D2.1
First 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 / D2.1 / v1.0
Workpackage:	WP 2
Due Date:	September 30, 2018
Actual Submission Date:	September 28, 2018
Responsible Organisation:	IBM
Editor:	Tommaso Gagliardoni
Dissemination Level:	PU
Revision:	v1.0
Abstract:	In this document we begin the analysis of quantum-resistant cryptographic primitives in respect to their use in FutureTPM. The final goal is to identify suitable algorithms for adoption in the FutureTPM specification.
Keywords:	quantum security, quantum resistant, post-quantum, cryptography, primitives, QR

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

Editor

Tommaso Gagliardoni (IBM)

Contributors (ordered according to beneficiary numbers)

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

Executive Summary

Identifying quantum-resistant (QR) cryptographic primitives with suitable properties of security and efficiency is a challenging task. Care must be taken when defining appropriate security modeling against quantum adversaries, and many tradeoffs in terms of security, bandwidth, and speed must be considered. In this part of the FutureTPM project we initiate the analysis of the quantum security of the low-level cryptographic schemes which must be supported by the TPM. We look at different classes of cryptographic schemes (hash functions, block ciphers, digital signatures, public-key encryption, and privacy-preserving primitives) and for each of them we study the appropriate security modeling and definitions. Then, for each of these classes we give a survey of promising candidates to be adopted by FutureTPM, and for each of them we analyze the pros and cons. The final goal of this document is to provide guidelines for the selection, implementation, and integration of the low-level cryptographic functionalities in FutureTPM.

Contents

List of Figures VI

List of Tables VII

1 Introduction 1
- 1.1 The Need for Trusted Hardware . . . 1
- 1.2 From the TPM to FutureTPM . . . 2
- 1.3 Scope of This Document . . . 4
- 1.4 Structure of This Document . . . 4
- 1.5 Interaction with Other FutureTPM Working Packages . . . 5

2 Requirements 6
- 2.1 Notation . . . 6
- 2.2 Computational Hardness Assumptions . . . 7
  - 2.2.1 Lattice-Based Problems . . . 7
  - 2.2.2 Code-Based . . . 8
  - 2.2.3 Multivariate Equations . . . 8
  - 2.2.4 Supersingular Isogenies . . . 8
  - 2.2.5 Hash-Based . . . 8
  - 2.2.6 Comparison . . . 9
- 2.3 Quantum Security . . . 10
  - 2.3.1 Hybrid Models. . . . 12
- 2.4 Other Security Considerations . . . 12
  - 2.4.1 NIST Security Categories . . . 13
  - 2.4.2 Forward Secrecy . . . 13
  - 2.4.3 Universal Composability . . . 13
  - 2.4.4 Leakage Resilience . . . 14
  - 2.4.5 Fault Attacks . . . 14
- 2.5 Efficiency . . . 15
  - 2.5.1 Speed . . . 15
  - 2.5.2 Bandwidth . . . 16
  - 2.5.3 Memory Footprint . . . 16
  - 2.5.4 Scalability . . . 17
- 2.6 Further Requirements . . . 17

3 Hash Functions 18
- 3.1 Security Models . . . 19
  - 3.1.1 One-Wayness (First-Preimage Resistance) . . . 19
  - 3.1.2 Second-Preimage Resistance . . . 19

3.1.3 Collision Resistance 20
3.1.4 Pseudorandomness 20
3.1.5 Indifferentiability 21
3.2 Proposed Candidates 21
3.2.1 SHA-1 Family 21
3.2.2 SHA-2 Family 22
3.2.3 SHA-3 Family 22
3.2.4 Other Candidates 23
3.3 Candidates Comparison 25
3.4 Open Issues 25

4 Block Ciphers **27**
4.1 Security Models 27
4.1.1 Indistinguishability of Ciphertexts 28
4.1.2 Integrity 31
4.1.3 Pseudorandomness 33
4.1.4 Authenticated Encryption 34
4.2 Modes of Operation 34
4.2.1 ECB Mode 35
4.2.2 CBC Mode 35
4.2.3 CFB Mode 35
4.2.4 OFB Mode 35
4.2.5 CTR Mode 36
4.2.6 XTS Mode 36
4.2.7 CCM Mode 36
4.2.8 GCM Mode 36
4.2.9 OCB Mode 36
4.2.10 AEX Mode 36
4.3 Proposed Candidates 37
4.3.1 AES Family 37
4.3.2 Camellia Family 38
4.3.3 Serpent Family 40
4.3.4 Twofish Family 41
4.4 Candidates Comparison 42
4.5 Open Issues 42

5 Digital Signatures **44**
5.1 Security Models 44
5.1.1 Unforgeability 44
5.1.2 ROM VS QROM 46
5.1.3 Transformations 47
5.1.4 Constructions 48
5.2 Proposed Candidates 50
5.2.1 Dilithium Family 50
5.2.2 Tesla Family 51
5.2.3 FALCON Family 52
5.2.4 pqNTRUSign Family 53
5.2.5 SPHINCS Family 54
5.3 Candidates Comparison 55

5.4 Open Issues 55

6 Public-Key Encryption and Key Exchange **57**
6.1 Security Models 57
6.1.1 Indistinguishability 57
6.1.2 Non-Malleability 59
6.1.3 Transformations 60
6.1.4 Construction 61
6.2 Proposed Candidates 62
6.2.1 NewHope Family 62
6.2.2 Frodo Family 64
6.2.3 Kyber Family 65
6.2.4 BIKE 68
6.2.5 Additional Primitives 69
6.3 Candidates Comparison 70
6.4 Open Issues 70

7 Privacy-Supporting Primitives **73**
7.1 Privacy Requirements 73
7.2 (Interactive and Non-Interactive) Zero-Knowledge Proofs 74
7.3 Anonymous Signatures 74
7.4 DAA 75
7.4.1 General Concept 75
7.4.2 Security Models 75
7.4.3 The Existing Schemes 77
7.5 Open Issues 82

8 Conclusions **83**

References **100**

A List of Abbreviations **101**

B List of Algorithms Supported by TPM V2.0 **102**
B.1 Hash Functions 102
B.2 Block Ciphers 103
B.3 Elliptic Curves 105

C DAA Security Model Details **107**

List of Figures

2.1 Key-exchange performance comparison between post-quantum and classical cryptography. Replicated from [163] 15

3.1 The indifferentiability setup 21

4.1 security domain classification for indistinguishability for SKES. 30
4.2 quantum security notion hierarchy for SKES. An arrow means “strictly implies”. . . 31
4.3 security domain classification of integrity for SKES. 33
4.4 quantum security notion hierarchy of integrity for SKES. An arrow means “strictly implies”. 33

5.1 security domain classification for Existential Unforgeability for DSS. 46
5.2 security domain classification for Strong Unforgeability for DSS. 47
5.3 quantum security notion hierarchy for DSS. An arrow means “stricty implies”. . . . 47

List of Tables

2.1 NIST categories for security evaluation of PQC algorithms. 13
2.2 A comparison between the public-key and data structures size for several post-quantum cryptographic systems when compared with classical cryptography. Adapted from [126] 16

3.1 Comparison of the candidate families of hash functions. 26

4.1 Quantum resource estimates for Grover's algorithm to attack AES-k, where $k \in \{128, 192, 256\}$ 38
4.2 Quantum security levels for block ciphers: 128 bit blocksize. 42
4.3 Quantum security levels for block ciphers: 256 bit blocksize. 42

5.1 Parameter sets for LWE-based signature schemes with dimension n, modulo q, (rank k in case of MLWE) over the ring $\mathbb{Z}_q/(\Phi(x))$. The NIST column indicates the NIST security category aimed at. The table also compares the public key and signatures sizes in both schemes. 55
5.2 Parameter sets for NTRU-based signature schemes with dimension n, modulo q, small polynomials f and g, and ring $\mathbb{Z}_q/(\Phi(x))$. The table also compares the public key and signatures sizes in both schemes. 55

6.1 Fujisaki-Okamoto Construction. 60
6.2 Naor-Yung Construction. 61
6.3 PKE from KEM Construction. 61
6.4 NewHope Simple protocol. All polynomials are elements of the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$, where $n = 1024$ and $q = 12289$ [12]. 63
6.1 Parameters proposed for instantiating NewHope in the submission to the NIST process. 64
6.2 Parameters proposed for instantiating FrodoKEM in the submission to the NIST process. 66
6.3 Parameters proposed for instantiating Kyber in the submission to the NIST process. 67
6.4 Comparison of candidate schemes and their public key, secret key, and ciphertext sizes in bytes 71
6.5 Cycle counts for implementations of various candidate schemes on a Cortex-M microcontroller 71
6.6 Cycle counts for implementations of various candidate schemes on CPUs. 72

7.1 Size comparison between these two schemes 82
7.2 Computational cost comparison between these two schemes 82

Chapter 1

Introduction

1.1 The Need for Trusted Hardware

Digital technology has become an embedded part of our everyday life. Despite the benefits that this relationship brings, it also comes with dangers that threaten our modern society. We use computers to manage our financial assets, to monitor health data, to exchange sensitive military data, to provide critical public services such as transportation and law enforcement. With just these few examples, it appears evident that technology should incorporate the best possible security in order to protect critical assets of our lives. Further proof of how technology handles important and highly valuable assets is the rise of cybercrime. Cybercrime is thriving nowadays: by just considering the up to 200 billion dollars laundered every year [118], it has become one of the most grossing illegal activities. This new version of crime includes: identity theft, transaction fraud, piracy, illegal content trafficking, cyberterrorism, and sensitive data theft and exploitation.
The rise of the Internet of Things (IoT), does undoubtedly increase the amount of privacy and security challenges, in addition to the already existing ones. This has led to great efforts in order to preserve the security and privacy of modern resource-constrained devices with protocol standardization, security algorithms, and encryption applied to endpoints as well as network components. Modern cryptography has proven to be one of the most useful tools at our disposal, but it has some caveats. For example, cryptography is based on specific security assumptions, which have to hold in order to guarantee the protection of the system in exam. A case in point: encryption keys should be stored in a safe and restricted place, out of the reach of malicious adversaries. However, due to loopholes and bugs in software, it is almost impossible for a solution relying solely on software to provide a completely trusted environment. Moreover, even the most secure software cannot provide good security guarantees if it's run on *insecure hardware*. Being the lowest-level of the security chain, a compromise at the hardware level could lead to more serious exploitation of the software stack, and make any software mitigation technique useless.
For this reason, an important focus of security research in the last years has been focusing on the realization of trusted hardware, in order to provide a so-called *root of trust*. The latter is essentially a set of hardware solutions that provide a trustworthy environment on which other parts of the system can rely upon. Solutions based on trusted hardware provide a safe and tamper-proof environment for sensitive operations. Based on trusted hardware, several security-critical functions can be realized, such as verification of the running software and the protection of cryptographic keys. Realizing trusted hardware is challenging, as it must be protected from several attack vectors, such as fault injection attacks and unauthorized firmware modification.

