialize: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \texttt{CCA.KeyGen}([\mathsf{seed}])$
Public: $(\mathsf{ct}, \mathsf{ss}) \leftarrow \texttt{CCA.Encaps}(\mathsf{pk}, [\mathsf{seed}])$
Private: $\mathsf{ss} \leftarrow \texttt{CCA.Decaps}(\mathsf{ct}, \mathsf{pk}, \mathsf{sk})$

CCA.PD and **CCA.SD**: One can define a CCA.PD public key distinguisher in an analogous manner to CPA.PD; however, failure of this test can be a result of leakage of public variables, not CSPs. CCA.SD can be constructed using synthetic keypairs in a similar fashion to CPA.SD. We note that FO re-encryption will generally fail due to mismatch between pk and synthetic sk', so CCA.SD has limited use.

CCA.PC class: Plaintext Checking Distinguisher.

The IUT is $\texttt{CCA.Decaps}()$ and the decapsulation operation uses a fixed keypair $(\mathsf{pk}, \mathsf{sk})$ for all tests. The difference is in ciphertexts; Set A consists of valid ciphertexts, while set B ciphertexts are encapsulated with a mismatching public key $\mathsf{pk}', \mathsf{pk}' \neq \mathsf{pk}$.

CCA.PC.A: $\mathsf{ct} = \texttt{CCA.Encaps}(\mathsf{pk})$, matching sk
CCA.PC.B: $\mathsf{ct} = \texttt{CCA.Encaps}(\mathsf{pk}')$, mismatch

Discussion: This CCA test invokes the Plaintext Checking (PC) oracle in Fujisaki-Okamoto transform present in many CCA KEMs. The existence of a PC oracle generally implies that the scheme's security is reduced from CCA to CPA level. It may also lead to compromise of CSPs [21].

B. Class Sig: PQC Signature testing

The tests focus on the signature function (private key operations) as it is assumed that signature algorithms are used with long-term keys. Key generation is rare.

Examples: Authors of [22] the use the TVLA / t-test to find leakage points in an implementation of the Dilithium signature algorithm. They then describe a masked variant of Dilithium and use the t-test to verify the security of components ("gadgets") against leakage. However, in ISO 17825 methodology, the tester may not be able to access subcomponents.

Sig.API – Signature API abstractions.

Initialize: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \texttt{Sig.KeyGen}([\mathsf{seed}])$

API with a signed message envelope sm of message m:

Private: $\mathsf{sm} \leftarrow \texttt{Sig.Sign}(\mathsf{m}, \mathsf{pk}, \mathsf{sk}, [\mathsf{seed}])$
Public: $\{\mathsf{m}, \mathsf{Fail}\} \leftarrow \texttt{Sig.Open}(\mathsf{sm}, \mathsf{pk})$

Alternative API with a detached signature s of message m:

Private: $\mathsf{s} \leftarrow \texttt{Sig.Signature}(\mathsf{m}, \mathsf{pk}, \mathsf{sk}, [\mathsf{seed}])$
Public: $\{\mathsf{Ok}, \mathsf{Fail}\} \leftarrow \texttt{Sig.Verify}(\mathsf{s}, \mathsf{m}, \mathsf{pk})$

Sig.MD class: Message distinguisher.

The IUT is either $\texttt{Sig.Sign}()$ or $\texttt{Sig.Signature}()$. Messages m and m' have equivalent length, e.g., 256 bits.

Sig.MD.A: m' = Random, $(\mathsf{pk}, \mathsf{sk})$ = Fixed
Sig.MD.B: m = Fixed, $(\mathsf{pk}, \mathsf{sk})$ = Fixed

Discussion: The message being signed may or may not be a CSP, depending on the application. For side-channel security we do not recommend deterministic signing that always produces the same signature, given the same m and keys. Dilithium can be operated in randomized mode too [23].

Sig.PD class: Public key distinguisher. The IUT is either $\texttt{Sig.Sign}()$ or $\texttt{Sig.Signature}()$. Fixed and varying keypairs can be generated offline for the test.

Sig.PD.A: m = Fixed, $(\mathsf{pk}, \mathsf{sk})$ = Fixed
Sig.PD.B: m = Fixed, $(\mathsf{pk}', \mathsf{sk}')$ = Varying

Discussion: Like with its CPA.PD counterpart, failure of Sig.PD test does not necessarily imply leakage of secret CSP information or other information leading to a signature forgery.

Sig.SD class: Secret key distinguisher.

The IUT is either Sig.Sign() or Sig.Signature(). For this test, one needs to be able to modify the secret components sk of the keypair while causing minimal changes to the public values pk; ideally, one constructs synthetic keypairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}, \mathsf{sk}')$ with differences only in the actual secret component. The exact procedure for such test vector generation depends on the algorithm under test, and should be standardized.

