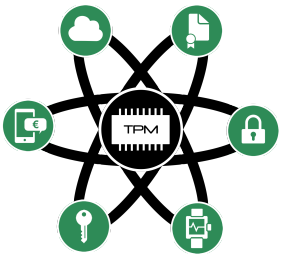

FutureTPM

D2.3

Third Report on New QR Cryptographic Primitives

Project number:	779391
Project acronym:	**FutureTPM**
Project title:	Future Proofing the Connected World: A Quantum-Resistant Trusted Platform Module
Project Start Date:	1st January, 2018
Duration:	36 months
Programme:	H2020-DS-LEIT-2017
Deliverable Type:	Report
Reference Number:	DS-LEIT-779391 / D2.3 / v1.0
Workpackage:	WP 2
Due Date:	December 31, 2020
Actual Submission Date:	December 22, 2020
Responsible Organisation:	IBM
Editor:	Bertram Poettering
Dissemination Level:	PU
Revision:	v1.0
Abstract:	This document describes the choices (and justification) of the public, symmetric and privacy-enhancing primitives for the TPM constructions.
Keywords:	Foundational primitives, basic protocols

The project FutureTPM has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779391.

Editor

Bertram Poettering (IBM)

Contributors (ordered according to beneficiary numbers)

TEC, SURREY, UBITECH, RHUL, IBM, UB, IFAG, IFAT, UL, INESC-ID, UPRC

Contents

1 Introduction

The present document is Deliverable D2.3 of Work Package WP2 of the FutureTPM project. The goal of WP2 is the identification of cryptographic schemes and primitives that can withstand quantum attackers. Within the FutureTPM project, such schemes and primitives would serve in the security foundation of quantum-resistant TPMs. This document should be seen as the logical successor of first D2.1 [13] and then D2.2 [14]. Generally speaking, Deliverable D2.1 lays out the general scenery, including descriptions of various attack strategies that quantum adversaries might pursue, and methods to overcome these. Building on the overview provided by D2.1, Deliverable D2.2 distills concrete recommendations regarding which schemes should be implemented in future TPMs, and in particular in the FutureTPM prototypes. The present document, Deliverable D2.3, refines the recommendations of D2.1 and D2.2 on the basis of both new research results and feedback received from the other WPs of the project that implement, prototype, and evaluate our suggestions. The extension from D2.1 and D2.2 to D2.3 can be summarized as follows.

1.1 Structure of the Document

The present Deliverable D2.3 builds on the results of Deliverables D2.1 and D2.2. Roughly, it provides the following:

- In **Section 2** we revisit the list of recommended schemes from D2.2 in light of the progress made by academic cryptanalysis and research since the release of D2.2. Concretely, new information on the security and efficiency of some schemes that were recommended in D2.2 was made available, and we correspondingly slightly adjust our recommendations. These adjustments are coordinated with WP5 (which focuses on implementational aspects), and they specifically take into account the recommendations of the NIST PQC standardization process[1].

- The primitives recommended in D2.2 are mostly basic encryption or authentication primitives. However, an important class of schemes required in the TPM setting is formed by privacy preserving authentication protocols, including Direct Anonymous Attestation (DAA). As none of the DAA solutions standardized so far by the Trusted Computing Group (TCG) is expected to be resilient against attacks employing quantum computers, a new quantum-safe anonymous attestation solution had to be developed. In Deliverable D2.1 we introduced two initial versions of quantum-secure DAA schemes, and in D2.2 we briefly reported on options for how to improve on them. In **Section 3** we expose and discuss a new DAA solution that is secure against quantum attackers yet more efficient than those of D2.1.

- In **Section 4** we discuss the general aspects of modeling the security of cryptographic primitives that are particularly relevant in settings that assume quantum adversaries and/or involve TPMs. This refines and clarifies on similar discussions we conducted in the earlier deliverables D2.1 and D2.2 of this work package.

- The security of some of the cryptographic primitives that were recommended in D2.2 and are re-assessed in Section 2 rely on hardness assumptions that could be considered relatively untested. For instance, some encryption schemes are based on the (assumed)

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/

hardness of certain problems related to structured lattices. These types of hardness assumption have been introduced to cryptography much more recently than the over four decades old RSA and DLP problems. A direct consequence of this is that the structured lattice assumptions did receive less testing by cryptanalysis, and ongoing academic efforts could ultimately lead to some of the D2.2-recommended schemes being eventually broken. To address this, in **Section 5** we study hybrid solutions that combine two ingredient primitives of the same type to a single primitive such that the latter is secure if at least one of the former is. If a hybrid is used, an untested assumption failing will thus lead to a broken TPM implementation only with reduced likelihood. With other words, by hedging against unforeseen security failures, hybrids promise an eased risk management.

This document further contains a summary and conclusion, a section with references, and the following three appendices:

- in **Appendix A** we list the abbreviations and acronyms used in this report, together with their expansions;
- in **Appendix B** we recall the asymmetric cryptographic primitives that take part in a currently ongoing competition organized by the NIST;
- in **Appendix C** we provide explicit references to the technical specifications of the cryptographic primitives that we recommend in this report.

2 Update on Deliverable D2.2

2.1 Reassessment of our Recommendations

In Deliverable D2.2 [14] of this project we recommended a variety of cryptographic primitives for the use in quantum-resilient TPMs. More precisely, we put forward two lists: A main list of recommended primitives and a secondary list of optional primitives. The reason to maintain the secondary list was to appreciate solutions that were deemed secure, yet were potentially less efficient to implement or less likely to be adopted by industry, so they could serve as backup options for unforeseen cases. In D2.2 we also carefully documented the selection criteria for the main list, and the rationale behind it. In Table 1 we recall the entries of the first list (of recommended schemes). See Tables 9–12 in D2.2 for the full selection.

Category	Recommended scheme
hash functions	SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512
block ciphers	AES-256
symmetric authentication	HMAC
symmetric encryption	CFB
authenticated encryption	– (only optional candidates)
public key encryption	NewHope, CRYSTALS-Kyber, NTTRU, BIKE
signature schemes	BLISS, CRYSTALS-Dilithium, Rainbow, SPHINCS+

Table 1: Scheme recommendations from Deliverable D2.2. (This list does not include the optional schemes.)

The horizontal line in Table 1 separates the symmetric schemes (top) from the asymmetric schemes (bottom). Regarding the symmetric schemes it can be stated as a general rule that they are standard solutions in the domain of symmetric cryptography that predate the quantum resilience debate. To ensure that the schemes are also secure against quantum adversaries, in Deliverable D2.2 we formulated lower bound requirements on the key sizes, block sizes, and digest sizes. See in particular D2.1 for why such conditions are necessary and (by the current academic understanding) sufficient.

The eight asymmetric schemes in Table 1 are all recent designs (from the past decade or so) that have been specifically proposed as quantum-resilient candidates. Here we based our choice both on the expected security and the efficiency profiles of the schemes, where efficiency encompasses computation time and sizes of keys, ciphertexts, and signatures.

Six of the asymmetric schemes from Table 1, specifically the public key encryption schemes NewHope, CRYSTALS-Kyber, and BIKE, and the signature schemes CRYSTALS-Dilithium, Rainbow, and SPHINCS+, were also submitted to the currently ongoing NIST competition[2][3] which aims at standardizing quantum-resilient cryptographic primitives. This competition was started in 2016, it consists of multiple rounds that each reduce the set of remaining candidates, until, but not before 2022, a concrete selection of to-be-standardized schemes is made.[4]

[2] https://csrc.nist.gov/Projects/post-quantum-cryptography/

[3] https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline

It is a great opportunity for the FutureTPM project to be able to align their selection of schemes with the schemes submitted to, and remaining in, the NIST process. Let us highlight three particular benefits:

1. **Security.** The attention of academic cryptanalysis efforts is currently mainly focused on the NIST submissions, and the FutureTPM project directly benefits from any emerging corresponding cryptanalytic result.

2. **Availability.** Open-source implementations of the NIST submissions are readily available for many different platforms (desktop CPUs, microcontrollers, custom hardware, etc.). Such implementations can be used within the FutureTPM project to accurately estimate the efficiency of candidate schemes within prototyped TPMs.

3. **Adoption.** The recommendations made in FutureTPM project deliverables are most likely adopted by industry if they correspond with the schemes standardized by NIST. Presenting industry-adoptable solutions is a main goal of the project.

In July 2020 the NIST announced which second-round candidates of the competition progress to the third round.[5] More precisely, for both public key encryption and signature schemes it announced a list of 'Finalists' and a list of 'Alternate Candidates'. According to the NIST report, finalists are likely to be standardized by the end of the competition, while alternate candidates are likely not. The idea is that the alternate candidates are currently recognized as secure and well-performing, yet not as much as the candidates tagged as finalists. For reference we reproduce the list of NIST Round-3 submissions in Appendix B.
In Table 2 we indicate the status of our D2.2 recommendations in light of the NIST evaluation report from July 2020. Note that five out of six evaluated recommendations from D2.2 were confirmed as simultaneously secure and well-performing, and in each category we have at least one finalist.

Category	Name	Assumption	In third round?
public key encryption	CRYSTALS-Kyber	lattice	finalist
public key encryption	BIKE	code	alternative
public key encryption	NewHope	lattice	no
public key encryption	NTTRU	lattice	(not submitted)
signature scheme	CRYSTALS-Dilithium	lattice	finalist
signature scheme	Rainbow	multivariate	finalist
signature scheme	SPHINCS+	hash	alternative
signature scheme	BLISS	lattice	(not submitted)

Table 2: Intersection of NIST third round finalists and our recommendations from D2.2.

The only D2.2-recommended scheme evaluated by the NIST that was listed neither as a finalist nor as an alternate candidate is NewHope. The rationale for removing NewHope from the competition can be found in Section 3.12 of NIST document NISTIR 8309.[6][7] The argument given is

[5] https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement
[6] https://csrc.nist.gov/publications/detail/nistir/8309/final
[7] https://doi.org/10.6028/NIST.IR.8309

that a close comparison of CRYSTALS-Kyber and NewHope reveals that "in a technical sense, the security of NewHope is never better than that of KYBER".[8] Thus, as CRYSTALS-Kyber was confirmed as a candidate of Round 3, even though no attack on NewHope has been found, there doesn't seem to be a reason for keeping NewHope in the competition. Note that this does not mean that any cryptanalytic insights would imply that NewHope is insecure.[9] We are thus pleased to observe that effectively all our recommendations have been confirmed by the NIST process as suitable choices (this includes NewHope, but excludes NTTRU and BLISS which were not evaluated in the NIST process).

2.2 Updates to our Recommendations

In this report we slightly update the recommendations from D2.2. Regarding symmetric primitives (hash functions, block ciphers, authentication schemes, encryption schemes, authenticated encryption schemes), we confirm all our prior recommendations, including the optional ones. We further continue to recommend four out of the eight asymmetric schemes proposed in D2.2. However, we move the public key encryption schemes NewHope and NTTRU, as well as the signature schemes BLISS and Rainbow, to the "optional" category. Further, we add the signature scheme "Picnic" to the same list. (This signature scheme was not yet recommended in D2.2.)

We provide rationale for our modifications below. Please find the updated list of recommendations in Table 3.

2.2.1 Rationale for moving NewHope, NTTRU, and BLISS to the Optional Set

Our decision to remove the schemes NewHope, NTTRU, and BLISS from the primary choice list for the use in quantum-resilient TPMs is not based on a suspicion that they might not be secure schemes. Rather, we remove them to ease the adoption of our recommendations by industry: We expect that CRYSTALS-Kyber, BIKE, CRYSTALS-Dilithium, and SPHINCS+ are better candidates for this, as they remain in the NIST competition and might eventually be standardized while this is unlikely to be the case for the schemes we removed.