1.2 From the TPM to FutureTPM

The Trusted Computing Group (TCG) is an industrial alliance of many large hardware and software vendors. In order to tackle the need for hardware solutions, TCG formulated the *Trusted Platform Module (TPM)*, a hardware module with the sole purpose of providing a root of trust. The TPM is a secure piece of hardware that provides an isolated and secure environment for sensitive operations, and which can serve as a trust anchor on top of which secure systems can be built. It is designed to enhance platform security beyond the capabilities of software, and shield cryptographic and sensitive material from software-based attacks. Moreover, augmenting computers with these hardware modules adds powerful functionality in distributed settings, allowing us to reason about the security of these systems in new ways. Although in its definition the TPM is a hardware solution, it can also be virtualized and codified if proper care is taken.
The purpose of a TPM is to provide security and privacy for the end system. In order to achieve that, the TPM must provide many different security functionalities: encryption, authentication, secure storage, attestation, and privacy preserving techniques amongst others. The basic benefit of a TPM is that other entities can be sure that the data provided from the TPM is untampered and legitimate, so that a chain of trust can be built from the hardware level up to the software level.
The TPM has undergone some changes during its lifetime. The latest version is 2.0 which was released in 2014 and the latest update for the 2.0 specification was made in 2018. Nowadays, TPMs can be found everywhere, from laptops to smartphones. Many vendors have developed hybrid software/hardware solutions based on the TPM in order to maximize its adoption. Some examples are: OPAL drive encryption, secure boot, and firmware lock. So far, the TPM is one of the most successful hardware security solutions in terms of production numbers and security assurance. However, so far the TPM uses standard cryptographic techniques as underlying building blocks. The rise of quantum computing could threaten the security achieved so far.

The Threat of Quantum Computing

Unfortunately, the cryptography used in many modern cyber infrastructures, and especially in the existing versions of TPM, is threatened by the advancement of quantum computing (QC) technology. A quantum computer is a computation system which exploits the laws of quantum mechanics in order to perform operations on data. In a theoretical sense, a quantum computer can always perform the same kind of computation and execute the same programs that a classical computer does, with the same speed and efficiency. However, in addition to that, a quantum computer can also perform operations on *quantum data* (represented as basic units of quantum information called *qubits*, or “quantum bits”), and execute complex tasks on classical and quantum data alike using sequences of quantum-mechanical transformations in the form of *quantum algorithms*. This gives two powerful advantages to quantum computers in respect to classical ones:

1. on one hand, quantum information is inherently more complex than classical information. A system composed of n classical bits can easily be represented by a quantum system of n qubits, but the converse does not hold, because quantum information exhibits properties, such as *superposition* and *entanglement* [127], which cannot be represented efficiently on a classical computer. The consequence is that, generally speaking, a quantum system composed of n qubits can store a larger amount of information than a classical system composed of n bits.

2. Moreover, quantum algorithms can exploit the aforementioned properties in order to execute certain tasks more efficiently, in a way that it is simply not possible with a classical

computer, regardless of its potency. The resulting speedup in solving certain problems makes a quantum computer capable of quickly solving tasks that the most powerful classical computer on Earth would take billions of years to solve.

Not all the tasks which are hard for a classical computer become easy for a quantum computer. Actually, for most of the mathematical problems studied in the scientific literature, the speedup allegedly offered by quantum computers is not too significant, and there is currently no proof that computational tasks exist, which are inherently unsolvable by a classical computer but solvable by a quantum one. Technically speaking, this is expressed by the fact that it is currently unknown whether the computational class BPP (the class of problems efficiently solvable with good probability on a classical computer) is a strict subset of BQP (the analogous class for quantum computers) or not. However, there exist a few problems which are potential candidates to show such separation: namely, problems which are believed to be hard for any classical computer (for which, to date, no efficient classical algorithm is known despite decades or even centuries of research) but for which an efficient quantum algorithm is known already.
Unfortunately for the current state of cryptography, many of these problems just happen to be right those ones which are extensively used to secure the most ubiquitous and successful cryptographic schemes ever invented. Cryptosystems such as RSA, DSA, and ECDSA, are currently deployed in countless applications, and it would be unthinkable to imagine modern technology keep working without them. Their security is based on the premise that the mathematical problems they are built upon (integer factorization, discrete logarithm, etc.) are too hard to be solved even by the most powerful classical computer today. However, it has been known since the ‘90s that a quantum computer could easily solve all of these problems, making the aforementioned schemes insecure, and depicting a grim “cyber-apocalypse” given the current social dependancy on trusted digital infrastructures.
The only reason why such a “cyber-apocalypse” has not manifested so far is that building quantum computers is very hard, and it is a feat which has not been substantially accomplished so far. Even if, theoretically, we know the underlying physical principles behind their working, from a practical standpoint there are huge engineering challenges to be solved before a quantum computer powerful enough to threaten modern cryptography can be built. Working prototypes of quantum computers exist, but they are limited to relatively few operations on a small number of qubits.
Despite this, the last few years have seen an incredible acceleration in the pace to the realization of a large-scale quantum computer. This trend seems to suggest that the currently deployed cryptographic schemes might not stay secure for long time still. This also means that most of today’s secure infrastructures (including the TPM) could be irremediably compromised in the future.

Quantum-Resistant Cryptography

In order to prevent the total breakdown of modern cryptography when quantum computers arrive, new solutions have been investigated in the last decades. The so-called *quantum-resistant (QR) cryptography* (sometimes called *post-quantum cryptography*) is a relatively recent branch of cryptography, which aims at building cryptosystems based on mathematical problems believed to be hard *even for a future quantum computer*. Such new cryptosystems can still be run on today’s classical computers, but they can secure our data and communication against future breaches.
QR cryptography is therefore an important direction for the future of cybersecurity, but comes with a few challenges.

First of all, it is imperative to accelerate the adoption of QR cryptography now, before it is too late. The reason is that large-scale adoption of new cryptographic standards has historically proven to be a long and tedious task. First, academic trust has to be built around the new proposed schemes, which have to undergo years of cryptanalytic scrutiny. Then, standards have to be issued, and trust in these standards has to be built in the industry and application sector. Then the new schemes have to deployed on final products, which on their side might have a long “shelf life”, and have therefore to maintain their security for all that time at least. Timescales for this kind of adoption easily reach decades. In the meantime, every infrastructure which does not adopt the new QR schemes remains vulnerable to quantum attacks. This is especially troublesome considering that in many cases sensitive data can be harvested right now in the present by malicious actors, to be decrypted at a later time by a future quantum computer. So, the sooner we can transition to a QR infrastructure, the better.
Another challenge is given by the fact that many QR solutions offer improved security at the cost of worse performance. This has to be taken into account when deploying QR cryptography on resource-constrained platform such as the TPM. Scientific research aimed at improving the efficiency of QR cryptography is ongoing.
Finally, when designing QR solutions, care must be taken in ensuring the right security properties and in correctly modeling the threat scenarios and trust assumptions. Large-scale quantum computers are still not around, but this also means that it is currently unclear what kind of attacks are going to be possible in the future. Correct cryptographic practice mandates studying the new proposed schemes in scenarios able to capture all the security properties required.
All these considerations tell us that making existing infrastructures such as the TPM resistant against QC is not straightforward.

1.3 Scope of This Document

The purpose of this document is to pinpoint, analyze and evaluate currently available QR cryptographic primitives as well as perform a security and performance-wise comparison of them, in order to gain a better understanding of QR algorithms suitable for FutureTPM, and their quantum security in general. The outcome of this deliverable will drive and facilitate the selection of most suitable QR algorithms for the purposes of FutureTPM. More specifically, in this document, there will be an extensive research on hash functions, block ciphers, digital signatures, asymmetric encryption and key exchange solutions, as well as privacy-preserving schemes that meet the technical requirements and goals of the FutureTPM project (as specified in deliverable D1.1). Furthermore, for all the presented QR cryptographic primitives, we analyze security models along with description of their parameters. We also provide a list of proposed candidates for each QR cryptographic primitive, as well as performing an evaluation of them from a security and performance point of view. Finally, open issues are also identified and highlighted.

1.4 Structure of This Document

After this introduction, chapter 2 will present technical requirements that should be met for QR algorithms to be feasible and deployable in TPM. In the next chapters, there will be an extensive analysis of each cryptographic primitive that has a QR counterpart. More specifically, in chapter 3 we are going to analyze hash functions, while in chapter 4 we present block ciphers. Chapter 5 elaborates on digital signatures and chapter 6 analyzes asymmetric encryption and key exchange. Moreover, in chapter 7, there is an analysis of privacy preserving schemes including zero knowledge proofs, anonymous signatures and direct anonymous attestation (DAA) protocols. Finally, chapter 8 concludes this deliverable, while in the appendix a list with cryptographic algorithms that TPM 2.0 supports is presented.

1.5 Interaction with Other FutureTPM Working Packages

This document interacts with five other packages namely WP1, WP3, WP4, WP5, and WP6. In particular, this deliverable is directly linked with D1.1 FutureTPM Use Cases and System Requirements of WP1, since the presented QR algorithms should fulfill all the requirements of FutureTPM and its use cases, as they were specifically defined and analyzed in D1.1. Moreover, this document sets the technical basis of the cryptography subsystem as it will be described in D1.2 FutureTPM Reference Architecture from WP1, in which the consortium will outline the specifications of the overall FutureTPM platform.
The interplay between this deliverable and D3.1 will be of paramount importance, since the definition of the security and threat models that will be defined in D3.1 will be based on the QR cryptographic primitives presented in this deliverable. Furthermore, the outcome of this document will be used in the risk assessment of the FutureTPM platform in WP4, in the context of the trust, threat, and adversary models identified in WP3. Additionally, the algorithms provided by this document will be implemented, tested, and evaluated for their performance in WP5. Finally, the implementations of these algorithms will be incorporated in demonstrators of the FutureTPM framework in WP6.

Chapter 2

Requirements

To guarantee the security of a cryptographic scheme two items are necessary: a theoretical model of the adversary and a security reduction, i.e., a mathematical proof that if the adversary can break the scheme, then she can also solve some hard mathematical problem. If the underlying problems are computationally infeasible for the particular type of adversary considered in the model, the cryptographic protocols are secure.
Once quantum computers will be introduced, those mathematical hardness assumptions should continue to hold even in presence of an adversary with quantum computing capabilities. Nevertheless, the last thirty years of research have shown that the impact of the quantum computer on security is different for symmetric and public key cryptography. Symmetric key algorithms, i.e., algorithms that use the same (secret) key for encryption and decryption, are impacted for now only by Grover's algorithm [83], that is an extensive search algorithm with a quadratic speed up when compared to classical brute force attacks. The case is different for public key cryptography, i.e., cryptographic schemes that have a pair of keys, a public key that can be shared and a secret key. Already in 1994 Shor proved that the schemes that are in use nowadays are not secure against a quantum computer [146]. His results stimulated cryptographers to look for alternatives, originating a new branch of cryptography that is now known as *post-quantum cryptography*. In the following we will present the possible hardness assumptions that can be used to guarantee security in a quantum world and the different adversarial models that can be used to build security proofs.