Tweak: $\mathsf{sk}' \leftarrow \mathtt{Sig.KeyMod}(\mathsf{pk}, \mathsf{sk})$

Sig.SD.A: m = Fixed, pk = Fixed, sk′ = Varying
Sig.SD.B: m = Fixed, pk = Fixed, sk = Fixed

Discussion: As with CPA.SD and CCA.SD, the synthetic secret keys sk′ do not need to be completely random but can have only small differences to sk; the aim is to keep changes to the pk public-key parameters required for verification at a minimum. For example, a Dilithium [23] secret key contains (ρ, K, tr) parameters that can be derived from the public key – in addition to the secret vectors (s_1, s_2). Only the latter usually needs to be protected against leakage. A practically relevant testing procedure is able to perform private key operations with the same ρ public values but varying (s_1', s_2') parameters; and compare those traces against fixed (s_1, s_2).

IV. Conclusions

We predict that the FIPS 140-3 non-invasive security baseline ISO 17825 will apply very broadly to commercial hardware cryptography. It can be used to demonstrate the existence and coverage of side-channel countermeasures – but these PASS/FAIL leakage detection tests do not really measure the cost of an attack by an adversary.

We have presented classes of simple ISO 17825 / TVLA "push button" test procedures for PQC KEM and Signature algorithms. Perhaps unsurprisingly, we find that side-channel leakage testing of PQC modules can't be entirely "black box", but requires carefully designed test vectors.

The straightforward application of "random vs. fixed" keypair distinguisher (CPA.PD, CCA.PD, Sig.PD) is not very meaningful as leakage of public key does not imply leakage of CSP information. Due to the complex linkage between private and public parts of the keypair in PQC, one needs a special method of constructing synthetic keypairs that modify only the secret CSP components without modifying the public ones.

For PQC key establishment (KEMs) implemented with the Fujisaki-Okamoto paradigm, verification of direct leakage of CSPs can be obtained via the CPA Decrypt component (CPA.SD). One needs to also test for the presence of the plaintext checking oracle (CCA.PC) for partial CCA assurance.

For digital signatures, one generally wants to implement randomized signatures instead of deterministic signatures; this can be verified with a message distinguisher (Sig.MD). As with KEMs, verification of direct secret leakage requires an understanding of the scheme to generate differentiated keys sk′ that can be used for CSP leakage detection (Sig.SD).

Standardized PQC test vector generation procedures are needed to obtain consistent testing results in all cases.

References

[1] NIST. Security requirements for cryptographic modules. Federal Information Processing Standards Publication FIPS 140-3, March 2019. doi:10.6028/NIST.FIPS.140-3.

[2] NIST. Security requirements for cryptographic modules. Federal Information Processing Standards Publication FIPS 140-2 (With change notices dated October 10, 2001 and December 3, 2002), May 2001. doi:10.6028/NIST.FIPS.140-2.

[3] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish, and Mike Boyle. Recommendation for the entropy sources used for random bit generation. NIST Special Publication SP 800-90B, January 2018. doi:10.6028/NIST.SP.800-90B.

[4] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. doi:10.1007/3-540-68697-5_9.

[5] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. doi:10.1007/3-540-48405-1_25.

[6] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the second round of the NIST post-quantum cryptography standardization process. Interagency or Internal Report NISTIR 8309, National Institute of Standards and Technology, July 2020. doi:10.6028/NIST.IR.8309.

[7] ISO. Information technology – security techniques – testing methods for the mitigation of non-invasive attack classes against cryptographic modules. Working Draft ISO/IEC WD 17825:2021(E), International Organization for Standardization, 2021.

[8] ISO. Information technology – security techniques – security requirements for cryptographic modules. Standard ISO/IEC 19790:2012(E), International Organization for Standardization, 2012.

[9] Common Criteria. Common methodology for information technology security evaluation: Evaluation methodology. Specification: Version 3.1 Revision 5, April 2017. URL: https://commoncriteriaportal.org/cc/.

[10] ISO. It security techniques – test tool requirements and test tool calibration methods for use in testing non-invasive attack mitigation techniques in cryptographic modules – part 1: Test tools and techniques. Standard ISO/IEC 20085-1:2019(E), International Organization for Standardization, 2019. URL: https://www.iso.org/standard/70081.html.

[11] ISO. IT security techniques – test tool requirements and test tool calibration methods for use in testing non-invasive attack mitigation techniques in cryptographic modules – part 2: Test calibration methods and apparatus. Standard ISO/IEC 20085-2:2020(E), International Organization for Standardization, 2020. URL: https://www.iso.org/standard/70082.html.

[12] ISO. Information technology – security techniques – testing methods for the mitigation of non-invasive attack classes against cryptographic modules. Standard ISO/IEC 17825:2016, International Organization for Standardization, 2016. URL: https://www.iso.org/standard/82422.html.

[13] Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284. Springer, 2019. doi:10.1007/978-3-030-34618-8_9.

[14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. doi:10.1201/b17668.