2.2.2 Rationale for moving Rainbow to the Optional Set

The fact that the NIST included the Rainbow signature scheme in the 'finalists' set of the third round of its competition (see Table 2) indicates that Rainbow is considered a promising candidate to provide quantum resilience. Only a few weeks after the third round officially started, in October 2020, the article "Improved Cryptanalysis of UOV and Rainbow"[10] by Ward Beullens reported new cryptanalytic results on the security of Rainbow.[11] Specifically, two new attacks

[8] The quote is taken verbatim from the NIST report, and it refers to the observation that the Ring-LWE assumption (of NewHope) is generically stronger than the Module-LWE assumption (of CRYSTALS-Kyber). However, in terms of bit-level security estimations, the NewHope 1024 parameter set is more secure than the CRYSTALS-Kyber parameter set. While we follow NIST's recommendations with preferring CRYSTALS-Kyber over NewHope, our assessment remains that NewHope is a very competitive scheme.

[9] A minor issue in the NewHope version that was submitted to Round 2 of the NIST competition was identified in [6], confirmed by the NewHope team in https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/0T7Hyd_NT_Y/m/iDdVR35QAAAJ, and fully removed by a minimal update of the NewHope specification. In our above statement about NewHope we refer to the fixed version.

[10] unpublished work; preprint available at https://eprint.iacr.org/2020/1343.pdf

[11] The attacks were acknowledged by members of the Rainbow team in https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/70We3SNi7Ss

Category	Recommended scheme
mandatory schemes:	
hash functions	SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512
block ciphers	AES-256
symmetric authentication	HMAC
symmetric encryption	CFB
authenticated encryption	– (only optional candidates)
public key encryption	CRYSTALS-Kyber, BIKE
signature schemes	CRYSTALS-Dilithium, SPHINCS+
optional schemes:	
hash functions	SHAKE128, SHAKE256, BLAKE2b-256, BLAKE2b-384, BLAKE2b-512, SM3, Ascon-XOF
block ciphers	Rijndael-256(192), Rijndael-256(256), Camellia-256
symmetric authentication	KMAC, CMAC
symmetric encryption	CBC, CTR
authenticated encryption	CCM, GCM, EAX, Ascon
public key encryption	**NewHope**, **NTTRU**
signature schemes	**BLISS**, **Rainbow**, **Picnic**

Table 3: Updated list of recommended symmetric and asymmetric cryptographic primitives. We modified our recommendations since D2.2 only with respect to the five schemes typeset in **bold** font.

against Rainbow were uncovered, each of which makes evident that the Rainbow configurations submitted to Round 3 fall short of satisfying the target security requirements formulated by the NIST. On first sight this seems troubling enough to make the removal of Rainbow from both the NIST competition and our set of recommendations unavoidable. On the other hand, Beullens concludes his article by clarifying that the new attacks, like similar prior attacks against Rainbow, require exponential running time, and that thus a rather small increase of parameters would be sufficient to let Rainbow again meet the NIST requirements.
Our assessment of the status of Rainbow for the use in a future TPM is as follows. On the one hand, while the new attacks nominally break the Rainbow versions as proposed to the NIST, neither do the attacks break Rainbow in practice nor would a TPM implementation be bound to operate Rainbow according to the Round 3 parameters (instead of with larger ones). Further, as Rainbow is the only multivariate scheme in our selection (see Table 2), it crucially adds to the diversity of our selection. Overall, our decision is to move Rainbow to the set of optional schemes (instead of removing it altogether). However, we also emphasize that future cryptanalytic results on Rainbow might emerge and should carefully be watched-out for.

2.2.3 Rationale for adding Picnic to the Optional Set

From a theory perspective, the most basic possible cryptographic component is a one-way function (a function with no specific properties beyond being hard to invert on random inputs). As it is a long-known result that one-way functions, in principle, are sufficient to construct signature schemes, and from a security stand point it is always advisable to select schemes based on the weakest possible hardness assumptions, it has to be noted that signature schemes used in practice are routinely based on considerably stronger components that exhibit an exploitable algebraic structure. (In particular the schemes CRYSTALS-Dilithium, BLISS, and Rainbow that we recommend in Table 3 are examples for this.) Only two NIST submissions are more directly based on one-way functions: SPHINCS+ and Picnic. We initially, in D2.2, only recommended the former as, out of the two, it has the more conservative design. Further, initial variants of Picnic were reported to be insecure and had to be fixed. (Similar concerns were also raised in NIST's second-round evaluation report[12][13].) However, the NIST did advance Picnic nevertheless to the third round of the competition, though only as an alternate candidate, with the main reason that the design of Picnic promises more options to improve its efficiency than its competitor SPHINCS+. That is, the NIST supports Picnic not for the performance it has today, but for the performance it might one day have. We believe that this is a convincing argument, and we thus added Picnic to our list of (optional) recommendations.

2.3 On Alternative Choices

The main goal of this work package of the FutureTPM project is to recommend a small set of cryptographic primitives that are both suitable and sufficient for defining a quantum-resistant TPM. Our approach was to first identify an initial set of options in Deliverable D2.1, to then thin out this set in D2.2 by taking cryptanalytic results and input from other work packages about the practical efficiency of the primitives into account, and to lastly revisit this list one more time in order to distill a final set of recommendations communicated in D2.3 (the present document). The results of this process, compressed into a nutshell, are presented in Table 3. While we do recommend all schemes in the table as suitable for a future TPM, and we extensively tested them in our implementations and use cases, we do not mean to suggest that our choice is the only possible one. More concretely, none of the NIST competition finalists (see Appendix B for the complete list) seems to be unsafe or otherwise specifically unfit for being employed in a future TPM. The same holds for many recently emerged symmetric primitives and schemes.[14] However, within this project we implemented and closely evaluated only the schemes listed in Table 3, and for those we confirm both that the schemes are efficient enough for TPM implementations, and that the emerging structures (key sizes, ciphertext sizes, signature sizes, state sizes) are not too large. We leave the evaluation of other candidates, in particular of the remaining finalists of the NIST competition, for future work.

[12] https://csrc.nist.gov/publications/detail/nistir/8309/final

[13] https://doi.org/10.6028/NIST.IR.8309

[14] For instance, the SATURNIN suite was designed specifically to be quantum-resilient, see https://project.inria.fr/saturnin/; also, some of the schemes submitted to, or scoring at, the CAESAR competition have quantum-resilient modes; see https://competitions.cr.yp.to/caesar.html. (Out of the latter, Table 3 only suggests Ascon explicitly.)

2.4 Explicit Scheme Specifications

For all schemes recommended in Table 3, in Appendix C we provide explicit references to their technical specifications. While in most cases these references relate to established standards and are thus normative, this is not the case for the submissions to the NIST competition as ongoing efforts connected to the latter will possibly result in minor tweaks of the schemes later in the process.

3 DAA Construction

The DAA protocol is one of the core functionalities of TPMs. The security of the DAA protocols that are standardized by the TCG relies on the hardness of either the DLP or the integer factorisation problem, which are known to be broken by Shor's polynomial-time quantum algorithm [42]. As a result, the DAA protocols that are currently in use are insecure when considering quantum adversaries and need to be replaced with ones whose security is based on mathematical problems that are hard to solve, even by adversaries in possession of a large-scale quantum computer. We thus describe a new compact quantum-safe lattice-based Direct Anonymous Attestation (LDAA) protocol of Chen, El Kassem, Lehmann and Lyubashevsky [11]. The security this LDAA scheme is proved in the Universal Composability (UC) model, under the assumed hardness of the Ring Short Integer Solution (Ring-SIS), the Ring Learning With Errors (Ring-LWE) and the NTRU problems. The main advantage of this LDAA scheme is that the signature size is estimated to be around 2MB, which is (at least) two orders of magnitude smaller compared to previously existing lattice-based DAA schemes [5, 32], which were discussed in D2.1. It is also expected to be faster in terms TPM computation cost in the **JOIN** and **SIGN** interfaces and has smaller keys.

In a DAA scheme, there are four main entities: a number of TPMs, a number of *Hosts*, an *Issuer* and a number of *Verifiers*. A TPM and a Host together form a *Platform* which performs the **JOIN** procedure with the Issuer who decides if the Platform is allowed to become a legitimate DAA signer or, in other words, a *member*. That is, the Issuer is responsible for managing a group of certified TPMs and in each new **JOIN** request, it verifies that the requesting Platform is equipped with a TPM that belongs in this group of certified TPMs. Once being a member, the TPM and the Host together can **SIGN** messages w.r.t. *basenames*, denoted bsn. If the Platform signs with $bsn = \perp$ or a fresh basename, the signature must be anonymous and unlinkable. That is, any Verifier can check that the signature stems from a legitimate Platform via a deterministic **VERIFY** algorithm, but the signature does not leak any information about the identity of the signer. If the Platform signs repeatedly with the same $bsn \neq \perp$, then it should be clear that the signatures were created by the same Platform. This is publicly tested via the deterministic **LINK** algorithm. DAA also supports *key-based revocation*, i.e. it assumes the availability of a revocation list (RL) which contains the secret keys of rogue TPMs. The verification will be done w.r.t. this revocation list in which case signatures of revoked TPMs fail. Intuitively, a DAA protocol must satisfy the following high-level security and privacy properties:

Anonymity: Given two signatures w.r.t. two different basenames or $bsn = \perp$, an adversary cannot distinguish whether they were created by one or two different honest Platforms.

(One-More) Unforgeability: When the Issuer is honest, an adversary controlling n TPMs can create at most n unlinkable signatures for the same $bsn \neq \perp$.

Non-Frameability: No adversary can create signatures on a message m w.r.t. bsn that links to a signature created by an honest Platform, when this Platform never signed m w.r.t. bsn.

We note that anonymity and non-frameability must hold even in the case of a corrupt Issuer.

Notation. Denote by $x \leftarrow S$ an element x being sampled uniformly at random from a set S. Define the polynomial ring $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$, for prime q, containing all polynomials of degree $d-1$ and coefficients in $\mathbb{Z}_q$, where d is the dimension of R_q, set as a power of 2. Denote by R_α, for some $\alpha \in \mathbb{Z}_{>0}$, a subset of R_q with polynomials whose coefficients are in the range $[-\alpha, \alpha]$. For some domain D, we define the domain extension function (e.g. SHAKE) as $H_D : \{0,1\}^* \rightarrow D$.