2.1 Notation

We use *w.l.o.g.* for "without loss of generality", *iff* for "if and only if", and *classical* meaning "non-quantum". We adopt commonly used notation in the academic cryptographic literature. In particular, we denote by λ an (integer) security parameter, or we write 1^λ if expressed in unary notation. We say that a function is *negligible* in λ (and we write this as $\mathsf{negl}(\lambda)$) iff it shrinks to zero faster than $\frac{1}{p(\lambda)}$ for any polynomial p. PPT stands for *probabilistic polynomial-time Turing machine*, while DPT stands for *deterministic polynomial-time*.
We refer to [127] for a treatment of quantum information processing, quantum information theory, and quantum computing. We recall some basic notation: $|\varphi\rangle$ represents a quantum state labeled by φ, and $\langle\varphi|$ its dual. If $f : X \to Y$ is a function (w.l.o.g., on bitstring spaces X and Y), we denote its *quantum-accessible counterpart* as the unitary:

$$U_f \, |x, y\rangle := |x, y \oplus f(x)\rangle$$

Such unitary can always be efficiently constructed by any quantum machine able to implement f classically. This unitary is called *canonical classical-quantum oracle*, or *type-(1) oracle* [77], and for the sake of clarity we will denote it simply as $|f\rangle$. More generally, if $\mathcal{A}$ is a QPT algorithm and f is a classical function, we denote by $\mathcal{A}^f$ that $\mathcal{A}$ has classical oracle access to f, while we denote by $\mathcal{A}^{|f\rangle}$ quantum oracle access to the type-(1) oracle for f.

2.2 Computational Hardness Assumptions

The post-quantum security of a cryptographic scheme should be based on mathematical problems that are unfeasible to solve even using a quantum computer. Hence, it is not possible to use assumptions like Factoring [133] or Diffie-Hellman[64], as they are known to be solvable in polynomial time with a quantum computer (cf. [146]). In the last 20 years 5 different classes of problems emerged that seem to be quantum-resistant, and they are classified depending on the algebraic structure they are based on. In the following we present these hardness assumptions, and we provide a comparison among them.

2.2.1 Lattice-Based Problems

There are many hardness assumptions on lattices, depending on the type of the underlying lattice and on the type of problem needed. The main assumption is the *Shortest Vector Problem* (SVP) that asks to find the vector with the smallest norm in the lattice. Anyway, most cryptographic protocols do not base their security on SVP, but on other assumptions like SIS and LWE. For example, lattice-based encryption is based on LWE [135], while the security of digital signatures is often based on SIS [2]. There are different versions of LWE (e.g., Ring-LWE, Module-LWE, ecc.) and SIS (e.g., Ring-SIS, Module-SIS, ecc.) depending on whether the underlying lattice is built over the space of the integers, or of the polynomials with integer coefficients. The drawback of these variants is that the lattices on which they are based have an additional algebraic structure and it is unclear whether such structure can be exploited for ad-hot attacks. Despite this, it is preferred to use their Ring versions when designing protocols, as they allow for more practical protocols in terms of storage requirements.
The extensive literature on lattices is one of their main advantages: the lack of quantum attacks to lattice-based protocols is taken now as a reliable indicator for their quantum-resistance. Furthermore, lattices are very versatile, and allow to build not only basic primitives but also more complex schemes, like fully homomorphic encryption [78], that have no counterpart based on classical hardness assumptions. From a protocol design point of view, they can count on worst-case to average-case reductions that guarantee that either a problem is hard on average, or the problem is not hard at all. This property allows to circumvent the instance selection problem that cryptography based on factoring or discrete logarithm has (factoring a number chosen at random is not always hard, e.g., if the selected number is even, or has small factors.). Finally, operations on them are highly parallelizable, and encryption schemes are nowadays as fast as RSA. Unfortunately, lattices are not suitable to build lightweight cryptography, as they still suffer from large key sizes despite multiple optimizations. Moreover, cryptographic primitives based on them are hard to combine efficiently, thus they do not allow to build more complex protocols that are also usable practical.

2.2.2 Code-Based

Algebraic codes were introduced to encode and transmit messages on noisy channels. Indeed, if the amount of noise is bounded, it is possible to recover the entire message using a trapdoor. Such property can be also transformed into a hardness assumptions, that is known as *Decoding Problem*, and that can have many different formulation depending on the type of code in use (e.g., decoding Reed-Solomon codes [134]) or on additional requirements on the message (e.g., *Bounded Distance Decoding*). Cryptography built on codes was one of the first to be introduced [113], and withstood a lot of attempted attacks. Both basic primitives (like digital signatures and public key encryption [113]) and more complex ones (like zero-knowledge proofs [115]) can be built on them.
From a security perspective, even if there are no targeted quantum attacks against codes, they still suffer from side channel attacks due to their non-negligible decoding error. Moreover, usually they have large public keys. To shorten them, some schemes use particular codes that have more algebraic structure and allow for a more compact representation (hence, smaller storage requirement for keys). Unfortunately, as in the case of lattices this added structure could allow for targeted attacks, as protocols based on them benefit from the extensive cryptanalysis already present in literature.

2.2.3 Multivariate Equations

The idea behind cryptosystems based on multivariate equations is that solving some systems of equations in many variables is hard under some constraints. This type of hardness assumptions were introduced in the early days of cryptography [111], and allow to build digital signatures and encryption schemes. Over the years, those protocols improved to be computationally efficient and to allow for short signatures and keys, thus making them suitable for use in small computing devices with limited resources. While the security of the basic schemes is well understood though, the most efficient versions are based on different problems that are not as well studied as the original ones.

2.2.4 Supersingular Isogenies

This recently introduced type of cryptography bases its security on the problem of finding some particular transformation (in particular, an isogeny) between two given elliptic curves. This problem is new, thus it does not have an extensive cryptanalysis, but it is seen favorably as it would allow to reuse some of the infrastructure already in place (like libraries or hardware optimized to run EC-based cryptography). As of now, they allow building efficient encryption schemes (one is in submission to NIST, see [98]), and signatures [168], even if the latter are still not suitable for practical use.

2.2.5 Hash-Based

Cryptographic hash functions are particular functions that map arbitrarily-long data into fixed-length strings. They allow to build digital signatures whose security does not require specific hardness assumptions, but relies on some general assumptions on the function itself. For example, the hash function should be hard to invert, or it should be hard to find a collision, i.e. two different inputs on which the function gives the same output. As long as it satisfies the general assumptions, the particular implementation of such functions can then be on any algebraic structure, for example on lattices, or on elliptic curves, or they can be constructions that are not based on mathematical problems but on an ad hoc basis, where the bits of the message are mixed to produce the hash (e.g. the SHA family). This means that if the security of the hash function is compromised, to restore the security it is enough to change the type of hash, not the entire signature structure (e.g., this is what is happening with the hash function SHA-1 after a collision was found [151]). This is important, as it means that these schemes feature hight security as they can be based on the weakest cryptographic assumptions. Unfortunately, Rudich and Impagliazzo proved that public key encryption (PKE) schemes cannot be built from hash functions [94].

2.2.6 Comparison

To summarize the previous remarks, we compare here the advantages and disadvantages of all the previous types of hardness assumptions.

From a *security* point of view, it is usually advisable to rely on hardness assumptions that have been well-studied, such as those in the lattice-based or code-based settings. Nevertheless, both of them suffer from quite large public keys, and to circumvent this problem the security of practical schemes is usually based on modifications of the original assumptions that have more algebraic structure and that are less studied. This last remark holds also for multivariate equations hardness assumptions. Hardness assumptions on supersingular isogenies are quite new (they were first stated in 2012), and there is not much cryptanalysis of their security.

Regarding *efficiency* of implementations, lattices can count on an inherently parallelizable structure and quite efficient implementations. As it was highlighted by D. Moody during the First PQC Standardization Conference, schemes based on lattice hardness assumptions tend to have more efficient implementations than schemes based on other assumptions.

Practicality requires not only an efficient implementation, but also small keysize and compatibility with legacy hardware and software infrastructure. This is a great advantage of codes and elliptic-curve isogenies, as they can exploit the already existing optimized hardware and coding libraries. Regarding lattices, research on optimized hardware is at its early stages (cf. [7]). On the storage requirement side, we have that codes, lattices, and isogenous elliptic curves suffer from long keys, so to build lightweight cryptography multivariate equations appear to be a better choice.

Finally, we analyze the primitives from a *versatility* point of view. It is possible to build both encryption schemes and digital signatures based on all the hardness assumptions we presented apart from hash-based. In fact, it is known that PKE schemes cannot be built from hash functions only [94]. We highlight though that on codes and lattices it is possible to build more complex primitives too. In particular, lattices allow to construct homomorphic encryption, that we do not know how to obtain from any of the other assumptions.

Hash-based digital signatures have extremely efficient implementations, and the security is based on very minimal assumptions, i.e. only the hash function used needs to have some security properties. The interesting point is that, while signature schemes based on other assumptions require secure hash functions *in addition* to their hardness assumptions, for hash-based signatures it is enough to have secure hash functions. The drawback is that these schemes usually allow to sign only a finite (although usually very large) number of messages, because they are inherently based on one-time signature schemes. Moreover, some of them (like XMSS) are stateful, i.e. require the signer to store an internal state that has to be updated every time a signature is produced.

2.3 Quantum Security

In this section we describe a classification of “quantum security models”, or scenarios. This is a useful labeling in order to clarify the type of threat we are taking into account. We will do so by identifying three main “quantum security classes”, or “domains”, each of them encompassing the security notions and constructions related to a particular scenario. We denote these classes by QS (standing for ‘quantum security’), followed by a number identifying the class.
Notice that such classification only takes into account notions of *computational security*, meaning, security against adversaries modeled as bounded computing machines, which is the most adopted paradigm in nowaday’s cryptography. *Information-theoretic security*, on the other hand, provides security guarantees holding *regardless* of the nature of the adversary model. This is a clearly stronger paradigm, and in particular makes no distinction between classical and quantum security: schemes which are information-theoretically secure remain unattackable by classical and quantum computers alike. In fact, a common technique to show that a certain scheme is quantum-secure is to show that it is actually information-theoretically secure.
Despite its appeal, information-theoretic security comes with many limitations though, both in terms of efficiency and in terms of the cryptographic tasks that can be realized. In the case of FutureTPM, quantum security will usually be achieved through other means. If and when information-theoretically secure solutions can be considered for being adopted in FutureTPM, we will explicitly mention so.

QS0: Classical Security.

We start with the class QS0, which is “classical computational security”, meaning that no quantum scenario is considered at all. This is going to be used as a mean of comparison with the state of the art of non–quantum-resistant security solutions. Usually, in QS0, an adversary is modeled as a probabilistic, polynomially bounded Turing machine (or PPT in short). This means that a “reasonable” adversary in QS0 runs in time polynomial on the security parameter of the scheme she tries to attack, by using a classical computing device, and performing classical input/output operations.

QS1: Post-Quantum Security.

In the class QS1 we describe the canonical paradigm of “post-quantum cryptography”. In this paradigm, the cryptographic schemes are still classical and meant to run natively and efficiently on a classical computer (because honest parties are classical). However, adversaries are now considered to be equipped with a large-scale quantum computer. This allows an adversary to run quantum algorithms (like Shor’s or Grover’s algorithm) locally.
In QS0, during interactive attacks, the interaction between the adversary and the honest parties is still classical - because the honest parties are classical, and are therefore unable of quantum communication. This interaction, in security proofs, is usually modeled by *oracles*, which map inputs x to outputs $\mathcal{O}(x)$. This, however, does not mean that in QS1 every oracle should be classical. A typical example is the *quantum random oracle model (QROM)* [42], where a quantum oracle of the form:

$$|\mathcal{O}\rangle : \sum_{x,y} \alpha_{x,y} |x, y\rangle \mapsto \sum_{x,y} \alpha_{x,y} |x, y \oplus \mathcal{O}(x)\rangle$$

is used to model the fact that a quantum adversary which knows the (public, classical) program code of a hash function can implement that code on a quantum computer, being therefore able to

evaluate the hash function on a superposition of inputs.
In fact, it is important to remember that a quantum computer is (from a theoretical perspective) a generalization of a classical one: every code which can run on a classical computer can also run on a quantum computer. This fact must be taken into account when considering QS1 (post-quantum) security: anything which does not require the adversary to interact with other honest parties and can be computed locally, can be computed by the adversary in a quantum way.