[15] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999. doi:10.1007/3-540-48405-1_34.

[16] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai

and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017. doi:10.1007/978-3-319-70500-2_12.

[17] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):173–214, 2021. doi:10.46586/tches.v2021.i4.173-214.

[18] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Daan Sprenkels. First-order masked Kyber on ARM Cortex-M4. IACR ePrint 2022/058, 2022. URL: https://eprint.iacr.org/2022/058.

[19] Michiel Van Beirendonck, Jan-Pieter D'Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of SABER. *ACM J. Emerg. Technol. Comput. Syst.*, 17(2):10:1–10:26, 2021. doi:10.1145/3429983.

[20] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked IND-CCA secure saber KEM implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):676–707, 2021. doi:10.46586/tches.v2021.i4.676-707.

[21] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic Power/EM analysis on post-quantum KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):296–322, November 2021. doi:10.46586/tches.v2022.i1.296-322.

[22] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium - efficient implementation and side-channel evaluation. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 344–362. Springer, 2019. doi:10.1007/978-3-030-21568-2_17.

[23] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-dilithium: Algorithm specifications and supporting documentation (version 3.1). NIST PQC Project, 3rd Round Submission Update, February 2021. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[24] ISO. Information technology – security techniques – test requirements for cryptographic modules. Standard ISO/IEC 24759:2017(E), International Organization for Standardization, 2017.

[25] NIST and CCCS. Implementation guidance for FIPS 140-3 and the cryptographic module validation program. CMVP, November 2021. URL: https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements.

[26] Kim Schaffer. CMVP approved non-invasive attack mitigation test metrics: CMVP validation authority updates to ISO/IEC 24759. NIST Special Publication SP 800-140F, March 2020. doi:10.6028/NIST.SP.800-140F.

[27] Kim Schaffer. CMVP approved non-invasive attack mitigation test metrics: CMVP validation authority updates to ISO/IEC 24759. NIST Special Publication SP 800-140F Rev 1 (Draft), August 2021. doi:https://doi.org/10.6028/NIST.SP.800-140Fr1-draft.

[28] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for sidechannel resistance validation. CMVP & AIST Non-Invasive Attack Testing Workshop (NIAT 2011), September 2011. URL: https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf.

[29] Josh Jaffe, Pankaj Rohatgi, and Marc Witteman. Efficient sidechannel testing for public key algorithms: RSA case study. CMVP & AIST Non-Invasive Attack Testing Workshop (NIAT 2011), September 2011. URL: https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/09_jaffe.pdf.

[30] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015. doi:10.1007/978-3-662-48324-4_25.

[31] Michael Tunstall and Gilbert Goodwill. Applying TVLA to public key cryptographic algorithms. IACR ePrint 2016/513, 2016. Presented at International Cryptography Module Conference – ICMC 2016. URL: https://eprint.iacr.org/2016/513.

[32] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2018. doi:10.1007/978-3-030-15462-2_5.

APPENDIX

Guide to the documents. The FIPS 140-3 document itself [1] is short and structured as a list of pointers to documents that modify the ISO security standards 19790 [8] and 24759 [24], the actual foundation documents. Requirements are issued in six NIST Special Publications in the 800-140 series (A-F), as well as in FIPS 140-3 Implementation Guidance [25].

For side-channel countermeasure testing, ISO 19790 [8] refers to its Annex F, as do various sections in ISO 24759 [24]. Annex F is an empty placeholder. For FIPS 140-3 it is amended by SP 800-140F [26], which was also initially empty.

In August 2021, a draft of SP 800-140F Revision 1 [27] was circulated, which provided references to ISO 17825 [12] for non-invasive attack mitigation testing.

Relation to TVLA. ISO 17825 tests are largely based on Test Vector Leakage Assessment (TVLA) methodology first proposed for AES and RSA by CRI / Rambus in 2011 [28], [29] and later refined in additional work [30], [31]. ISO 17824 and the TVLA method have been criticized for the non-detection of practical higher-order attacks [32] and also for statistical correctness of its experiments [13]. Newer working drafts of ISO 17825 [7] adopt Bonferroni correction to address some of the correctness issues.

Note on Calibration. The draft SP 800-140F Revision 1 [27] also references ISO 20085-1 [10] for test tools and techniques, and ISO 20085-2 [11] for calibration methods and apparatus. Since [27] is in draft status, NVLAP testing laboratories are not offering official testing capability, and the FIPS 140-3 Implementation Guidance [25] still discusses non-invasive attacks under the general "mitigation of other attacks" banner. However, it is reasonable to expect that ISO 17825 will be adopted for this purpose.

The calibration process in [11] states that the test equipment must be able to detect leakage and fail certain specified IUTs. It also states that "an IUT which is built to pass, shall pass," (Sect 6.2.4). Hence the goal of these calibrated tests should not be seen as "detecting all side-channel leakage," but detecting leakage if and only if it exceeds a certain threshold.