3.1 The New LDAA in a Nutshell

As mentioned in Deliverable D2.2 [14], DAA can be viewed as a special variant of group signatures with the Issuer controlling membership to the group of certified TPMs, and TPMs being able to sign anonymously on behalf of the group. More precisely, there are three main differences between DAA and group signatures, namely: (1) extra privacy properties are required in the case of a malicious Issuer, (2) the user in DAA is split into two parts, the TPM and the Host, and they do the signing in a way that does not reveal the TPMs secret (even to the Host) and (3) there is no opener, but there is instead a linking procedure that should allow anyone to link two signatures for a common basename (this property is known as *user-controlled linkability*).
The new LDAA uses the group signature scheme of Del Pino, Lyubashevsky and Seiler [16], which is currently the most efficient quantum-safe group signature scheme for large group sizes. This scheme is a modified version of the Ducas–Micciancio scheme [20], in which a signature $\mathbf{s}$ on a message $\mathbf{m}$ (both of small norm), w.r.t. a tag τ, is a polynomial satisfying the equation:

$$[\mathbf{A} \mid \mathbf{B} + \tau\mathbf{G}]\mathbf{s} = \mathbf{u} + \mathbf{a} \cdot \mathbf{m}, \tag{1}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{G}, \mathbf{u}$ and $\mathbf{a}$ are public parameters defined over some polynomial ring and τ is a fresh tag for every newly created signature. If the group member sends $\mathbf{v} = \mathbf{am}$ to the authority, along with a proof of knowledge of $\mathbf{m}$, then the authority can create a signature on $\mathbf{m}$ without its knowledge. In this way, the group signature scheme of [16] can be modified in order to satisfy a stronger definition of *dynamic group signatures*, in which a malicious issuer cannot impersonate a group member [11]. This procedure can be adopted in the case of DAA by assuming that $\mathbf{m}$ (or at least a part of it) is the TPM secret key and $\mathbf{s}$ is the secret membership credential of the Host. Both values, together with the tag τ, are used by the TPM and the Host, working together, for producing a Zero-Knowledge Proof of Knowledge (ZKPoK) of Eq. 1, in order to sign messages.
As mentioned before, another difference between DAA and group signatures is that DAA provides the user-controlled linkability property. That is, it should be possible, upon user's decision, to be able to link two or more signatures originating from the same TPM. In the new LDAA scheme, we consider the Ring-LWE instance $(bsn, nym = bsn \cdot m_1 + e)$, where m_1 is a part of the TPM secret and e is an error that is produced by the output of a Pseudorandom Function (PRF) when evaluated at bsn and m_1. In order to complete the signing process, the proof of knowledge of m_1 and e, satisfying $nym = bsn \cdot m_1 + e$ is added to the proof of Equation (1) [11].

3.2 Description of the new LDAA Protocol

Setup

The Issuer secret key is a matrix $isk = \mathbf{R} \in R_1^{2\times 2}$ and its public key is $ipk = (\mathbf{h}, \mathbf{b}, \mathbf{a}, u)$, consisting of $\mathbf{h} = [h\ 1] \in R_q^2$ for some uniformly-random polynomial $h \in R_q$, the vector $\mathbf{b} = \mathbf{h} \cdot \mathbf{R} \in R_q^2$, the public parameters $\mathbf{a} = [a_1\ a_2] \in R_q^2$, and $u \in R_q$. By the Ring-LWE assumption the pair $(h, \mathbf{b})$ is indistinguishable from uniform. We also assume that when the Issuer registers its public key with the Certification Authority (CA), it also proves the knowledge of its secret key in an extractable manner. Because the efficiency of this step is not very important, it can be performed using a standard Stern-type scheme (e.g. [36]). The TPM's secret consists of a polynomial vector $\mathbf{e} = [e_1\ e_2] \in R_3^2$ and a secret key $sk \in \{0,1\}^{256}$.[15]

[15] The reason that the polynomials e_1, e_2 are chosen to have coefficients larger than $\{-1, 0, 1\}$ is because the TPM will give out a lot of Ring-LWE samples with this secret and so the space of secrets needs to be a little bit larger to avoid the Arora-Ge attack [3].

The JOIN procedure

For DAA it is crucial that only legitimate TPMs can join in a controlled manner. This is handled via the endorsement keys which are preinstalled on each TPM and their public keys are known to the Issuer. We use the same abstraction as in Camenisch et al. [9] and assume an authenticated channel $\mathcal{F}^*_{\text{auth}}$ between the TPM and the Issuer, via the Host. That is, during the **JOIN** procedure, the Issuer learns the identity $\mathcal{M}_i$ of the TPM, in an authenticated manner, and should only proceed if $\mathcal{M}_i$ is a legitimate TPM. The Issuer should also keep track of the already joined TPMs and ensure that each can join at most once. The **JOIN** procedure is illustrated in Figure 1.

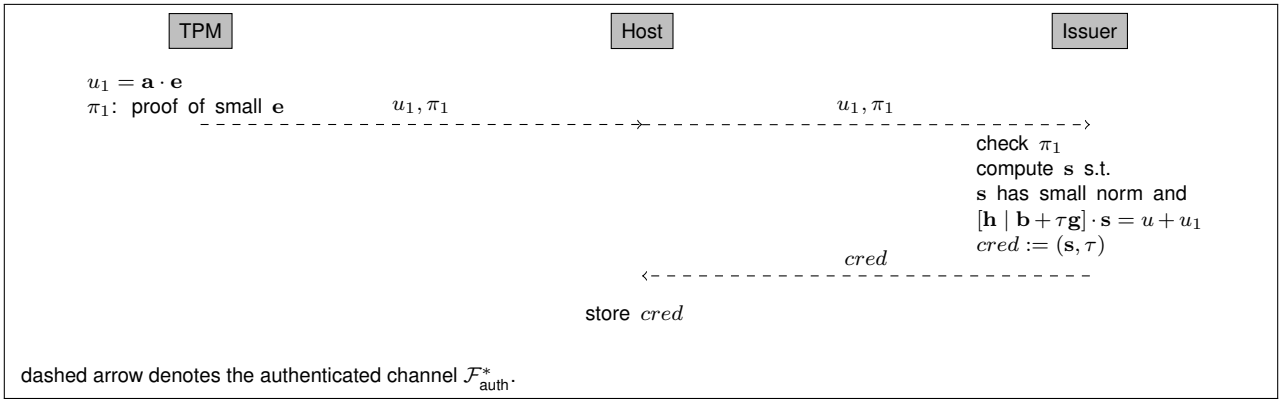

Figure 1: The **JOIN** procedure

The TPM computes $u_1 = \mathbf{a} \cdot \mathbf{e} = a_1 e_1 + a_2 e_2$ and sends u_1 along with a proof of knowledge π_1 of short $\mathbf{e}$, satisfying the equation for u_1, to the Issuer via $\mathcal{F}^*_{\text{auth}}$. We note here that since the TPM will do the **JOIN** process only once, it is not important for this proof to be very efficient and so it can be done using a ZKP system that proves *exact* knowledge of $\mathbf{e}$. The Issuer, upon receiving u_1, π_1, will check the proof and then use the Micciancio–Peikert [39] sampling algorithm to compute an $\mathbf{s}$ with small norm satisfying

$$[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}] \cdot \mathbf{s} = u + u_1, \tag{2}$$

where $\mathbf{g} = [1 \ \sqrt{q}] \in R_q^2$ and $\tau \in \mathbb{Z}_q$ is a fresh integer *tag*. The tag τ is initialized by the Issuer as $\tau = 1$ and it is incremented by one in every new execution of the **JOIN** procedure. Since in this protocol the prime q is relatively large (around 2^{70}), we get that τ is always in $\mathbb{Z}_q$ and hence every **JOIN** procedure will have a unique tag τ, which is crucial for security. The Issuer creates the credential $cred = (\mathbf{s}, \tau)$, and sends it to the Host via $\mathcal{F}^*_{\text{auth}}$, which stores the credential.

The SIGN procedure

In order to sign a message m w.r.t. a basename bsn (if $bsn = \perp$, the TPM picks a random bsn from the domain of basenames, but does not reveal it), the TPM creates a value $d = H_{R_q}(bsn)$ and an error polynomial $e' = H_{R_3}(sk, bsn)$ and outputs the pseudonym $nym = de_1 + e' \in R_q$. Notice that d is (and needs to be) publicly computable, while e' is generated deterministically based on bsn and sk, rather than chosen arbitrarily at random. This is because the TPM might be asked to create a pseudonym multiple times and it would be insecure to send $de_1 + e'_1, \ldots, de_1 + e'_k$ for different e'_k. Thus, for the TPM's safety, the same basename should lead to the same pseudonym. The Host commits to the values τ and $\tau\sqrt{q}$ as

$$\mathbf{Cr} + \begin{bmatrix} 0 \\ \tau \\ \tau\sqrt{q} \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} = \mathbf{t}, \tag{3}$$

where $\mathbf{C}$ is a public 3×4 matrix

$$\mathbf{C} = \begin{bmatrix} a_1 & a_2 & a_3 & 1 \\ b_1 & 0 & b_1' & 0 \\ b_2 & b_2' & 0 & 0 \end{bmatrix}$$

and $\mathbf{r} \in R_1^4$ is a polynomial vector with small coefficients. The precise description of this commitment scheme is given in [11, Section 2.5].
Let $\mathbf{C_1}, \mathbf{C_2}$ be the second and third row of $\mathbf{C}$. The Host and the TPM jointly compute a ZKP π (using the message m inside the Fiat–Shamir transformation) of small-normed $\overline{\mathbf{r}}, \overline{\mathbf{s}}, \overline{\mathbf{v_1}}, \overline{\mathbf{v_2}}, \overline{\mathbf{e}}$ and $\overline{c}$, satisfying the following equations:

$$\mathbf{C}\overline{\mathbf{r}} + \overline{c} \begin{bmatrix} 0 \\ \tau \\ \tau\sqrt{q} \end{bmatrix} = \overline{c} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} \quad \text{and} \quad \tau \in \mathbb{Z}_q \tag{4}$$

$$[\mathbf{h} \mid \mathbf{b} + [t_1\ t_2]]\overline{\mathbf{s}} - \mathbf{C_1}\overline{\mathbf{v}}_\mathbf{1} - \mathbf{C_2}\overline{\mathbf{v}}_\mathbf{2} - \mathbf{a} \cdot \overline{\mathbf{e}} = \overline{c}u \tag{5}$$

$$d\overline{e}_1 + \overline{e}' = \overline{c}nym \tag{6}$$

$$d\hat{e}_1 + \hat{e}' = 2nym \tag{7}$$

Equations (4), (5) and (6) are proved simultaneously (to ensure that the values $\overline{c}, \overline{e}_1$ are consistent throughout) and jointly by the TPM and the Host. In particular, Equation (4) is proved using the "automorphism stability" proof (see [11, Figure 3]) to ensure that $\tau \in \mathbb{Z}_q$. Equations (5) and (6) are proved via the standard "Fiat–Shamir with Aborts" technique that is described in [11, Figure 1]. The TPM needs to provide the additional proof π_1 of Equation (7), because having $\overline{c}$ in front of nym is not sufficient for linking, since one would actually have to know $\mathbf{c}$ in order to perform the linking operation. This proof is done according to [11, Figure 2].