QS2: Superposition-Based Quantum Security.

The QS2 domain represents a strengthening, or generalization, of QS1. It is, to some extent, *quantum security beyond post-quantum*. In this domain, the schemes are classical and the adversaries are quantum, as in QS1. However, unlike in QS1, the adversaries are given quantum access to classical oracles not only when the "realistic" model requires it. So, for example, encryption schemes in QS2 must provide security against adversaries with quantum access to the encryption oracle, even in the secret-key case, and digital signature schemes must be unforgeable towards adversaries with quantum access to the signing oracle, even if such schemes are still classical. The resulting security notions can be strictly stronger than post-quantum notions as defined in the QS1 sense. Constructions which are secure in QS2 retain in particular their security in QS1, but the converse does not always hold. When a cryptographic construction is secure in the QS2 sense, we will often just call it quantum-secure.
It has to be stressed that a QS2-secure cryptographic scheme is, in particular, QS1-secure, meaning that QS2-secure schemes automatically inherit post-quantum security. Unfortunately, QS2 security is often more "expensive", in terms of efficiency or requirements, than "normal" post-quantum cryptography as defined in QS1. However, in the context of FutureTPM, QS2-secure cryptography offers stronger security guarantees, and should be considered as an appealing option when the performance overhead is not too much, for two main reasons.
The first reason is that QS2-secure primitives have often desirable properties which, although not necessary in the "standard" post-quantum view of security, allow for better composition and improved results in post-quantum security proofs. A typical example is the emulation of a quantum random oracle: since the QROM describes an object with quantum superposition access by definition, emulating it in a security proof by using post-quantum PRFs would not be enough, because post-quantum PRFs are only accessed classically (because they depend on a secret key, cf. Section 3.1.4), and their security model says nothing about what happens when the access is quantum. For this reason, if we want to emulate a quantum oracle with PRFs, we need a security model which covers the quantum superposition access, even if we are using the quantum random oracle "only" in a post-quantum security proof. Another example is the case of post-quantum obfuscation, in particular indistinguishability obfuscation (iO). This is a relatively recent branch of cryptographic techniques which, roughly speaking, achieves certain functionalities by "obfuscating" the code of some algorithm in a secure way. One typical example (which has also received interest [CEJvO02] from an application perspective) is how to build public-key encryption schemes from symmetric-key encryption schemes. The idea is to hardcode the secret key of the symmetric-key encryption scheme in the code of the encryption routine, and then obfuscate the resulting code and distribute it as a public key. In the standard model, it is known [IR88] that it is impossible to achieve key-exchange and public-key encryption in a black-box way just from symmetric-key encryption. However, by using iO and the approach just described (often dubbed '*"whiteboxing"*), it might be possible. Regardless whether iO is a reasonable assumption or not, it is clear that for this to work, the post-quantum security of the underlying symmetric-key scheme would not be enough because post-quantum public-key encryption can be queried in

superposition. Therefore, for this application we would also need a superposition-based (QS2) security notion for symmetric-key encryption.
The second, less obvious reason for considering QS2 security for FutureTPM regards the physical interaction between the adversary and the device where the cryptographic code is running. An adversary able to “trick” a classical computation device into quantum behavior might exploit such behavior to gain superposition access to the function computed by the device. Consider an adversary equipped with some future technology which subjects a FutureTPM device to a fault-injection environment, by varying the physical parameters (temperature, power, speed, etc.) under which the device usually operates. As a figurative example, our “quantum hacker” could place the chip into an isolation pod, which keeps the device at a very low temperature and shields it from any external electromagnetic or thermal interference. This situation would be analogous to what happens when security researchers perform side channel analysis on cryptographic hardware in nowaday’s labs, using techniques such as thermal or electromagnetic manipulation which were previously considered futuristic. There is no guarantee that, under these conditions, the chip does not start to show full or partial quantum behaviour. At this point, the adversary could query the device on a superposition of plaintexts by using, e.g., a laser and an array of beam splitters when feeding signals into the chip via optic fiber. It is unclear today what a future attacker might be able to achieve using such an attack. As traditionally done in cryptography, we assume the worst-case scenario where the attacker can actually query the target device in superposition. Classical and post-quantum security notions do not cover this scenario. This setting is an example of what we mean by “tricking classical parties into quantum behaviour”.
It is important to stress that no such “quantum fault injection attacks” are known to exist today, but there is currently no guarantee that they will not become feasible in the future, also considered the rapid pace of miniaturization of the current electronic components, up to the nanometer scale where quantum effects are not negligible anymore. Moreover the threat deriving from these kind of attacks is potentially high considering that, unlike for the post-quantum scenario, they do not necessarily require the adversary to build a fully-fledged quantum computer. It is possible that such sort of attacks might threaten the security of a non–QS2-resistant TPM platform way before large-scale quantum computers become reality.

2.3.1 Hybrid Models.

We will also consider certain quantum security models, which fall somewhere in between any two of the above categories. For example, QS0.5 represents quantum domains intermediate (in terms of security) between QS0 and QS1. Two typical cases of QS0.5 notions are:

- schemes (e.g., signatures) which are based on quantum-hard computational assumptions, but whose security proof is only given in the (classical) ROM; and
- schemes with hybrid classical / post-quantum security guarantees (for example, group signatures with post-quantum anonimity but classical unforgeability).

2.4 Other Security Considerations

Herein, security properties are considered that complement the ones provided by the security models put forth in the previous section.

NIST Category	Description
1	Attack with similar complexity of those required for breaking AES-128
2	Attack with similar complexity of those required for collision search SHA256/ SHA3-256
3	Attack with similar complexity of those required for breaking AES-192
4	Attack with similar complexity of those required for collision search SHA384/ SHA3-384
5	Attack with similar complexity of those required for breaking AES-256

Table 2.1: NIST categories for security evaluation of PQC algorithms.

2.4.1 NIST Security Categories

To simplify comparisons, NIST has introduced their own methodology of broad security strength categories that are easier to compare that bit-security levels. The NIST security categories are listed in Table 2.1. Every submitter of an algorithm has to assess the security of a parameter set and has to claim a NIST category. The recommendation by NIST is to focus on categories 1, 2, and/or 3. However, a parameter set in category 4 or 5 does provide some additional margin in case of future improvements on the efficiency of attack algorithms. This is more likely for quantum-resistant cryptography, as the underlying computational problems are less studied.

2.4.2 Forward Secrecy

Forward secrecy is a characteristic of certain cryptographic protocols, wherein the compromise of a long-term key does not compromise past session keys. The TPM 2.0 standard specifies decrypt and encrypt sessions, which allow to protect command/response parameters in insecure mediums [17]. Two different symmetric-key modes can be used to encrypt and decrypt the parameters: XOR and CFB. For the XOR mode, a mask is generated based on nonces, an HMAC key and other data, and XORed with the data to be encrypted or decrypted. For the CFB mode, a KDF is used to generate the encryption key and the initialisation vector. The inputs to the KDF are, among others, the session-key and nonces. Since the generation of both the mask and the encryption key rely on values derived from an authorisation value, compromise of the authorisation value may compromise past sessions. In contrast, the TPM may support the forward secrecy of other protocols by providing the means to securely generate random key material.

2.4.3 Universal Composability

Universal composability guarantees that protocols maintain their security in any context, even in the presence of an unbounded number of arbitrary concurrent protocol instances controlled by an adversary [54]. This notion matches well today's computational and network settings. This framework is based on the indistinguishability between an ideal process to carry out the task at hand and the protocol whose security is being evaluated. In the ideal process, all parties would transfer their inputs to a trusted party, that would operate on the data and transfer the results back to the expected recipients. A protocol π securely evaluates a function f if, for any adversary $\mathcal{A}$ there exists an ideal adversary $\mathcal{S}$, such that no environment $\mathcal{E}$ can tell with non-negligible probability whether it is interacting with π and $\mathcal{A}$ or with $\mathcal{S}$ and the ideal process for f. The environment $\mathcal{E}$ represents processes external to the protocol execution, such as other protocol executions, human users, etc. Proving the security of protocols within this framework may allow for the TPM to serve several requests simultaneously. While specifying security with respect to an ideal process can be intricate, the UC model can express any combination of properties and is amenable to modular analysis.

2.4.4 Leakage Resilience

Side-Channel Attacks (SCAs) attempt to break a cryptosystem through the information that is gained from the implementation of a computer system [148]. 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 randomise 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 [149]. Generally, the basic blocks used in these constructions use subkeys, and techniques are applied so that an adversary can only observe the encryption of q different plaintexts under a certain subkey. System designers will then limit the success rate of the best available adversary for q queries, by limiting the information leakage.
Protection against physical attacks might be taken into account when developing the FutureTPM, since otherwise even non-invasive attacks, such as those based on Electromagnetic Radiation and Timings may lead to a complete breakage of the system.

2.4.5 Fault Attacks

Fault Attacks (FAs) are active attacks 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 [100]. This type of attack may be achieved through non-invasive means: attacks such as RowHammer have shown that it is possible to change certain memory bits by changing the bits of neighbouring rows 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 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.
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

Scheme	Type	Time $[ms]$			Communication $[bytes]$		Security bits	
		Alice0	Bob	Alice1	$A \rightarrow B$	$B \rightarrow A$	classical	quantum
RSA 3072 bit	RSA	-	0.08	3.76	384	384	128	-
ECDH nistp256	ECDH	0.225	0.586	0.268	32	32	128	-
BCNS15	R-LWE	0.881	1.41	0.179	4 096	4 224	86	78
NewHope	R-LWE	0.055	0.084	0.015	1 824	2 048	229	206
NewHope MSR LN16	R-LWE	0.046	0.079	0.014	1 824	2 048	128	128
NTRU EES743EP1	LWE	1.08	0.116	0.068	1 027	1 022	256	128
Frodo	LWE	1.95	2.30	0.091	11 280	11 282	144	130
SIDH	SIDH	113	251	106	576	576	192	128
SIDH (compressed)	SIDH	387	586	158	336	336	192	128
McBits	Error-correcting codes	129	0.038	0.106	311 736	141	157	157

Figure 2.1: Key-exchange performance comparison between post-quantum and classical cryptography. Replicated from [163]

improve the SNR of an SCA attacker. Furthermore, FAs may be combined with SCAs to produce more powerful attacks.
While it is impossible to fully protect against FAs, and certain attack-vectors, such as the RowHammer attack might not be applicable under the considered FutureTPM design, it might be useful to take into consideration possible FA countermeasures when designing the FutureTPM system, so as to prevent private-key material from being disclosed to an attacker, who could afterwards impersonate the attacked chip.

2.5 Efficiency

Low costs and the usage of long-established cryptographic primitives have been main factors for the wide dissemination of the TPM standard. Typical designs of TPM chips include a secure controller of 8, 16 or 32 bits (such as ARM SecurCore), hardware accelerators for operations such as long-integer modular arithmetic and hash functions, and RAM, ROM and EEPROM memories of tens of kilobytes [152, 95]. TPMs typically communicate with the main processor via a Low Pin Count (LPC), a Serial Peripheral Interface (SPI) or a Inter-Integrated Circuit (I^2C) bus.