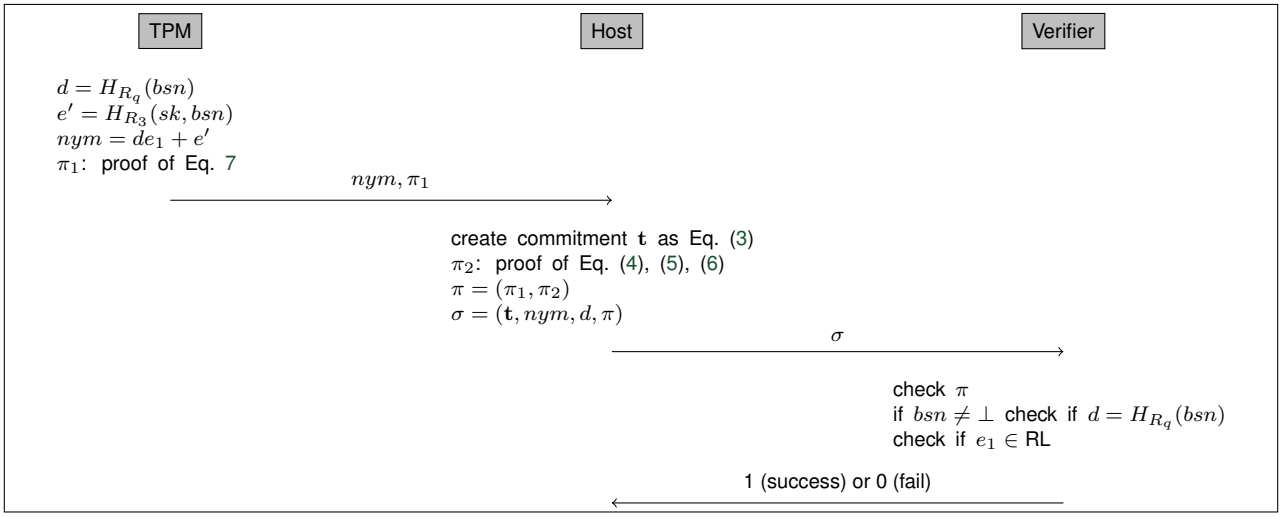

Figure 2: The **SIGN**/**VERIFY** procedures

The proofs of Equations (4) and (5) can be combined into one proof by multiplying Equation (4) by $\overline{c}$ and substituting $\overline{c}t_i$ with $\mathbf{C}_i\overline{r} + \overline{c}\tau$, which yields:

$$\overline{c}[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}]\overline{\mathbf{s}} = \overline{c}^2 u + \mathbf{a}' \cdot \overline{\mathbf{w}}, \tag{8}$$

where $\mathbf{a}' = [\mathbf{a}\ \mathbf{C}_1\ \mathbf{C_2}]$ and

$$\overline{\mathbf{w}} = \begin{bmatrix} \overline{\mathbf{e}}\overline{c} \\ \overline{c}\mathbf{v}_1 - \overline{\mathbf{r}} \\ \overline{c}\mathbf{v}_2 - \overline{\mathbf{r}} \end{bmatrix}.$$

The final signature output by the Host is $\sigma = (\mathbf{t}, nym, d, \pi)$, where $\pi = (\pi_1, \pi_2)$. The **SIGN** procedure is illustrated in Figure 2. By the analysis in [11], the total number of polynomials in R_q in the signature σ is 45, where 40 of these polynomials are in the proof π.

The VERIFY procedure

Given the message m, the basename bsn, the signature σ and the Issuer's public key ipk, the Verifier checks the proof π. If $bsn \neq \perp$, it also checks that $d = H_{R_q}(bsn)$. DAA also requires the verification process to be done w.r.t. the revocation list RL. Thus, for every $e_1 \in$ RL, the Verifier checks that $\|2(nym - de_1)\|$ is not small. If all checks pass, it outputs 1, otherwise it outputs 0. The **VERIFY** procedure is also depicted in Figure 2.

The LINK procedure

On input two triples (m_1, bsn, σ_1) and (m_2, bsn, σ_2) (with the same basename), the **LINK** algorithm checks whether both pseudonyms nym_1 and nym_2 in the signatures σ_1 and σ_2 match. We say that two signatures w.r.t. the same basename are linked to the same TPM, if $2(nym_1 - nym_2)$ is a polynomial in R_q with small norm.
To prove linkability, by Equation (7) we have:

$$2(nym_1 - nym_2) = d(\hat{e}_{1,1} - \hat{e}_{1,2}) + (\hat{e}'_1 - \hat{e}'_2),$$

where the left side has small norm and all coefficients except for d on the right side also have small norm. This implies that there exist polynomials f_1, f_2, such that $df_1 + f_2 = 0$ and one can show that for a random d, the probability that f_1, f_2 are non-zero is close to 0 (see [33, proof of Lemma 4.8]). Therefore, f_1 and f_2 must be both 0, which implies that $\hat{e}_{1,1} = \hat{e}_{1,2}$.

3.3 Security of LDAA Protocol

In Deliverable D2.2 [14] we pointed out that the security of the LDAA schemes is analyzed in the ROM, or in the QROM under slightly stronger (or less tight) assumptions. The reason for these stronger assumptions is that all the LDAA constructions have an underlying protocol whose classical security proof requires the reprogramming (or rewinding) of the random oracle that is accessed by the adversary which, in the strict sense, is generally not possible when considering quantum adversaries. However, the recent works [45, 33, 19, 37] show that there is nothing fundamentally insecure about the construction of any natural scheme built via the Fiat-Shamir framework whose security can be proven in the ROM. In our opinion, evidence is mounting that the distinction between schemes secure in the ROM and QROM will soon become treated in the same way as the distinction between schemes secure in the standard model and ROM - there will be some theoretical differences, but security in practice will be similar. In other words, a scheme that is proven secure in the ROM will still be fundamentally secure in the QROM.
We give a sketch of the security analysis and proof of the recommended LDAA scheme. For more details we refer to the original paper [11]. The authors show that under the Ring-LWE and Ring-SIS assumptions, proving knowledge of Equation (8) implies that $\overline{\mathbf{e}} = \mathbf{e}\bar{c}$, where $\mathbf{e} = [e_1\ e_2]$ is one of the TPM secret keys used during some **JOIN** session [11, Lemmas 3.1, 3.2]. In particular, Lemma 3.1 proves the indistinguishability of public keys. That is, one can construct a public key and a sampling algorithm, which are indistinguishable, under the Ring-LWE and NTRU assumptions, from the real public key and sampling algorithm. In addition, Lemma 3.2 proves

that every $\bar{e}_1$ extracted from the signer in Equation (6), must be equal to $e_1^*\bar{c}$, for some e_1^* that the authority created a credential on the **JOIN** procedure.
The security of this LDAA scheme is proved in the Universal Composability (UC) model. The proof follows the sequence of 16 games that was proposed by Camenisch, Drijvers and Lehmann in [9] and shows that there exists no environment $\mathcal{E}$ that can distinguish the real world protocol Π_{daa} with an adversary $\mathcal{A}$, from the ideal world $\mathcal{F}_{\text{daa}}^{l}$ with a simulator $\mathcal{S}$. Starting with the real world protocol game, the authors change the protocol, game by game in a computationally indistinguishable way, finally ending with the ideal world protocol.
In particular, the proof starts with the real world protocol execution. In the next game, one entity $\mathcal{C}$ is constructed that runs the real world protocol for all honest parties. Then the entity $\mathcal{C}$ is split into two pieces, a functionality $\mathcal{F}$ and a simulator $\mathcal{S}$, where $\mathcal{F}$ receives all inputs from honest parties and sends the outputs to honest parties. The proof starts with a useless functionality $\mathcal{F}$ and gradually change $\mathcal{F}$ and update $\mathcal{S}$ accordingly, to end up with the full $\mathcal{F}_{\text{daa}}^{l}$ and a satisfying simulator. Of the 16 games, the most crucial steps appear in Game 7, where the signature of honest Platforms are replaced by signatures on "dummy" keys (guaranteeing anonymity) and Games 12-15, where we let the functionality enforce the expected unforgeability and non-frameability properties. For unforgeability, we rely on the security of signatures created by the Issuer, which is proved in [11, Lemmas 3.1–3.3]. For the complete description of the 16 games we refer to the original paper [11, Section 4], where an advanced sketch of the security proof is provided. For a more detailed proof, we refer to El Kassem's PhD Thesis [21].

3.4 Comparison of LDAA Schemes

Recall that in the proposed compact LDAA scheme and following the instantiation of del Pino–Lyubashevsky–Seiler [16], the polynomial ring $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ has dimension $d = 4096$ and $q \approx 2^{70}$ is prime. On the other hand, in the previous LDAA schemes presented in [5] and [32], both the dimension d and the prime modulus q of the polynomial ring R_q are four times smaller, i.e. $d = 1024$ and $q \lessapprox 2^{18}$. Despite this difference in the ring size, the compact LDAA scheme of Chen, El Kassem, Lehmann and Lyubashevsky [11] should be faster in terms of the TPM computation cost in the **JOIN** and **SIGN** interfaces and, in addition, the new LDAA should have smaller TPM keys and signature sizes.
In particular, as pointed out in Section 3.2, the TPM secret consists of two polynomials in R_3, i.e. their coefficients are integers in the range $[-3, 3]$. On the contrary, the TPM secret in El Kassem et al. [32] consists of 24 polynomials in R_q, which in turn is half the size of the TPM secret in Bansarkhani–El Kaafarani [5] LDAA, based on the comparison in [32, Figure 3]. In addition, the signature of the compact LDAA consists of around 45 polynomials in R_q. According to [11], the instantiation $d = 4096$ and $q \approx 2^{70}$ implies that a rough estimate for the size of the signatures is 2MB. This is at least two orders of magnitude shorter that the signature size in El Kassem et al. [32] (which, based on [32, Figure 4], is half the size of the signatures in Bansarkhani–El Kaafarani [5] LDAA).
In conclusion, the compact LDAA scheme presented by Chen, El Kassem, Lehmann and Lyubashevsky [11] is believed to provide a significant improvement in terms of efficiency, as well as storage requirements. It should be pointed out that despite the reduced LDAA signature size, this is still 5 to 6 orders of magnitude longer than the signatures produced in the (non-quantum resistant) discrete logarithm based DAA scheme [8] of Camenisch et al., which is currently implemented in TPM2.0.

4 Models

In Section 4.1 we revive and refine a discussion from Deliverable D2.1 on general approaches towards defining the security of cryptographic primitives in a world that assumes the existence of quantum computers. Then, in Section 4.2, we discuss security issues that might emerge when using a cryptographic scheme even if the latter is provably secure (in the classic or quantum setting); this discussion is particularly focused on the TPM setting.

4.1 Security Models in the Quantum World

Generally speaking, any positive or negative statement about the security of a cryptographic scheme is explicitly or implicitly formulated with respect to a security model. A security model is a formal description of how different actors and entities interact with the scheme and each other, and what the goals of an attacking adversary are. For instance, typical models for a digital signature scheme formally require (1) that the signing/verification key pair is generated honestly (i.e., according to specification); (2) that the verification key used by any verifier is an authentic copy of the original one; (3) that the adversary can (partially or totally) influence the messages that are signed; and (4) that the adversary is deemed successful in their attack if they present *any* correctly-verifying not-replayed message–signature pair (independently of whether the forged message makes sense or not in the context of a specific application). Starting with the advent of the provable security approach in the mid 1980s, a canonic set of such security models has been proposed for all types of primitives (signature schemes, public key encryption, block ciphers, etc.). While, in principle, for each primitive a large number of models are possible, the ones in the canonic set can be characterized by meeting just the right balance between being useful and being achievable. Here, we say that a model is *useful* if the security guarantees that it establishes for the tested primitive are logically sufficient to imply the security of higher-level protocols constructed from it, while we say that a model is *achievable* if for the primitive effective secure constructions are feasible that can be operated in realistic settings.

Note that, while the usefulness of a specific security model is independent of the computational power of the considered adversary (e.g., whether it is a classic computer, a quantum computer, or is computationally unbounded), this does not hold for the achievability of the model. Indeed, the fact that secure signature schemes based on the hardness of the RSA or DLP problems, which are achievable against classic attackers, are not achievable against quantum attackers, shows that the set of feasible constructions for a scheme crucially depends on the computational power of the adversary.

The intuitive conclusion of the above is that the general security models established for cryptographic primitives do not have to be revisited with respect to their usefulness against quantum attackers, while only the potential constructions have to be. However, as we expand on in the following, the situation becomes a little different if the adversary can also interact with attacked schemes in quantum super-position.