2.5.1 Speed

The performance of several key-exchange mechanisms was replicated from [163] in Fig. 2.1. In the considered scenario, Alice wants to establish a session-key with Bob. Alice will produce a first message in the `Alice0` step and transmit it to Bob ($A \rightarrow B$). Afterwards, Bob will derive the shared key and compute the reply in the `Bob` step, and send the message to Alice ($B \rightarrow A$). Finally, Alice will compute the session key in the `Alice1` step. All the results were measured with a Intel i7-4790K running at 4 GHz. These results might give pointers on how the performance of the TPM will be affected when post-quantum cryptography is considered, namely by comparing the performance figures of post-quantum cryptography with those of classic cryptosystems. While lattice-based cryptography (LWE and R-LWE) achieves execution times that are in the same order of magnitude as classical cryptosystems, it needs to communicate messages that are tens or hundreds of times larger. In contrast, isogeny-based cryptography (SIDH) executes hundreds of times slower than classical cryptography, but the exchanged messages are of about the same size

Scheme	Public-key size (bytes)	Data size (bytes)	Classical security bits	Quantum security bits
Public-key signatures				
XMSS [93]	64	2500-2820	256	128
SPHINCS+-256s [28]	64	29792	-	255
HFEv [132]	58212-142576	15-20	80-120	-
Public-key encryption				
McEliece [30]	192192-958482	370-828	128-256	-
NTRUEncrypt [89]	604-1022	604-1022	128-256	-
Key exchange				
New Hope [11]	-	1824-2048	-	128
SIDH [58]	-	564	192	128
Classical schemes				
RSA-2048	256	256	112	-
RSA-3072	384	384	128	-
ECC-256	32	32	128	-
ECC-512	64	64	256	-
DH-2048	-	256	112	-
DH-3072	-	512	128	-
ECDH-256	-	32	128	-
ECDH-512	-	64	256	-

Table 2.2: A comparison between the public-key and data structures size for several post-quantum cryptographic systems when compared with classical cryptography. Adapted from [126]

as classical cryptosystems. Finally, the considered code-based cryptosystem seems to perform worse than classical cryptography both in terms of execution time and message size complexity.

2.5.2 Bandwidth

Hardware TPMs are connected to main processors via a bus. LPC buses achieve an average bandwidth of 20.48 Mbps [142]. While SPI does not have a formal standardisation, implementations often go over 10 Mbps. I^{2}C is limited to 3.4 Mbps in High Speed Mode. If the trend that is shown in Fig. 2.1 is verified in practice for the TPM, protocols that now require ~ 0.1 ms to transmit a message, may instead take ~ 1 ms when post-quantum cryptography is introduced. While for previous versions of the TPM, communication represented a small overhead, the same may not be true if certain post-quantum cryptographic primitives, such as those based on lattices, are used to replace them.

2.5.3 Memory Footprint

The sizes of the public-key material and data structures for several post-quantum cryptographic systems have been analysed in [126] and adapted in Table 2.2. Hash-based signatures achieve similar key sizes to those of classical cryptography, but the signatures themselves are much larger. The opposite is true for multivariate-based encryption: the keys are several orders of magnitude larger than classical schemes, but the signature size is relatively small. When considering public-key encryption, it seems lattice-based cryptography, especially the one based on rings, achieves a nice balance between the size of the key and the generated ciphertexts. In general, it appears one would need $\sim 5\times$ more memory to support the keys and data structures of post-quantum cryptography than classical schemes. A possible avenue of research might be on the secure delegation of cryptographic primitives by the TPM to the host, in order to reduce the amount of memory required by the TPM and reduce production costs. Furthermore, it seems hash-based signatures would affect the code size the least, since they may benefit from the optimisations of other operations required by the TPM like the extension of PCRs. Complex operations required by post-quantum cryptography, like Gaussian sampling for RLWE or the evaluation of Vélu's formulae for isogeny-based cryptosystems, may lead to a more significant impact on the code memory size.

2.5.4 Scalability

The usage and extension of existing libraries, such as TPM-SIM [142] or IBM's software TPM 2.0 [80], may accelerate the development of proof-of-concept platforms for testing and evaluation. Moreover, within the context of this project, the designs should be made as open as possible, in order to facilitate future research on the TPM platform.

2.6 Further Requirements

A main aspect regarding the decision of which post-quantum cryptographic primitives will be integrated in FutureTPM is their maturity and acceptance within the scientific community. The FutureTPM project will closely follow the standardisation bodies of both the USA and the EU to achieve this. The NTRU cryptosystem has been standardised by both the IEEE [1] and X9 [15]. There's an ongoing effort to standardise the XMSS by the IETF [92]. Finally, NIST's call to the standardisation of post-quantum cryptography has received 69 submissions, which include lattice, code, multivariate, hash, and isogeny-based cryptography, among others [129]. It might be also useful to consider patent-free systems, to avoid encumbering their usage. While the NTRU-Encrypt system used to be patented, the patents have been released into the public domain in 2017 [126]. A type of hash-based signatures, called Rainbow Signature Scheme has also been patented [145].

Chapter 3

Hash Functions

A hash function is a mapping that takes as input an arbitrarily long bit sequence (message) and outputs a hash value (digest) of fixed length. Moreover, cryptographic hash functions are a special class of hash functions that are designed to satisfy a number of mandatory requirements. Even though they were originally proposed as the input to generate digital signatures, this primitive is used today as a fundamental building block of many cryptographic schemes and protocols. Just to cite some examples: key derivation, authentication, or commitment schemes benefit from the use of cryptographic hash functions.
Instead of a single function, it is usual from a theoretical perspective to deal with a *family* of hash functions indexed by a *key* k. This is not a conventional encryption/decryption key in the usual sense, it is merely an index that specifies the particular hash function used from the family, and, depending on the application, need not be kept secret. For example, one can regard the SHA-256 function as one function from a family, keyed, e.g., by the initial chaining value. Although in practice it is customary to employ keyless hash functions, the reason to introduce the keyed version is that a rigorous treatment of security models for keyless hash functions does not work [137]. In the real world, however, this formality is bypassed by "picking the parameter k" when the hash function is designed, and it is ignored from that point onward. Moreover, hash functions can be used to build message authentication codes (MACs) to provide data origin authentication as well as message integrity. In this case, the definition below is also convenient to establish the security definitions, taking into account that the key k must be kept secret.

Definition 1 *[Keyed Hash Function] A keyed hash function is a pair of PPT algorithms:*

- H.Gen *takes as input a security parameter* 1^λ *and outputs a key* $k \in \mathcal{K}$*.*
- H.Digest *is a collection of functions, indexed by a key* $k \in \mathcal{K}$*, which take as input a message* $m \in \mathcal{M}$*, and outputs a digest value* $h \in \mathcal{H} = \{0,1\}^n$*, where* n *is a polynomial function of* λ*.*

The message space is usually regarded as $\mathcal{M} = \{0,1\}^*$, i.e., the set of all finite bit strings. Nevertheless, some hash functions have a restriction on the input length (for example, $2^{64} - 1$ bits $\approx 10^{6,3}$ TB for SHA-256), which does not have much relevance for practical purposes. Thus, Definition 1 ensures that H is a hash function in the classic sense in that it compresses the input, although, formally, only for messages of length $\geq n$.

3.1 Security Models

The requirements for a cryptographic hash function are that, for a randomly chosen key k, it must be hard to find an input message for a given digest value, and it must be difficult to find two different messages m, m' that produce the same digest. For a comprehensive analysis of security models of hash functions in the QS0 domain, we refer the reader to the work by Rogaway and Shrimpton [138], where a more elaborated discussion can be found.
In a quantum computing scenario, the main source of weakness for hash functions comes from Grover's algorithm. Whereas to invert a hash function by exhaustive search takes $\mathcal{O}\left(2^n\right)$ queries on a classical computer, the problem can be reduced to $\Theta\left(2^{n/2}\right)$ queries for a quantum adversary using this algorithm. This fact essentially halves the bit strength n of any hash function.
In this section, we recall the most well-known security properties that hash functions should have, and for each of them (when possible) we discuss how these notions translate in terms of quantum security domains.

3.1.1 One-Wayness (First-Preimage Resistance)

A one-way function (OWF) is a function that is "easy" to compute but "hard" to invert. The existence of this kind of functions is still an open question and its existence, if proven, would imply that P$\neq$NP. The assumption that cryptographic hash functions are one-way is one very minimal assumption extensively used in computational cryptography and elaborated security proofs. A one-way hash function is also called *first-preimage resistant*.
More formally, we say that a hash function H is (a family of) OWFs if given $k = \mathsf{H.Gen}()$ and a random digest value $h \in \mathcal{H}$, the advantage of any PPT adversary in finding a message $m \in \mathcal{M}$ such that $\mathsf{H.Digest}_k(m) = h$ is negligible in terms of the security parameter. A brute-force approach requires $\mathcal{O}\left(2^n\right)$ queries to the hash function in the QS0 domain.
Similarly, in the QS1 scenario, we have the notion of post-quantum one-way functions (pqOWF) as a basic security assumption. Because the hash function code is public, we expect quantum adversaries to be able to query such a function on a superposition of values. For the same reason, since in the security definition of OWF there is no oracle access, it is enough to define pqOWFs by just replacing PPT adversaries with QPT adversaries.
This implies that, once the adversary learns the key k and implements the public code on a quantum computer, she can execute Grover's algorithm to query the hash function with a superposition of values. As stated above, a brute-force attack reduces to $\Theta\left(2^{n/2}\right)$ queries, in this case.
The QS2 domain reduces to QS1 in this particular case, as the adversary does not require oracle access.

3.1.2 Second-Preimage Resistance

Unless a certain function f is injective, there is going to be at least two different inputs $m \neq m'$ in its domain such that $f(m) = f(m')$. We call the pair (m, m') a *collision for* f. For example, hash functions are usually many-to-one, and not injective unless the domain is very small. In this case we say that a collision occurs if two different messages $m, m' \in \mathcal{M}$ produce the same digest value.
Second-preimage resistance is a strengthening of first-preimage resistance. In first-preimage resistant hash functions, an adversary is given a target element and has to find a preimage for it. In second-preimage resistance, instead, an adversary is given a preimage value, and his task

is to come up with a second, different preimage which causes a collision. I.e., given m, the adversary has to find $m' \neq m$ such that $\mathsf{H}(m) = \mathsf{H}(m')$.
Even if the security notion itself is stronger, from a quantum security point of view nothing changes compared to the case of one-wayness: the fact that the adversary can implement the function's code quantumly implies that in both QS1 and QS2 models a brute-force attack reduces to $\Theta\left(2^{n/2}\right)$ queries.

3.1.3 Collision Resistance

A collision-resistant function (CRF) is a function where it is “difficult” to find *arbitrary* collisions. In the QS0 domain, the security definition for this property states that a hash function H is (a family of) CRFs if given $k = \mathsf{H.Gen}()$, the advantage of any PPT adversary in finding a collision is negligible. Since this time the adversary is not given a target image to start with, she is free to come up with any valid collision. This security notion is hence stronger than second-preimage resistance.
Formaly speaking, this security definition will not work when the hash function is a single function (as opposed to a family of functions). The fact that $|\mathcal{M}| > |\mathcal{K}|$ implies that there exist some m, m' that produce the same digest. Therefore, an efficient PPT adversary exists: she could be defined depending on H's code, and simply hardcode and print m, m'. This is why, as previously discussed, hash functions are formally defined as keyed families.
A simple calculation, following from the birthday paradox, shows that the expected number of queries to find a collision in a hash function is $\mathcal{O}\left(2^{n/2}\right)$ in QS0.
Again, recalling the QS1 domain definition, the adversary has quantum access to classical oracles and local quantum computing capabilities. Therefore, when considering a post-quantum collision-resistant function (pqCRF), a QPT adversary has a reduced need to interact with the hash function. Based on Grover's algorithm, Brassard, Høyer, and Tapp [49] give a quantum algorithm (henceforth called the BHT algorithm) requiring only $\mathcal{O}\left(2^{n/3}\right)$ queries to the hash function to produce a collision with overwhelming probability. Also, for the same reasons above, QS2 reduces to QS1 in this case as well.

3.1.4 Pseudorandomness

A pseudorandom function (PRF) is a family of efficiently-computable functions, indexed by a key k, such that it is computationally indistinguishable from a random oracle without the knowledge of k. In fact, in security reductions, PRFs are usually modeled as random oracles.
In the security definition of this property, as opposed to the two previous properties above, the key $k = \mathsf{H.Gen}()$ is kept secret. Indeed, the chosen function $\mathsf{H.Digest}_k$ can be regarded as an oracle $\mathcal{O}_{\mathsf{H.Digest}_k}$.
In the QS0 domain it is required that the advantage of any PPT adversary in distinguishing $\mathsf{H.Digest}_k$ from a randomly chosen function (among all possible functions mapping messages from $\mathcal{M}$ to $\mathcal{H}$) is negligible. PRFs, being indistinguishable from random functions, can be used as pseudo-random number generators (PRNGs) for a vast majority of the possible keys.
In the QS1 setting, post-quantum pseudorandom functions (pqPRFs) are defined by merely replacing the PPT adversary with a QPT adversary, and keeping the oracle access classical.
In contrast, in the case of QS2, we need to replace the traditional oracle by providing the QPT adversary with quantum oracle access, which we denote $|\mathcal{O}_{\mathsf{H.Digest}_k}\rangle$. That is, the PRF oracle produces outputs on a quantum superposition of inputs. The security definition arising from the new scenarios leads to the concept of quantum-secure pseudorandom functions (qPRFs).

3.1.5 Indifferentiability

The indifferentiability framework introduced by Maurer et al. [112] is an extension of the classical notion of indistinguishability. This notion characterizes exactly when one can replace a subsystem of a cryptosystem by another subsystem without affecting the security. As opposed to the notion of indistinguishability, indifferentiability is applicable in the important case of settings where a possible adversary is assumed to have access to additional information about a system, such as the internal state. This generality has been identified as crucial in the setting of the random oracle methodology, and it leads to a generalization of the related notion of reducibility of one system to another. Essentially, the indifferentiability property states that a hash function does not suffer from structural defects: if a hash function is indifferentiable from a random oracle, then we immediately get that it is OWF, CRF, PRF, etc.
More formally, in QS0, we say that a hash function H (using an internal ideal primitive f) is indifferentiable from a random oracle $\mathcal{O}$ if for any PPT adversary there is a system $\mathcal{P}$ such that the advantage in distinguishing the two setups depicted in Fig. 3.1 is negligible.

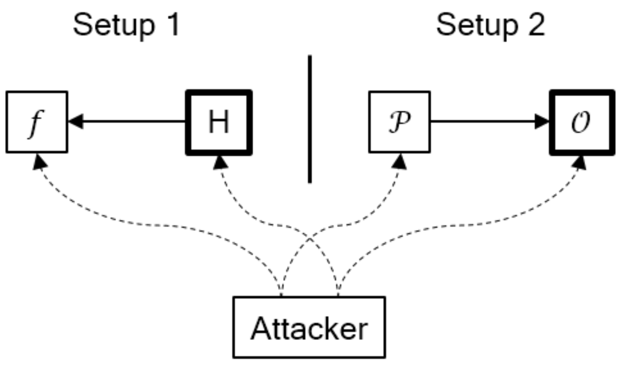

Figure 3.1: The indifferentiability setup

In the first scenario from Fig. 3.1 the first setup represents the hash function construction using an ideal compression function f. The adversary can make queries to both components separately, where H calls f to compute its output. In the second setup consists of a random oracle $\mathcal{O}$ plus an ideal functionality $\mathcal{P}$. Its goal is to simulate the ideal compression function f in such a way that its output looks consistent with what the distinguisher can obtain from the random oracle $\mathcal{O}$ if $\mathcal{P}$ was f and $\mathcal{O}$ was H.
Unfortunately, few results are known about indifferentiability in a quantum setting. In fact, some authors give evidence that perfectly secure quantum indifferentiability is impossible to attain in a wide variety of cases, and that it would be very difficult to build a quantum indifferentiable construction, or to prove the post-quantum security of existing cryptosystems (such as SHA-3) using the indifferentiability framework [55].

3.2 Proposed Candidates

3.2.1 SHA-1 Family

The original specification of the SHA-1 hash function was published in 1995 by NIST [122], as a revised version of a flawed, short-lived proposal, namely SHA-0. Even though it has been the most common hash function, widely used for cryptographic purposes until 2017, it is no longer considered secure, even in the non-quantum QS0 domain.

The first major attack to SHA-1 was presented in 2005, by Wang et al., where they showed that collisions on SHA-1 can be found with complexity less than 2^{69} hash operations, less than the 2^{80} expected operations. Moreover, the technique used was further improved to find collisions in 2^{63} operations. Further attacks have subsequently appeared on reduced versions (i.e., less than 80 rounds) of SHA-1, and, independently, for its compression function. Another noteworthy result of theoretical significance was the 2006 work by De Cannière and Rechberger, where they showed two-block collision for 64-round SHA-1 using only 2^{35} compression function evaluations. This result was later extended further to 73 rounds by Grechnikov in 2010 (remark that the total number of rounds of SHA-1 is 80). The last major breakthrough in terms of breaking the full-round SHA-1 appeared in 2017, when researchers from CWI Amsterdam and Google Research demonstrated that SHA-1 collision attacks have finally become practical by providing the first known instance of a collision using roughly $2^{63.1}$ SHA-1 evaluations.
For this reason, many organizations and standardization bodies have already deprecated their usage in digital certificates and other cryptographic applications. For example, NIST has officially deprecated it in 2011; Google Chrome browser has regarded any website protected with a SHA-1 certificate chains as insecure since January 2017; Firefox deprecated SHA-1 as of February 2017; a similar action was taken by Microsoft regarding Edge and Internet Explorer browsers, by May 2017.

3.2.2 SHA-2 Family

The first draft of the SHA-2 family of hash functions was first published by NIST in 2001, and the latest revision of the standard dates from 2015 [123]. It consists of six functions that produce a digest of size 224, 256, 384 or 512 bits.
This family of functions consists essentially of two original functions, SHA-256 and SHA-512, the rest of them being truncated versions of one of these two functions, with different initial values.
Although no significant cryptographic weakness has already been identified for the SHA-2 family, it shares much of its algorithmic foundations with SHA-1, as it is based in the Merkle–Damgård construction. Currently, the best attacks known break preimage resistance for reduced versions of SHA-256 (52 out of 64 rounds) and SHA-512 (57 out of 80 rounds) [102], and collision resistance for 47 out of 64 rounds of SHA-256 [38]. Moreover, SHA-256 and SHA-512 are vulnerable to length extension attacks, making them insecure for some applications. Most security experts believe that the lifespan of SHA-2 will be similar to that of SHA-1. Therefore, and although NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard, it has already approved in August 2015 a replacement for this primitive, namely SHA-3.

3.2.3 SHA-3 Family

SHA-3 appeared as the result of the NIST hash function competition to create a new hash standard, a process which started in 2006. The outcome of the competition ended with a standard for the new family of hash functions being published by NIST in 2015 [124] .
The SHA-3 family is based on a completely different approach than the Merkle–Damgård construction, and therefore does not share the same mathematical properties than its predecessors. SHA-3 is a subset of the family Keccak, based on the sponge construction. Hence, it is expected to resist cryptographic attacks longer than SHA-2. Also, and unlike its predecessors, SHA-3 was created through a publicly sourced competition.
SHA-3 was designed to provide resistance against collision, preimage, and second preimage attacks that equals or exceeds the resistance that the corresponding SHA-2 functions provide.

The SHA-3 functions are also designed to resist other attacks, such as length-extension attacks, that would be resisted by a random function of the same output length, in general providing the same security strength as a random function, up to the output length.
Some published works show how to perform attacks on reduced-round versions of SHA-3. For example, preimage attacks for 8-round SHA3-512 with $2^{511.5}$ time and 2^{508} memory [119], or collision-finding attacks for 3-round SHA3-384 and SHA3-512, and 8-round SHA3-256 [65]. All these attacks are way far to pose any significant threat.

3.2.4 Other Candidates

BLAKE2

BLAKE is one of the hash functions that was submitted to the NIST SHA-3 hash function competition, and it was selected as one of the five finalists. An improved version of BLAKE, called BLAKE2 was announced in late 2012 [20, 19]. They rely on a core algorithm borrowed from the ChaCha stream cipher. BLAKE2 supports keying, salting, personalization, and hash tree modes. It comes in two flavours: BLAKE2b (or just BLAKE2), optimized for 64-bit platforms (including NEON-enabled ARMs) and produces digests of any size between 1 and 64 bytes, and BLAKE2s which is optimized for 8- to 32-bit platforms and produces digests of any size between 1 and 32 bytes. BLAKE2 also includes the BLAKE2x variants [21], which can produce digests of arbitrary length, using a similar extract-then-expand scheme (but not identical) to that of HKDF key generation function. There are also parallel versions designed for increased performance on multi-core processors. BLAKE2 has been specified in IETF RFC7693 in 2015 [71].
BLAKE2 is claimed to be at least as secure as SHA-3 by its authors. It relies on (essentially) the same core algorithm as BLAKE, which has been intensively analyzed since 2008 within the SHA-3 competition. The best academic attack on BLAKE (and BLAKE2) works on a reduced version with 2.5 rounds, whereas BLAKE2b does 12 rounds, and BLAKE2s does 10 rounds. But even this attack is not practical: it only shows for example that with 2.5 rounds, the preimage security of BLAKE2b is downgraded from 512 bits to 481 bits, or that the collision security of BLAKE2s is downgraded from 128 bits to 112 bits (which is similar to the security of 2048-bit RSA).
It is worth to mention that BLAKE2 has been adopted in many projects and systems: Argon2 (the winner of the Password Hashing Competition), WolfSSL, OpenSSL, Wireward, Botan, Crypto++, Noise (cryptographic protocol used in WhatsApp), Cifra Extrema, Bouncy Castle, Peerio, 8th, librsync, etc. It is claimed to achieves very fast speeds, compared to the current SHA-3 standard.