In Section 2.3 of Deliverable D2.1 we distinguished between the three attack settings QS0, QS1, and QS2. Here, the term QS0 corresponds with the standard attack setting that assumes classic adversaries and the term QS1 corresponds with the the attack setting in which a classic scheme is attacked by a quantum adversary. In the QS2 setting, a quantum adversary attacks a quantum-implementation of a classic scheme. For instance, let us take the AES block cipher as an example of a classic block cipher, i.e., a block cipher that was designed for implementation in classic hardware or software. While the QS1 setting considers attacks of a quantum adversary against

such a classic implementation of AES, the QS2 setting considers that the targeted AES instance is actually not operated on a classic computer but instead on a quantum computer. The crucial assumption in the QS2 setting is that the honest actors evaluate the attacked AES instance in quantum super-position, that is, simultaneously on many different inputs, and make the result of such evaluations accessible to the adversary. The QS2 attack model is considerably stronger than the QS1 attack model, and it is correspondingly harder to achieve.

Which attack setting, QS1 or QS2, should be considered the right one in a future TPM, i.e., recommended by this deliverable? In Section 2.3 of D2.1 we defined both QS1 and QS2 and expressed a tendency towards using the QS2 setting as our target. In Deliverable D2.2, in which we recommended a series of concrete instantiations for the different cryptographic building blocks, we gave, where applicable, analyses in both the QS1 and QS2 models, allowing for a comparison. However, we also changed the tendency of our recommendations towards the QS1 model. In the current deliverable D2.3 we continue to support QS1 as the right model for the use in the FutureTPM project. We do this because the costs involved with achieving QS2 security do not seem justifiable given the little relevance of the offered additional security in specifically the TPM setting: The QS2 setting assumes that scheme algorithms are implemented deliberately or accidentally on a quantum computer, but in practice hardware TPMs are produced using technology that is clearly fully classic and not capable of performing quantum computations, and software TPMs are programmed for use in pre-specified CPUs, all of which are fully classic as well. Unless explicitly noted otherwise, all analyses in the different sections of this deliverable are hence in the QS1 setting.

4.2 Attacks outside the Model: Implementational Security

Practice has shown that even if a provably secure scheme is deployed in some application, the result might still be insecure. In line with the observation that security models for cryptographic primitives typically consider the security of the primitives only when they are implemented and operated according to specification, in most cases the emerging security issues can be traced down to implementations not fully following the specification, but unintentionally deviating from it (e.g., by generating side-channel leakage that the adversary can exploit). Partially overlapping with a similar list in Deliverable D2.1, in the following we discuss some important forms of how implementations can unexpectedly deviate from their specification. While we caution that this list is by no means complete, we believe it is a good starting point for studying what to keep in mind when developing a real-world TPM implementation.

4.2.1 Leakage Resilience

Side-Channel Attacks (SCAs) attempt to break an implemented cryptosystem through the physical information that can be observed when executing it [43]. Typical secondary information channels include power consumption and execution timing, which might be interpreted as a noisy function of secret values. Two main approaches have been considered in the literature to protect cryptographic implementations against SCAs. A first approach deals with local countermeasures: Hiding techniques aim at decreasing the Signal-to-Noise Ratio (SNR) in order to hide the information leakage among the random noise; and masking countermeasures use secret sharing and multi-party computation to randomize intermediate values, and reduce interdependencies. This first approach is limited, since no solution has up to now been able to completely get rid of leakages, and should therefore be complemented with more global approaches.

In a global approach setting, one assumes that a single iteration of a cryptographic primitive leaks a certain amount of information, but tries to achieve security after computing that primitive multiple times nonetheless [44]. Generally, the building blocks used in these constructions use subkeys, and techniques are applied so that an adversary can only observe the encryption of a maximum number of different plaintexts under a certain subkey. System designers will then limit the success rate of the best available adversary for that number of queries, by limiting the information leakage.

Protection against physical attacks should be taken into account when developing a future TPM, since otherwise even non-invasive attacks, such as those based on Electromagnetic Radiation and Timings may lead to a complete breakage of the system.

4.2.2 Fault Attacks

Fault Attacks (FAs) are active attacks against implementations wherein an attacker is able to forcibly change the status of a device, leading e.g. to the flipping of bits residing in memory or in a processor, to infer secret information from the faulty results, or to circumvent authentication procedures [31]. This type of attack may be achieved through non-invasive means: Attacks such as RowHammer [34] have shown that it is possible to change bits in computer memory by changing the bits of neighboring memory cells at a quick pace in software; changing the circuit's environment (e.g., by changing the clock's frequency or the operating temperature) may also lead to changes in the program and data flow. More expensive equipment may be used to implement semi-invasive and invasive attacks, like optical or electromagnetic faults, to change the device's state in a more focused manner.

The most popular types of fault attacks include Differential Fault Analysis (DFA), where a cryptographic computation is performed first without any fault, and a second time with the same input but with a fault, and private key material is inferred from the output differential; and Collision Fault Attacks (CFAs) where an adversary obtains first a faulty output of a cryptographic computation, and afterwards exhaustively searches for the input that produces a similar output when no fault is inserted, deriving key material from the difference between the two executions.

Regarding protecting against them, FAs may either be detected or prevented. Detection techniques include circuits that detect voltage glitches and laser detectors. Prevention techniques include adding redundant hardware or software modules and detecting differences between them or using Error Correcting Codes (ECCs) for cryptographic linear operations or during memory decoding. The implementation of FA mitigation techniques should be done carefully: For instance, the replication of hardware modules may improve the SNR of an SCA attacker (see Section 4.2.1). Furthermore, FAs may be combined with SCAs to produce more powerful attacks.

While it is impossible to fully protect against FAs, and certain attacks, such as the RowHammer attack [34], might not be applicable in the restricted setting of a TPM, it might be useful to take possible FA countermeasures into consideration when designing a future TPM system, so as to prevent private key material from being disclosed to an attacker, who could afterwards impersonate the attacked chip.

4.2.3 Algorithm Substitution Attacks

A common property of the above-described attacks leveraging on side-channel leakage or fault injection is that the attack target is a benign implementation of the scheme (though with insufficient care taken when developed). Further, to conduct the attacks, in most cases physical proximity

to the victim is required. This is in contrast with Algorithm Substitution Attacks (ASAs, [7]) which complement the scene for assuming malicious implementations but not requiring physical proximity to the target. In an ASA, the party commissioned with implementing the scheme consciously deviates from the specification, embedding some kind of backdoor into the implementation, the knowledge of which crucially reduces the security of the scheme. The shattering power of an ASA lies in the fact that the observable behavior of the manipulated implementation is precisely the same as that of a benign implementation, that is, the malicious deviation cannot be noticed during regular operation.

As Algorithm Substitution Attacks (ASAs) have been proposed against virtually all types of cryptographic primitives, including symmetric encryption [7, 15], authenticated encryption [2], message authentication schemes [1], hash functions [23], public key encryption [12], signature schemes [4], and Direct Anonymous Attestation (DAA) [10], all of which are relevant in the TPM context, we caution that the impact of ASAs in the TPM setting should not be underestimated.

5 Hybrid solutions

Very briefly, the aim of this work package of the FutureTPM project is to identify cryptographic primitives that lead to safe TPM solutions in the presence of attackers that have access to quantum computers. Many established primitives, in particular those based on classic hardness assumptions like RSA or DLP, are weak in this setting and cannot be used. In Deliverable D2.2 and Section 2 of the current report, we identified a number of alternative constructions that are based on non-classic hardness assumptions related to finding shortest vectors in lattices, decoding random codes, and so on. While these constructions, and the hardness assumptions they rely upon, have been extensively studied by researchers and, so far, no flaws have been identified, it is also true that all these assumptions appeared considerably more recently than those connected to RSA and DLP. We conclude that the alternative assumptions should be considered less tested, and a successful break with non-quantum methods could be expected to emerge with higher probability than an attack against RSA or DLP. Further, the construction of quantum computers is still in its infancy and so far only few cryptanalytic results leverage on them.
The current section of Deliverable D2.3 is dedicated to evaluating options of combining classic and recently emerging construction approaches at a primitive level. The goal is to reach a maximum of security against both classic and quantum attackers. For instance, we study how two input signature schemes, one assumed classically secure and one assumed secure against quantum attackers, can be combined into a hybrid signature scheme that (a) is classically secure even if the second input scheme eventually gets broken, and (b) is quantum resilient as long as the second input scheme withstands quantum attacks. That is, such a combined signature scheme inherits the best properties of both of its components. Besides signature schemes, we also consider various other kinds of cryptographic primitive. We treat them in the order as they appear in D2.2.

5.1 Hash Functions and Block Ciphers

Sections 2 and 3 of D2.2 studied the fitness of hash functions and block ciphers for the use in quantum-resilient TPMs. A list of recommendations was derived, where the choice was mainly made according to the parameter sizes of the primitives. For instance, block ciphers were selected according to their key and block sizes, and hash functions according to their block and digest sizes. (See Deliverable D2.1 for the rationale behind the selection.) Beyond that, a further selection criteria was the status of acceptance of the primitive by the TCG and international standardization bodies.

5.1.1 Hybrid Hash Functions

See Table 3 (on page 6 of the current document) for the list of recommended hash functions. We note that these hash functions (e.g., the members of the SHA2, SHA3, and BLAKE2b families), despite being classic designs, i.e., despite having been designed to be secure against classic attackers that do not have access to a quantum computer, are generally assumed to be secure in both the classic and the quantum setting. (See the corresponding discussion in Deliverable D2.2 and note the stricter requirements on parameter sizes in the quantum setting.) Thus, we believe that studying options for hybrid hash functions (that securely combine two hash functions into a single hash function) is not strictly necessary from a security perspective. For the case that

concerns remain, we nevertheless provide a brief discussion of how any two hash functions can be combined to a secure hybrid.

Let H_1 and H_2 denote two cryptographic hash functions. The security goals that one typically associates with a cryptographic function are pre-image resistance (one-wayness), second pre-image resistance, and collision resistance. It turns out that the method that is optimal to combine H_1 and H_2 into a hybrid hash function H depends on the specific security goal that is pursued. For instance, sequentially composing H_1 and H_2 in the form $H = (H_2 \circ H_1); x \mapsto H_2(H_1(x))$ leads to a pre-image resistant (respectively, second pre-image resistant) hash function H if at least one of H_1 and H_2 is pre-image resistant (resp., second pre-image resistant).[16] However, the sequential combiner does *not* yield a secure hybrid if the security goal is collision resistance. Indeed, if distinct x, x' are known that collide under H_1, i.e., that have the property $H_1(x) = H_1(x')$, then also $H(x) = H_2(H_1(x)) = H_2(H_1(x')) = H(x')$, i.e., the pair x, x' also manifests a colliding pair for H.

A different approach to combine H_1, H_2 to a hybrid would be to operate the two hash functions in parallel and concatenating the results. This would be formalized by defining $H = (H_1 \mid\mid H_2); x \mapsto (H_1(x), H_2(x))$. Note that this function preserves collision resistance: If distinct x, x' exist with $H(x) = H(x')$ then x, x' also constitute a colliding pair for simultaneously both H_1 and H_2. With other words: If at least one of H_1 and H_2 is collision resistant, then also H is. On the other hand, the parallel combiner fails to preserve the other properties: For instance, assume H_1 is the identify function while H_2 is pre-image resistant, i.e., for known $y = H_2(x)$ it is hard to find x, then from $y = (H_1 \mid\mid H_2)(x) = (H_1(x), H_2(x))$ it is immediate to recover x (by outputting a prefix of y), thus breaking pre-image resistance.