SM3

SM3 is a cryptographic hash function [167] approved as a Chinese National Standard by the Organization of State Commercial Administration of China (OSCCA) [150] in 2016, and authorized for the use in commercial cryptography schemes in China. Under the Chinese law, OSCCA requires that any company or individual selling encryption products in China to first obtain its approval. As a result the choice of encryption products lawfully available to buy and use in China is limited. Moreover, no foreign encryption technology is allowed to be sold in China, even by OSCCA-approved retailers.
The SM3 hash function is therefore the result of the Chinese National Cryptographic Administration Bureau, in releasing the specification of a Trusted Cryptograpy Module, in order to detail a cryptoprocessor to be used within the Trusted Computing framework in China.
SM3 has a Merkle-Damgård construction and is similar to SHA-2. It is claimed that it has several strengthening features, including a more complex step function and stronger message depen-

dency than SHA-256. Unfortunately there's no explanation for how the changes made to SHA-2 strengthen it. SM3 has 64 rounds and produces an output hash value of 256 bits long, based on 512-bit input message blocks.
The amount of cryptanalytic results on SM3 is limited, compared to other hash function standards. To cite some results, Kircanski et al. [103] presented distinguishers for its compression function on reduced-round versions, with complexity ranging from $\mathcal{O}\left(2^{14.4}\right)$ for 32 rounds, to $\mathcal{O}\left(2^{117.1}\right)$, for 35 rounds. Zhow et al. [173] show the feasability of preimage attacks on SM3 for 30 rounds with complexity $\mathcal{O}\left(2^{249}\right)$. Also, practical collision attacks have been presented by Mendel et al. [116] for 20 rounds SM3, and also for 24 rounds with free selection of initial chaining value.

Lightweight hash functions

Lightweight hash functions are specially designed for systems and solutions using low-cost 8-bit CPUs, such as passive RFID tags. The main challenge in this area is the design of cryptographic primitives and protocols that meet the system requirements within the devices capabilities, which are usually extremely limited.
For the security levels, there is always a trade-off between security and cost when it comes to design a cryptographic algorithm. Much of the effort on the design of these lightweight primitives is usually put on the cost. Namely, these kinds of hash functions have been developed to reduce cycle rate, throughput rate, power consumption and chip area. The sacrifice for the security is usually justified due the non-criticality of the data transmitted by a single RFID chip. A small device with a reduced security is still secure enough because an attacker must compromise the hashed data of all RFIDs. This is still a rather difficult challenge, even with a lightweight hash function. Typically in IoT scenarios, the availability might be more important than the confidentiality. But this depends on the use case.
However, while attaining a reasonable security level for an average attacker on these scenarios, they should not be considered a substitute for general hash functions, such as SHA-3, when high security levels are required. Moreover, lightweight hash functions have received less attention from cryptanalysts than lightweight block ciphers.

PHOTON is a family of hash functions [85] that has been approved as part of the ISO standard as a lightweight hash function optimized for hardware implementations [96]. It is claimed by its authors to be the most compact hash function known so far, reaching areas very close to the theoretical optimum (derived from the minimal internal state memory size).
PHOTON uses the sponge functions framework in order to keep the internal memory size as low as possible. The framework is extended so as to provide reasonable trade-offs in hardware between preimage security and small messages hashing speed (small message scenario is a classical use-case and can be problematic for sponge functions because of their squeezing process that can be very slow in practice). The internal permutations of PHOTON can be seen as AES-like primitives especially derived for hardware.
Several variants of PHOTON can be specified. Each variant is defined by its internal permutation size $t = c + r$, where c and r denote the capacity and the bitrate, respectively, of the underlying sponge construction. For a fixed permutation size t, the choice of c and r provides a security efficiency trade-off. PHOTON-t denotes the variant using a t-bit internal permutation. Therefore, depending on the available computing power, the PHOTON family can be precisely instantiated for several security levels ranging from 64-bit preimage resistance security to 128-bit collision resistance security. The standard [96] specifies five variants of PHOTON-t, for $t \in \{100, 144, 196, 256, 288\}$. PHOTON-100 does not provide the minimum security strength as

required in ISO/IEC 29192-1. It shall not be used as a general purpose hash function. PHOTON-144 does not provide the minimum security strength for collision resistance and second preimage resistance as required in ISO/IEC 29192-1. It shall only be used in applications where collision resistance and second preimage resistance are not required.
The features of the internal permutation make that the security analysis for this hash function family can benefit from all previous extensive cryptanalysis performed on AES and on AES-based hash functions. While 8 rounds over 12 of the internal permutations of PHOTON can be distinguished from a random permutation, the authors provide strong arguments that this is very unlikely to be much improved.

Lesamnta-LW The Lesamnta-LW family [88] is a ligtweight family of hash functions which is approved by ISO/IEC as an optimized hash function for software implementations [96]. The most important aspects considered in the design of Lesamnta-LW are to have security reductions, to have a small hardware footprint, and to have a low working memory requirement for software.
Lesamnta-LW has a 256-bit output, and its domain extension is the strengthened Merkle-Damgård construction. Its underlying component is an AES-based block cipher taking a 256-bit plaintext and a 128-bit key. The compression function is a new mode of a block cipher, called the LW1 mode, which enables to provide proofs reducing the security of Lesamnta-LW to that of the underlying block cipher. As in the case of PHOTON, the block cipher employed is based on AES in order to gain confidence in its security. Moreover, Lesamnta-LW is a lightweight variant of a hash function that was accepted for Round One of the NIST SHA-3 competition. Even though it did not manage to pass to Round Two, the hash function has neither been conceded by the submitters nor has had substantial cryptographic weaknesses found.
That being said, is a distinguisher for the full internal block cipher of Lesamnta-LW [140], however its consequences for the hash function itself are unclear.

3.3 Candidates Comparison

In Table 3.1 we present a comparison between the different candidates of families of hash functions. We present the bit security figures for the different notions in QS0, QS1 and QS2. Speed values are presented in cycles per byte (cps) for a Skylake (506e3) 2015 Intel Core i5-6600 processor. The figures have been taken from the median values from eBACS benchmark for hash functions [165] for long messages, or estimated from a variety of sources [162, 22, 85, 88] when unavailable.

3.4 Open Issues

There are several open issues in terms of security notions in quantum scenarios concerning hash functions.
First, in connection with Section 3.1.4, it is worth to mention that there is currently some ongoing discussion on how to define correctly QS2 security for MACs. We refer the reader, e.g., to [5] for a further discussion.
Also, for the case of QS1 and QS2, it is an open problem how to define the notion of indifferentiability introduced in Section 3.1.5. For example, in [31] it was shown that the sponge construction for hash functions is indifferentiable from a random oracle, assuming that the internal compres-

Name	Outp. size[a]	Int. state[a]	Block size[a]	Max. inp. size[a]	Nr[b]	Collision capacity[a]		Preimage capacity[a]		Perf.[c]	Struct.[d]	Year
						QS0	QS1, QS2	QS0	QS1, QS2			
SHA-224	224	256	512	$2^{64}-1$	64	112	74.7	224	112	7.62	MD	2004
SHA-512/224	224	512	1024	$2^{128}-1$	80	112	74.7	224	112	5.12	MD	2012
SHA-256	256	256	512	$2^{64}-1$	64	128	85.3	256	128	7.63	MD	2001
SHA-512/256	256	512	1024	$2^{128}-1$	80	128	85.3	256	128	5.12	MD	2012
SHA-384	384	512	1024	$2^{128}-1$	80	192	128.0	384	192	5.12	MD	2001
SHA-512	512	512	1024	$2^{128}-1$	80	256	170.7	512	256	5.06	MD	2001
SHA3-224	224	1600	1152	∞	24	112	74.7	224	112	8.12	PS	2015
SHA3-256	256	1600	1088	∞	24	128	85.3	256	128	8.59	PS	2015
SHA3-384	384	1600	832	∞	24	192	128.0	384	192	11.06	PS	2015
SHA3-512	512	1600	576	∞	24	256	170.7	512	256	15.88	PS	2015
SHAKE128	n (any)	1600	1344	∞	24	$\min(n/2, 128)$	$\min(n/3, 128)$	$\geq \min(n, 128)$	$\geq \min(n/2, 128)$	7.08	PS	2015
SHAKE256	n (any)	1600	1088	∞	24	$\min(n/2, 256)$	$\min(n/3, 256)$	$\geq \min(n, 256)$	$\geq \min(n/2, 256)$	8.59	PS	2015
BLAKE2b-160	160	1024	512	$2^{128}-1$	12	80	53.3	160	80	3.33	HAIFA	2012
BLAKE2b-256	256	1024	512	$2^{128}-1$	12	128	85.3	256	128	3.33	HAIFA	2012
BLAKE2b-384	384	1024	512	$2^{128}-1$	12	192	128.0	384	192	3.33	HAIFA	2012
BLAKE2b-512	512	1024	512	$2^{128}-1$	12	256	170.7	512	256	3.33	HAIFA	2012
BLAKE2s-128	128	512	256	$2^{64}-1$	10	64	42.7	128	64	4.87	HAIFA	2012
BLAKE2s-160	160	512	256	$2^{64}-1$	10	80	53.3	160	80	4.87	HAIFA	2012
BLAKE2s-224	224	512	256	$2^{64}-1$	10	112	74.7	224	112	4.87	HAIFA	2012
BLAKE2s-256	256	512	256	$2^{64}-1$	10	128	85.3	256	128	4.87	HAIFA	2012
SM3	256	256	512	$2^{64}-1$	64	128	85.3	256	128	(9.81)	MD	2007
PHOTON-100	80	100	16	∞	12	40	26.7	64	40	(21.96)	PS	2011
PHOTON-144	128	144	16	∞	12	64	42.7	112	64	(32.43)	PS	2011
PHOTON-196	160	196	36	∞	12	80	53.3	124	80	(16.64)	PS	2011
PHOTON-256	224	256	32	∞	12	112	74.7	192	112	(21.21)	PS	2011
PHOTON-288	256	288	32	∞	12	128	85.3	224	128	(16.22)	PS	2011
Lesamnta-LW	256	256	128	$2^{64}-1$	64	128	85.3	256	128	(12.24)	MD	2012

[a] Measured in bits.
[b] Number of rounds of the compression function.
[c] (Estimated) performance measured in cycles per byte (cpb) for long messages, on a Skylake (506e3) 2015 Intel Core i5-6600 processor [165].
[d] Structure of the hash function construction: Merkle-Damgård (MD) [117, 61], permutation sponge (PS) [32], or hash iterative framework (HAIFA) [33].

Table 3.1: Comparison of the candidate families of hash functions.

sion function f is a random oracle or an (invertible) random permutation. However, the proof from [31] works only in the classical case.

Currently, quantum indifferentiability in QS1 and QS2 is a topic of active research and controversy. Whereas some works seem to point out that the sponge and the Merkle-Damgård constructions for hash functions exhibit desirable properties in terms of this property, other authors are quite skeptical about the fact that a suitable notion of quantum indifferentiability can be found at all. We refer the reader to some of the most recent and relevant references of ongoing work in the topic for a further discussion [170, 59, 55].

Chapter 4

Block Ciphers

Block ciphers are building blocks used for symmetric-key encryption. They work by splitting the input data into *blocks* of a fixed length, and operating sequentially on these blocks using a *secret key* of a fixed size. The bitsize of the blocks and the secret key are called *blocksize* and *keysize*, respectively. The exact way block ciphers operate on sequential blocks is specified by a *mode of operation*, which is a set of rules determining how a single secret key is applied to different consecutive blocks.