A line of work that studies combiners for hash functions that preserve all security properties simultaneously (i.e., that are secure with respect to security notion N if at least one of H_1, H_2 provides security according to notion N), was conducted in [24, 25, 26, 35]. A specifically practical setting was considered in [27]. Note that [25, 26] also study how two hash functions can be combined such that, if one hash function is assumed to behave like a random oracle, then also the hybrid is (indifferentiable from) a random oracle.

5.1.2 Hybrid Block Ciphers

See Table 3 (on page 6) for the list of recommended block ciphers. Similarly to the hash functions discussed above, the recommended block ciphers AES, Rijndael, and Camellia are expected to withstand quantum attackers on their own. (Conditions on the key and block sizes apply, see D2.2.) That is, strictly speaking, it does not seem to be required to seek for hybrid solutions that combine any two block ciphers into a single block cipher. For the case that concerns remain, we briefly discuss existing results that consider the combination of two block ciphers into a secure hybrid.

Let E_1, E_2 be two block ciphers with compatible domain (i.e., both with with a blocksize of 256 bit). Fairly obvious, the parallel composition $E = (E_1 \mid\mid E_2)$ such that $((K_1, K_2), (x_1, x_2)) \mapsto (E_1(K_1, x_1), E_2(K_2, x_2))$, where K_1, K_2 are independent keys, does not require much further discussion: Whatever the security goal under consideration is, if for instance E_1 is not a secure cipher but the identity mapping, then also $E_1 \mid\mid E_2$ will not be a secure cipher. Consider hence the sequential composition and let $E = (E_2 \circ E_1); (K_1, K_2, x) \mapsto E_2(K_2, E_1(K_1, x))$. This composition is also known as (two-key) double-enciphering. One can easily show that if E_1 or E_2 is a

[16] This is more an intuitive observation than a formal statement. For instance, if H_1 is a constant function then the statement is formally incorrect. See [41] for a set of possible additional requirements on H_1, H_2 that are required for the statement to hold formally.

secure block cipher then also their composition E is. More formally, if at least one of E_1 and E_2 is secure as a pseudo-random permutation (PRP), then also $E = E_2 \circ E_1$ is secure as a pseudo-random permutation. Similarly, if at least one of E_1 and E_2 is secure as a strong pseudo-random permutation (SPRP), then also $E = E_2 \circ E_1$ is secure as a strong pseudo-random permutation. Note that these results are sufficient to securely instantiate all block cipher based modes of operation recommended in D2.2 and the current document (including CMAC, GCM, etc.).
Sequential composition is a well-established technique in the domain of block ciphers.[17] However, historically speaking, the goal of proposing such a composition was not to obtain hybrids that specifically withstand quantum attacks. Rather, the goal was to strengthen ciphers against exhaustive key enumeration attacks (a.k.a. brute force attacks). The most prominent example is likely the lifting of DES, which natively only supports 56 bit keys, to a block cipher with an effective key size of more than 100 bits. It was found that the sequential composition of only two block ciphers is insufficient to reach this goal: A class of meet-in-the-middle attacks (MITM) allows the independent enumeration of the keys of the two ciphers and later align the results. However, MITM attacks are not an issue in our hybrid setting, and they also don't contradict the above result on PRPs and SPRPs.

5.2 Symmetric Modes of Operation

5.2.1 Hybrid Message Authentication Codes

See Table 3 (on page 6) for the list of recommended message authentication codes (MACs): HMAC, KMAC, and CMAC. These represent a diverse selection of candidates, as HMAC is internally constructed from a Merkle–Damgard (MD) hash function, KMAC is internally constructed from the Keccak sponge, and CMAC is constructed generically from any block cipher.
As it was the case above for hash functions and block ciphers, it is not expected that MACs become weaker in the presence of quantum adversaries (if certain restrictions on key sizes etc. are met), that is, strictly speaking, it does not seem a priority to employ a hybrid MAC construction. However, if a hybrid construction is sought for, there are two distinct general approaches: (1) One can stick to one of the three MAC modes and strengthen the underlying building block. For instance, and more concretely, one could choose to operate CMAC and instantiate its block cipher with a hybrid of two block ciphers, e.g. AES and Camellia. Or, (2) one can combine two different MAC modes, instantiated individually with their respective building blocks, to obtain a single overall MAC. Here, and in contrast to the approach described above for block ciphers, the sequential composition is not secure while the parallel composition is. Concretely, let M_1, M_2 be two MAC schemes and consider first their sequential composition $M = (M_2 \circ M_1)$ defined as per $(K_1, K_2, m) \mapsto M_2(K_2, M_1(K_1, m))$. Now assume that M_1 is insecure for outputting a constant input-independent tag, while M_2 is secure. Then any tag produced by M for a message m can be replayed with any other message m', resulting in a successful forgery. That is, the sequential composition is not secure. Consider hence the parallel composition $M = (M_1 \mid\mid M_2)$ defined as per $(K_1, K_2, m) \mapsto (M_1(K_1, m), M_2(K_2, m))$. For this MAC it is easy to verify that if either M_1 or M_2 is unforgeable, then also M is unforgeable. That is, the parallel composition is secure.
We note that many practical MACs not only provide unforgeability, but they can also be deployed as pseudo-random functions (PRFs). This holds in particular for the functions HMAC, KMAC, and CMAC that we recommended. While the parallel combiner is secure with respect to unforgeability,

[17] See, for instance, https://en.wikipedia.org/wiki/Triple_DES#Algorithm. For corresponding academic works, see [17, 38].

it is not secure with respect to pseudo-randomness (PR). Indeed, assume M_1 is a pseudo-random function (i.e., secure) while M_2 is constant (i.e., insecure). Then the right component of the output of $M = (M_1 \mid\mid M_2)$ is constant as well, and M thus not pseudo-random. We thus propose a third combiner that preserves pseudo-randomness, at the cost of requiring that the MAC tags have consistent lengths.[18] Concretely, define the XOR combiner $M = (M_1 \oplus M_2)$ of M_1, M_2 as per $(K_1, K_2, m) \mapsto M_1(K_1, m) \oplus M_2(K_2, m)$. It is easy to verify that this combiner preserves pseudo-randomness, i.e., the construction serves as a secure PRF if at least one of M_1 and M_2 is a secure PRF.

5.2.2 Hybrid Symmetric Encryption

See Table 3 (on page 6) for the list of recommended symmetric encryption modes: CFB, CBC, and CTR. As for MACs, hybrids for symmetric encryption can be defined either (1) by using one of the three modes without modification but instantiating them with hybrid block ciphers, or (2) by constructing hybrid encryption modes generically. The first approach was already covered in Section 5.1.2. We provide details about the second approach. Let Enc_1 and Enc_2 denote two encryption functions that take a key and a message and output a ciphertext from which the message can be recovered using some decryption algorithm. Define the sequential composition $\mathsf{Enc} = (\mathsf{Enc}_2 \circ \mathsf{Enc}_1)$ of $\mathsf{Enc}_1, \mathsf{Enc}_2$ as per $(K_1, K_2, m) \mapsto \mathsf{Enc}_2(K_2, \mathsf{Enc}_1(K_1, m))$ [22]. Then, in attack settings that consider passive attackers, this combiner preserves confidentiality. Formally, if one of Enc_1 and Enc_2 provides confidentiality in the sense of indistinguishability against chosen-plaintext attacks (IND-CPA), then also Enc provides confidentiality in the IND-CPA sense. (The analogous result for parallel composition does not hold for reasons similar to those above.)
We caution that sequential composition is not secure in the face of active adversaries that can manipulate ciphertexts while they are transported from the sender to the receiver. Formally, even if one of Enc_1 and Enc_2 provides confidentiality in the sense of indistinguishability against chosen-ciphertext attacks (IND-CCA), then Enc does not necessarily provide confidentiality in the IND-CCA sense. (To see this, consider the case where scheme Enc_2 is malleable.) This in fact does not suggest a limitation of the sequential combiner when applied to the CFB, CBC, and CTR schemes as, taken for themselves, they provide only IND-CPA but not IND-CCA security.

5.2.3 Hybrid Authenticated Encryption

Authenticated encryption (AE) is a form of symmetric encryption that meets the security goals of integrity and confidentiality protection simultaneously. In principle, the two goals can be achieved by first symmetrically encrypting the message and then authenticating the ciphertext using a MAC (see Sections 5.2.2 and 5.2.1, respectively). However, as discussed in Deliverable D2.2, integrating the two primitives into a single object proved to be more robust and, more importantly, demonstrated higher degrees of efficiency. In particular, most integrated AE modes get along with a single key instead of two independent keys, and they often can achieve the two security goals using the same shared internal building blocks (e.g., the same block cipher).
See Table 3 (on page 6) for the list of four recommended AE modes. While the modes CCM, GCM, and EAX are based on a block cipher, the mode Ascon is sponge-based. Similarly to what we discussed above, one way to obtain a hybrid AE mode is by employing CCM, GCM, or EAX without modification but instantiating it with a hybrid block cipher as per Section 5.1.2. The

[18] This requirement is in fact mild: If the MAC tags happen to have different lengths, the longer one can be truncated to the length of the shorter one.

second approach, which is considerably more powerful and flexible, is to generically combine two AE schemes into one, independently of how they are instantiated. We discuss the second approach in the following.

Recall from Sections 5.1.2 and 5.2.2 that the right way to combine confidentiality primitives is by sequential composition (while parallel composition is not secure). The contrary is the case for authentication primitives: Recall from Section 5.2.1 that the right way to combine them is by parallel composition (while sequential composition is not secure). This makes the AE case particularly challenging, as it falls into both classes and thus the standard combining methods don't seem to apply. In fact, the challenge of securely combining AE schemes was resolved only by dedicated research conducted within the FutureTPM project. In the following we provide an easily accessible version of the results of the corresponding publication [40].

Let $\mathsf{Enc}_1, \mathsf{Enc}_2$ be the encryption algorithms of two AE modes, that is, for both $i \in \{1, 2\}$ we assume that algorithm Enc_i takes a key K_i and a message m_i, and outputs a ciphertext c_i.[19] Let $\mathsf{Dec}_1, \mathsf{Dec}_2$ be the corresponding decryption algorithms. The proposal of [40] to securely combine $\mathsf{Enc}_1, \mathsf{Enc}_2$ into a single AE mode is via a *sequential* encryption that is matched by a *nested* decryption. Concretely, while a message m is encrypted under key $K = (K_1, K_2)$ to ciphertext $c \leftarrow \mathsf{Enc}_2(K_2, \mathsf{Enc}_1(K_1, m))$, the decryption is more involved: First ciphertext c is decrypted under key K_2 to $c_0 \leftarrow \mathsf{Dec}_2(K_2, c)$, then c_0 is re-encrypted to $c' \leftarrow \mathsf{Enc}_2(K_2, c_0)$, then, if $c' = c$, c_0 is decrypted as per $m \leftarrow \mathsf{Dec}_1(K_1, c_0)$. The decryption algorithm aborts (outputting an error indicator) if either $c' \neq c$ or one of $\mathsf{Dec}_1, \mathsf{Dec}_2$ aborts with an error. It is proved in [40] that the decrypt-encrypt-decrypt approach of the combined decryption indeed simultaneously provides authentication and confidentiality. For more details, including the formal security argument, see [40, Sec. 3].

Note that the decryption algorithm of the AE combiner that we just described internally invokes both the encryption and decryption algorithms of the second ingredient scheme. Intuitively, this seems to imply an avoidable overhead. However, in [40, Sec. 5] it is proved that this approach is effectively optimal, i.e., that simpler combiners do not exist. More precisely, [40] argues that optimal generic combiners necessarily follow the described structure, while non-generic combiners (i.e., those that exploit internal properties of the input AE schemes) may get along with less. Indeed, for the non-generic case, a number of more efficient combiner constructions is proposed in [40].

5.3 Asymmetric Schemes

5.3.1 Hybrid Public Key Encryption

See Table 3 (on page 6) for the list of recommended public-key encryption (PKE) schemes. The four schemes are internally built from a number-theoretic component that provides some kind of trapdoor functionality: Assuming that a public-secret key pair was established a priori, on input the public key this component generates a ciphertext and corresponding secret value, where the latter can be recovered from the former using a secret key as a trapdoor. Taken in isolation, this trapdoor primitive does not provide the targeted security properties of indistinguishability against chosen-plaintext attacks or chosen-ciphertext attacks (IND-CPA and IND-CCA, respectively). Rather, IND-CCA security is obtained from the primitive by applying a generic transformation to it. Such transformations are in the spirit of the Fujisaki–Okamoto transform (FO, [28]) and

[19] Some authors assume that an AE encryption algorithm also outputs a value referred to as the tag. Our notation is more generic by assuming that, if a tag value exists, then it is a suffix of the ciphertext.

the recent improvements and re-analyses of [30]. We note that the four PKE schemes indicated in Table 3 use slightly different such transformations.

Analogously to the techniques discussed in Sections 5.2.1, 5.2.2, and 5.2.3, there are two main approaches for combining two PKE schemes into one. The first approach would be to securely combine the number-theoretic primitives underlying the PKE schemes, and to then apply a common FO-like transform. While this, if applicable, would result in a very efficient combined scheme, we suggest avoiding this approach as the specific FO-like transforms employed by the four recommended PKE schemes are carefully adjusted to their particular peculiarities. For instance, certain primitives react sensitively to decryption errors that emerge with non-negligible probability, and the correspondingly chosen and fine-tuned FO-like transforms take such issues into account. That is, first combining the number-theoretic primitives and then strengthening them with an independent FO-like transform seems questionable. The second approach is to employ a generic PKE combiner that does not try to deconstruct the input PKE schemes. The following paragraphs are dedicated to discuss the second approach.

A notion closely related to that of public-key encryption (PKE) is that of a key encapsulation mechanism (KEM). In contrast to a PKE scheme, the goal of a KEM is not to encrypt a message to another party, but instead to establish a secure session key with it. If this key is used with an AE scheme (see Section 5.2.3) to encrypt a message, the overall result is precisely a PKE scheme. As also encrypting a random session key with a PKE scheme results in a KEM, the two concepts (PKE and KEM) are effectively equivalent. While combiners for PKE have been proposed earlier (e.g., in [18]), it is made evident in [29] that building KEM combiners is much more direct and efficient.

The work of [29] proposes a variety of KEM combiner constructions. Here we reproduce only one of them. Let $\mathsf{Enc}_1, \mathsf{Enc}_2$ be two KEM encapsulation algorithms, and let pk_1, pk_2 be two corresponding public keys. Invoking $(c_1, K_1) \leftarrow \mathsf{Enc}_1(pk_1)$ and $(c_2, K_2) \leftarrow \mathsf{Enc}_2(pk_2)$ establishes the two session keys K_1 and K_2 that can be recovered by the receiver by invoking $K_1 \leftarrow \mathsf{Dec}_1(sk_1, c_1)$ and $K_2 \leftarrow \mathsf{Dec}_2(sk_2, c_2)$ (where the sk_i are the secret keys corresponding to the pk_i). If F is an auxiliary pseudo-random function (PRF), then, by the results of [29], the key K computed as per $K = F(K_1, c_2) \oplus F(K_2, c_1)$ is an IND-CCA secure session key of the combined KEM. (In words: The PRF is invoked once with each session key and on input the ciphertext of the *other* KEM instance, and then the results are XORed together.) While this construction is compact and elegant, we note that it is actually the least efficient one presented in [29].

Observe that the suggested KEM combiner requires a PRF as a building block. It might be attractive to instantiate this component with a PRF derived as a hybrid from two ingredient PRFs. Fortunately, in Section 5.2.1 we already discussed how to combine PRFs in a quantum-resilient way.

5.3.2 Hybrid Signature Schemes

See Table 3 (on page 6) for the list of recommended signature schemes. Like discussed above for public-key encryption, a majority of the recommended signature schemes depend on rather recent and untested hardness assumptions that might eventually be broken, either by a classic or a quantum computer. Here we discuss techniques for combining two signature schemes generically, i.e., independently of their underlying assumptions.

A classic approach to combining two signature schemes into one is by signing the message with the two signature schemes independently, and to concatenate the signatures. Specifically, if (sk_1, vk_1) and (sk_2, vk_2) are key pairs for signature schemes, and the signing and verification algorithms are denoted Sig and Vfy respectively, then to sign a message m compute

$\sigma_1 \leftarrow \text{Sig}_1(sk_1, m)$ and $\sigma_2 \leftarrow \text{Sig}_2(sk_2, m)$ and output $\sigma = \sigma_1 \mid\mid \sigma_2$. To verify such a signature, parse σ back to σ_1 and σ_2, and accept if both $\text{Vfy}_1(vk_1, m, \sigma_1)$ and $\text{Vfy}_2(vk_2, m, \sigma_2)$ evaluate positively. The most important security notion for signature schemes is existential unforgeability under chosen-message attacks (EUF), and we confirm the concatenation combiner achieves it. The situation regarding the strong unforgeability (SUF) notion is more involved, and unpublished research results by the authors of this report indicate that no secure combiner exists that wouldn't rely on a third (trusted) signature scheme as a building block.

6 Summary and Conclusion

The main contributions of this report can be found in Sections 2–5. The four sections are independent and cover the following topics:

- Section 2 completes our recommendations of cryptographic primitives and schemes to be used in a future TPM. (Recall we had similar sections in Deliverables D2.1 and D2.2.) While our recommendations with respect to symmetric schemes remained mostly stable across the three deliverables, we slightly adapted our recommendations with respect to asymmetric schemes over time. As documented in Section 2, this is mostly due to two reasons: (a) The impact of quantum attackers on classic asymmetric assumptions like RSA and ECC is considerably bigger than the impact on symmetric primitives; and (b) we align our recommendations with the ongoing efforts of the NIST to standardize asymmetric quantum-resilient schemes.

- Section 3 is dedicated to developing a novel efficient DAA protocol that is based exclusively on quantum-safe assumptions. Developing such a protocol was necessary as most prior DAA solutions, in particular the ones standardized by the TCG, relied on RSA or ECC related assumptions. Our new protocol is proved secure in the UC framework and thus guaranteed to be a good replacement for prior DAA protocols. The relevant assumptions underlying our protocol are all lattice based, namely Ring-SIS, Ring-LWE, and NTRU.

- Section 4 contributes a general discussion on security models that are adequate for cryptographic primitives in the TPM setting. Besides the new quantum threats that need to be taken into account, and are discussed in Section 4.1, in Section 4.2 we acknowledge that TPMs are often implemented in physically approachable devices, meaning that side-channel attacks (SCAs) and fault-injection attacks (FAs) are applicability with increased likelihood, and need to be taken into account in implementations.

- Section 5 is dedicated to discussing generic hybrid methods that allow combining classic cryptographic primitives with quantum-resilient ones such that a maximum of robustness against failures is reached: Even if one of the ingredient schemes fails, the security of the hybrid is maintained. While we report on such combiners for all the primitives considered in Section 2, we note that combiners of asymmetric primitives (public key encryption and signature schemes) seem to be particularly attractive in our project, as they might smoothen the transition from (well-tested) classic schemes to (less-tested) quantum-resilient schemes.

To conclude, from a high level perspective, the different parts of this report complement each other and contribute a robust groundwork for building and implementing a future TPM. The latter is achieved in the remaining work packages of this project.

References

[1] Marcel Armour and Bertram Poettering. Substitution attacks against message authentication. *IACR Trans. Symmetric Cryptol.*, 2019(3):152–168, 2019.

[2] Marcel Armour and Bertram Poettering. Subverting decryption in AEAD. In Martin Albrecht, editor, *Cryptography and Coding - 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16-18, 2019, Proceedings*, volume 11929 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2019.

[3] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011.

[4] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 364–375. ACM, 2015.

[5] Rachid El Bansarkhani and Ali El Kaafarani. Direct Anonymous Attestation from Lattices. *IACR Cryptol. ePrint Arch.*, 2017:1022, 2017.

[6] Mihir Bellare, Hannah Davis, and Felix Günther. Separate your domains: NIST PQC kems, oracle cloning and read-only indifferentiability. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2020.

[7] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2014.

[8] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 901–920. IEEE Computer Society, 2017.

[9] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 234–264. Springer, 2016.

[10] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation with subverted TPMs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 427–461. Springer, 2017.

[11] Liqun Chen, Nada El Kassem, Anja Lehmann, and Vadim Lyubashevsky. A Framework for Efficient Lattice-Based DAA. In Liqun Chen, Chris J. Mitchell, Thanassis Giannetsos, and Daniele Sgandurra, editors, *Proceedings of the 1st ACM Workshop on Workshop on Cyber-Security Arms Race, CYSARM@CCS 2019, London, UK, November 15, 2019*, pages 23–34. ACM, 2019.

[12] Rongmao Chen, Xinyi Huang, and Moti Yung. Subvert KEM to break DEM: practical algorithm-substitution attacks on public-key encryption. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 98–128. Springer, 2020.

[13] The FutureTPM Consortium. First report on new QR cryptographic primitives. Deliverable D2.1, FutureTPM, September 2018.

[14] The FutureTPM Consortium. Second report on new QR cryptographic primitives. Deliverable D2.2, FutureTPM, December 2019.

[15] Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 579–598. Springer, 2015.

[16] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-Based Group Signatures and Zero-Knowledge Proofs of Automorphism Stability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 574–591. ACM, 2018.

[17] W. Diffie and M. E. Hellman. Special feature exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, June 1977.

[18] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209. Springer, 2005.

[19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. *CoRR*, abs/1902.07556, 2019.

[20] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In *Annual Cryptology Conference*, pages 335–352. Springer, 2014.

[21] Nada El Kassem. *Lattice-based direct anonymous attestation.* PhD thesis, University of Surrey, 2020.

[22] Shimon Even and Oded Goldreich. On the power of cascade ciphers. *ACM Trans. Comput. Syst.*, 3(2):108–116, 1985.

[23] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: Immunizing HMAC and HKDF. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 105–118. IEEE Computer Society, 2018.

[24] Marc Fischlin and Anja Lehmann. Security-amplifying combiners for collision-resistant hash functions. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 224–243. Springer, 2007.

[25] Marc Fischlin and Anja Lehmann. Multi-property preserving combiners for hash functions. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 375–392. Springer, 2008.

[26] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions revisited. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 655–666. Springer, 2008.

[27] Marc Fischlin, Anja Lehmann, and Daniel Wagner. Hash function combiners in TLS and SSL. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, volume 5985 of *Lecture Notes in Computer Science*, pages 268–283. Springer, 2010.

[28] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptology*, 26(1):80–101, 2013.

[29] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218. Springer, 2018.

[30] 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.

[31] Dusko Karaklajic, Jörn-Marc Schmidt, and Ingrid Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Trans. Very Large Scale Integr. Syst.*, 21(12):2295–2306, 2013.

[32] Nada El Kassem, Liqun Chen, Rachid El Bansarkhani, Ali El Kaafarani, Jan Camenisch, Patrick Hough, Paulo Martins, and Leonel Sousa. More efficient, provably-secure direct anonymous attestation from lattices. *Future Gener. Comput. Syst.*, 99:425–458, 2019.

[33] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, pages 552–586, 2018.

[34] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*, pages 361–372. IEEE Computer Society, 2014.

[35] Anja Lehmann. *On the security of hash function combiners*. PhD thesis, Darmstadt University of Technology, 2010.

[36] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2013.

[37] Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 326–355. Springer, 2019.

[38] Ralph C. Merkle and Martin E. Hellman. On the security of multiple encryption. *Commun. ACM*, 24(7):465–467, July 1981.

[39] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[40] Bertram Poettering and Paul Rösler. Combiners for AEAD. *IACR Trans. Symmetric Cryptol.*, 2020(1):121–143, 2020.

[41] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[42] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[43] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[44] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and

David Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[45] Dominique Unruh. Post-quantum security of Fiat-Shamir. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pages 65–95, 2017.

A List of Abbreviations

Abbreviation	Translation
AE	Authenticated Encryption
AES	Advanced Encryption Standard
ASA	Algorithm Substitution Attack
ANSI	American National Standards Institute
BIKE	Bit Flipping Key Encapsulation
BLISS	Bimodal Lattice Signature Scheme
CA	Certification Authority
CBC	Cipher Block Chaining
CCM	CTR mode with CBC-MAC
CFA	Collision Fault Attack, Collision Fault Analysis
CFB	Ciphertext Feedback
CMAC	Cipher-based MAC
CPU	Central Processing Unit
CRYSTALS	Cryptographic Suite for Algebraic Lattices
CTR	Counter
DAA	Direct Anonymous Attestation
DES	Data Encryption Standard
DFA	Differential Fault Attack, Differential Fault Analysis
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
EAX	Encrypt-then-Authenticate-then-Translate
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve DSA
ETSI	European Telecommunications Standards Institute
EUF	Existential Unforgeability (under Chosen Message Attack)
FA	Fault Attack, Fault Analysis
FO	Fujisaki–Okamoto
GCM	Galois/Counter Mode
HMAC	Hash-based MAC
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IND-CPA/CCA	Indistinguishability under Chosen-Plaintext/Ciphertext Attack
IV	Initialization Vector
KDF	Key Derivation Function

Abbreviation	Translation
KEM	Key Encapsulation Mechanism
KMAC	Keccak MAC
LDAA	Lattice-based Direct Anonymous Attestation
LWE	Learning With Errors
MAC	Message Authentication Code
MD	Merkle–Damgard
MITM	Meet-in-the-Middle
NIST	National Institute of Standards and Technology
NTT	Number-Theoretic Transform
PKE	Public Key Encryption
PQC	Post-Quantum Cryptography
PR	Pseudorandom
PRF	Pseudorandom Function
PRP	Pseudorandom Permutation
OFB	Output Feedback
QR	Quantum-Resistant
QROM	Quantum Random Oracle Model
QS	Quantum Security
ROM	Random Oracle Model
RL	Revocation List
RSA	Rivest–Shamir–Adleman
SCA	Side-channel Attack, Side-channel Analysis
SHA	Secure Hash Algorithm
SIS	Short Integer Solution
SNR	Signal-to-Noise Ratio
SPRP	Strong Pseudorandom Permutation
SUF	Strong Unforgeability (under Chosen Message Attack)
TCG	Trusted Computing Group
TDES	Triple-DES
TPM	Trusted Plaform Module
UC	Universal Composability
XOF	Extendable-Output Function
XOR	Exclusive OR
ZKP	Zero-knowledge Proof
ZKPoK	Zero-knowledge Proof of Knowledge

B NIST Round-3 Candidates

In Table 4 we list the schemes that progressed from Round 2 to Round 3 in the currently ongoing NIST competition that aims at standardizing quantum-resilient cryptography. For further information we refer to the corresponding page on the NIST website: https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

Type	Status	Schemes
public key encryption	finalist	Classic McEliece, **CRYSTALS-KYBER**, NTRU, SABER
	alternate	**BIKE**, FrodoKEM, HQC, NTRU Prime, SIKE
signature scheme	finalist	**CRYSTALS-DILITHIUM**, FALCON, **Rainbow**
	alternate	GeMSS, **Picnic**, **SPHINCS+**

Table 4: Third round candidates of NIST competition. We typeset those schemes in **bold** font that we also suggest in Table 3.

C Explicit References to Technical Scheme Specifications

In Section 2 we present the set of cryptographic primitives that we recommend for a future TPM. The current appendix provides for each of these primitives a reference to its technical specification. We make this reference list self-contained by deliberately separating it from the bibliography of this report.

Category	Ref	Scheme
hash functions	[A]	SHA-256, SHA-384, SHA-512
	[B]	SHA3-256, SHA3-384, SHA3-512
block ciphers	[C]	AES-256
symmetric authentication	[D]	HMAC
symmetric encryption	[E]	CFB
public key encryption	[F]	CRYSTALS-Kyber
	[G]	BIKE
signature schemes	[H]	CRYSTALS-Dilithium
	[J]	SPHINCS+

Table 5: List of cryptographic primitives recommended for mandatory implementation. This list corresponds with the upper part of Table 3.

C.1 Recommendations for Mandatory Implementation

The references [A]–[J] in Table 5 refer to the following documents.

[A] *"FIPS PUB 180-4: Secure Hash Standard (SHS)"*; National Institute of Standards and Technology, 2015; https://doi.org/10.6028/NIST.FIPS.180-4

[B] *"FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions"*; National Institute of Standards and Technology, 2015; https://doi.org/10.6028/NIST.FIPS.202

[C] *"FIPS-197: Advanced Encryption Standard (AES)"*; National Institute of Standards and Technology, 2001; https://doi.org/10.6028/NIST.FIPS.197

[D] *"FIPS PUB 198-1: The Keyed-Hash Message Authentication Code (HMAC)"*; National Institute of Standards and Technology, 2008; https://doi.org/10.6028/NIST.FIPS.198-1

[E] *"SP 800-38A: Recommendation for Block Cipher Modes of Operation"*; National Institute of Standards and Technology, 2001; https://doi.org/10.6028/NIST.SP.800-38A

[F] as submitted to Round 3 of the NIST post-quantum cryptography competition; see https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions; note that the scheme specification and parameter sizes might slightly change in the future.

[G] as submitted to Round 3 of the NIST post-quantum cryptography competition; see https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions; note that the scheme specification and parameter sizes might slightly change in the future.

[H] as submitted to Round 3 of the NIST post-quantum cryptography competition; see https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions; note that the scheme specification and parameter sizes might slightly change in the future.

[J] as submitted to Round 3 of the NIST post-quantum cryptography competition; see https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions; note that the scheme specification and parameter sizes might slightly change in the future.

Category	Ref	Scheme
hash functions	[a]	SHAKE128, SHAKE256
	[b]	BLAKE2b-256, BLAKE2b-384, BLAKE2b-512
	[c]	SM3
	[d]	Ascon-XOF
block ciphers	[e]	Rijndael-256(192), Rijndael-256(256)
	[f]	Camellia-256
symmetric authentication	[g]	KMAC
	[h]	CMAC
symmetric encryption	[j]	CBC
	[k]	CTR
authenticated encryption	[l]	CCM
	[m]	GCM
	[n]	EAX
	[o]	Ascon
public key encryption	[p]	NewHope
	[q]	NTTRU
signature schemes	[r]	BLISS
	[s]	Rainbow
	[t]	Picnic

Table 6: List of cryptographic primitives recommended for optional implementation. This list corresponds with the lower part of Table 3.

C.2 Recommendations for Optional Implementation

The references [a]–[t] in Table 6 refer to the following documents.

[a] *"FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions"*; National Institute of Standards and Technology, 2015; https://doi.org/10.6028/NIST.FIPS.202

[b] *"RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)"*; M-J. Saarinen and J-P. Aumasson, 2015; https://www.rfc-editor.org/rfc/rfc7693.html

[c] *"The SM3 Cryptographic Hash Function"*; S. Shen, X. Lee, R. Tse, W. Wong, and Y. Yang, 2018; https://tools.ietf.org/html/draft-sca-cfrg-sm3-02

[d] *"Submission to NIST: Ascon v1.2"*; Christoph Dobraunig, Florian Mendel, Maria Eichlseder, Martin Schläffer, 2019; https://ascon.iaik.tugraz.at/files/asconv12-nist.pdf

[e] *"AES Proposal: Rijndael"*; Joan Daemen and Vincent Rijmen, 2003; https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf

[f] *"RFC 3713: A Description of the Camellia Encryption Algorithm"*; M. Matsui, J. Nakajima, and S. Moriai, 2004; https://rfc-editor.org/rfc/rfc3713.html.

[g] *"SP 800-185: SHA-3 Derived Functions"*; National Institute of Standards and Technology, 2016; https://doi.org/10.6028/NIST.SP.800-185

[h] *"SP 800-38B: Recommendation for Block Cipher Modes of Operation"*; National Institute of Standards and Technology, 2005; http://dx.doi.org/10.6028/NIST.SP.800-38B

[j] *"SP 800-38A: Recommendation for Block Cipher Modes of Operation"*; National Institute of Standards and Technology, 2001; https://doi.org/10.6028/NIST.SP.800-38A

[k] *"SP 800-38A: Recommendation for Block Cipher Modes of Operation"*; National Institute of Standards and Technology, 2001; https://doi.org/10.6028/NIST.SP.800-38A

[l] *"SP 800-38C: Recommendation for Block Cipher Modes of Operation"*; National Institute of Standards and Technology, 2004; http://dx.doi.org/10.6028/NIST.SP.800-38C

[m] *"SP 800-38D: Recommendation for Block Cipher Modes of Operation"*; National Institute of Standards and Technology, 2007; http://dx.doi.org/10.6028/NIST.SP.800-38D

[n] *"The EAX Mode of Operation"*; M. Bellare, P. Rogaway, and D. Wagner, 2004; http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf

[o] *"Submission to NIST: Ascon v1.2"*; Christoph Dobraunig, Florian Mendel, Maria Eichlseder, Martin Schläffer, 2019; https://ascon.iaik.tugraz.at/files/asconv12-nist.pdf

[p] *"Version 1.1 of the NewHope specification"*; The NewHope team, April 2020; https://newhopecrypto.org/

[q] *"NTTRU: Truly Fast NTRU Using NTT"*; Vadim Lyubashevsky and Gregor Seiler, 2019; https://doi.org/10.13154/tches.v2019.i3.180-201 with a reference implementation available in https://github.com/gregorseiler/NTTRU

[r] *"Lattice Signatures and Bimodal Gaussians"*; Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky, 2013; https://doi.org/10.1007/978-3-642-40041-4_3; we recommend and use the GALACTICS implementation, available at https://github.com/espitau/GALACTICS and documented in https://dx.doi.org/10.1145/3319535.3363223.

[s] as submitted to Round 3 of the NIST post-quantum cryptography competition; see https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions; note that the scheme specification and parameter sizes might slightly change in the future.

[t] as submitted to Round 3 of the NIST post-quantum cryptography competition; see https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions; note that the scheme specification and parameter sizes might slightly change in the future.