Definition 2 (Block Cipher) *A* block cipher $\mathcal{B}$ *with blocksize* b *and keysize* λ *is a pair of DPT algorithms:*

- $\mathsf{BC.Enc}$ *takes as input a block* $X \in \{0,1\}^b$ *and a secret key* $\mathsf{k} \in \{0,1\}^\lambda$*, and outputs another block* $Y \in \{0,1\}^b$*. We write this as* $Y \leftarrow \mathsf{Enc}_\mathsf{k}(X)$
- $\mathsf{BC.Dec}$ *takes as input a block* $Y \in \{0,1\}^b$ *and a secret key* $\mathsf{k} \in \{0,1\}^\lambda$*, and outputs another block* $X \in \{0,1\}^b$*. We write this as* $X := \mathsf{Dec}_\mathsf{k}(Y)$*.*

Correctness of the block cipher requires that:

1. $\mathsf{BC.Dec} \circ \mathsf{BC.Enc} = \mathsf{Id}$.

Block ciphers are crucial cryptographic primitives. They are used for bulk encryption and decryption of data, and are also used in combination with other primitives, e.g., public-key encryption. They are usually lightweight in terms of resources, and very fast. Usually they do not increase the size of the data during encryption.

4.1 Security Models

Block ciphers security can be complex. At a minimum, secrecy of the encrypted data must be ensured. However, other security property are often desirable, mainly regarding integrity of data. In terms of quantum security, we will adopt the usual classification with QS0 being classical security property, QS1 being post-quantum, and QS2 being superposition-based quantum security.
The first step in order to analyze the security of block ciphers is to define the final cryptographic object that block ciphers realize: a *symmetric-key encryption scheme (SKES)*. A block cipher plus a mode of operation produces a SKES. Not all SKES are constructed from block ciphers, but for practical purposes (and especially for FutureTPM) most are. The security properties we are interested in are actually the ones that the final SKES should provide: we will hence first introduce the right security notions for SKES, and then analyze the desirable security properties of block ciphers and modes of operations leading to the aforementioned notions.

Definition 3 (Symmetric-Key Encryption Scheme (SKES)) *A* symmetric-key encryption scheme (SKES) $\mathcal{E}$ *with plaintext space* $M = \{0,1\}^m$ *and ciphertext space* $C = \{0,1\}^c$ *is a tuple of PPT algorithms:*

- Kgen *takes as input a (unary representation of) security parameter* 1^λ *and outputs a secret key* $\mathsf{k} \in \{0,1\}^{k(\lambda)}$ *for some polynomially bounded function* k.
- Enc *takes as input a message* $\mathsf{msg} \in M$ *and a secret key* k*, and outputs a ciphertext* $c \in C$*. We write this as* $c \leftarrow \mathsf{Enc}_\mathsf{k}(m)$
- Dec *takes as input an element* $y \in C$ *and a secret key* k*, and outputs an element* $x \in M \cup \{\perp\}$*. The algorithm* Dec *is deterministic. We write this as* $x := \mathsf{Dec}_\mathsf{k}(y)$.

Correctness of the SKES requires that:

1. $\mathsf{Dec} \circ \mathsf{Enc} = \mathsf{Id}$*; and*
2. $\mathsf{Dec}_\mathsf{k}(y) = \perp, \forall \mathsf{k}, \forall y \notin \mathsf{Supp}(\mathsf{Enc}_\mathsf{k})$.

4.1.1 Indistinguishability of Ciphertexts

We start by defining the basic security property that a SKES has to guarantee: *indistinguishability of ciphertexts (IND)*. In this definition, an adversary against the scheme is modeled as a *pair* of machines $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ (for example, in QS0, these machines are modeled as PPT Turing machines or uniform families of circuits). $\mathcal{A}_1$ and $\mathcal{A}_2$ are allowed to share an internal classical state.
Security is given in terms of an *indistinguishability game* against a *challenger algorithm* $\mathcal{C}$. In this game, on input the security parameter λ, $\mathcal{C}$ generates a fresh key $\mathsf{k} \leftarrow \mathsf{Kgen}(1^\lambda)$, and then $\mathcal{A}_1$ is executed. $\mathcal{A}_1$ chooses two messages msg_0 and msg_1 of his choice (of the same length), and sends them both to $\mathcal{C}$. $\mathcal{C}$ will draw a bit uniformly at random $b \xleftarrow{\$} \{0,1\}$, then it encrypts one of the two messages $y \leftarrow \mathsf{Enc}_\mathsf{k}(\mathsf{msg}_b)$, while the other message is discarded. The ciphertext y is then sent to $\mathcal{A}_2$, whose goal is then to guess which one of the two messages was encrypted by outputting the correct secret bit b. We say that $\mathcal{A}$ wins the IND game if this guess is correct. For a scheme to be secure, we want that no adversary can win this game with probability substantially better than guessing, even if the adversary has additional power, modeled as access to additional encryption and decryption oracles.

Definition 4 (Indistinguishability in QS0) *A SKES* $\mathcal{E}$ *has* indistinguishable ciphertexts *(or, it is IND secure) iff, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, it holds:*

$$\left| \Pr[\mathcal{A} \textit{ wins the IND game}] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Moreover, $\mathcal{E}$ *has* indistinguishable ciphertexts under chosen-plaintext attack *(or, it is IND-CPA secure) iff, in addition to IND,* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *have oracle access to* Enc_k.
Moreover, $\mathcal{E}$ *has* indistinguishable ciphertexts under non-adaptive chosen-ciphertext attack *(or, it is IND-CCA1 secure) iff, in addition to IND-CPA,* $\mathcal{A}_1$ *has oracle access to* Dec_k.
Moreover, $\mathcal{E}$ *has* indistinguishable ciphertexts under adaptive chosen-ciphertext attack *(or, it is IND-CCA2 secure) iff, in addition to IND-CCA1,* $\mathcal{A}_2$ *has oracle access to* $\mathsf{Dec}^*_\mathsf{k}$*, where* $\mathsf{Dec}^*_\mathsf{k}$ *is an oracle defined within the IND game itself as:*

$$\mathsf{Dec}^*_\mathsf{k}(y) := \begin{cases} \perp, & \textit{if } y = c \\ \mathsf{Dec}_\mathsf{k}(y), & \textit{otherwise} \end{cases}$$

where c is the challenge ciphertext produced in the IND game.

Defining indistinguishability for other quantum security domains requires thinking how to model the quantum adversary. In the case of QS1, this process is pretty straightforward: we need to consider quantum adversaries (modelled as QPT machines, or uniform families of poly-depth quantum circuits, as usual). Regarding the type of oracle access to Enc and Dec, it is sufficient to notice that, in reality, these oracles are implemented by the (classical) challenger $\mathcal{C}$, because they depend on the secret key k, which is never in possession of the adversary during the IND game. This means that the access to these oracles remains classical, leading to the following definition.

Definition 5 (Indistinguishability in QS1) *A SKES $\mathcal{E}$ has* post-quantum indistinguishable ciphertexts *(or, it is pqIND secure) iff, for all QPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds:*

$$\left| \Pr[\mathcal{A} \textit{ wins the IND game}] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Moreover, $\mathcal{E}$ has post-quantum indistinguishable ciphertexts under chosen-plaintext attack *(or, it is pqIND-CPA secure) iff, in addition to pqIND, $\mathcal{A}_1$ and $\mathcal{A}_2$ have (classical) oracle access to* Enc_k*.*
Moreover, $\mathcal{E}$ has post-quantum indistinguishable ciphertexts under non-adaptive chosen-ciphertext attack *(or, it is pqIND-CCA1 secure) iff, in addition to pqIND-CPA, $\mathcal{A}_1$ has (classical) oracle access to* Dec_k*.*
Moreover, $\mathcal{E}$ has post-quantum indistinguishable ciphertexts under adaptive chosen-ciphertext attack *(or, it is pqIND-CCA2 secure) iff, in addition to pqIND-CCA1, $\mathcal{A}_2$ has oracle access to* $\mathsf{Dec}^*_\mathsf{k}$*, where* $\mathsf{Dec}^*_\mathsf{k}$ *is defined as in Definition 4.*

The QS2 case is more involved. The adversaries are still modeled as QPT machines, but now they get quantum oracle access to Enc and Dec. Let's consider just Enc for now. The canonical way to define quantum oracle access is to give to the adversary oracle access to a unitary gate $U_{\mathsf{Enc}_\mathsf{k}}$ defined as:

$$U_{\mathsf{Enc}_\mathsf{k}} |x, y\rangle := |x, y \oplus \mathsf{Enc}_\mathsf{k}(x)\rangle .$$

The same notation and definition applies for Dec, except that some care must be taken when defining the "rejecting" $\mathsf{Dec}^*_\mathsf{k}$ oracle for the CCA2 game: the quantum version of this oracle is defined as

$$U_{\mathsf{Dec}^*_\mathsf{k}} |x, y\rangle := \begin{cases} |\perp\rangle , & \text{if } x = c \\ |x, y \oplus \mathsf{Dec}_\mathsf{k}(x)\rangle , & \text{otherwise} \end{cases}$$

where c is the challenge ciphertext produced in the IND game, and the equality check is perfomed "in quantum superposition". The structure of the game itself, however, is the same as the classical IND, as in QS1, only this time the oracles are quantum, and they are just "initialized" (instead of "emulated") by the classical challenger $\mathcal{C}$. The corresponding QS2 notions of *indistinguishability under quantum chosen plaintext/ciphertext attack* have been first introduced in [44].

Definition 6 (Indistinguishability in QS2 (Classical Challenge)) *A SKES $\mathcal{E}$ has* indistinguishable ciphertexts under quantum chosen-plaintext attack *(or, it is IND-qCPA secure) iff, for all QPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds:*

$$\left| \Pr[\mathcal{A}^{|\mathsf{Enc}_\mathsf{k}\rangle} \textit{ wins the IND game}] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

quantum security domain	base security game	security notions
QS0	IND	IND-CPA, IND-CCA1, IND-CCA2
QS1	IND	pqIND-CPA, pqIND-CCA1, pqIND-CCA2
QS2	IND	IND-qCPA, IND-qCCA1, IND-qCCA2
QS2	qIND	qIND-CPA, qIND-CCA1, qIND-CCA2

Figure 4.1: security domain classification for indistinguishability for SKES.

Moreover, $\mathcal{E}$ has indistinguishable ciphertexts under quantum non-adaptive chosen-ciphertext attack *(or, it is IND-qCCA1 secure) iff, in addition to IND-qCPA, $\mathcal{A}_1$ has (quantum) oracle access to $|\mathsf{Dec_k}\rangle$.*
Moreover, $\mathcal{E}$ has indistinguishable ciphertexts under quantum adaptive chosen-ciphertext attack *(or, it is IND-qCCA2 secure) iff, in addition to IND-qCCA1, $\mathcal{A}_2$ has quantum oracle access to $|\mathsf{Dec^*_k}\rangle$, where $\mathsf{Dec^*_k}$ is defined as in Definition 4.*

It is important to notice that the security notions from Definition 6 qualify as QS2, because of the quantum oracle access to encryption and decryption (which, in the symmetric-key scenario, are implemented by a code which is private and normally not available to the adversary). However, also notice that the "basic" security experiment is the same (the IND game) and in particular the challenge query is still classical. This apparent limitation actually arises from technical considerations discussed in [44, 77]. In [77] alternative security definitions in QS2 have been proposed, which overcome this limitation by changing the structure of the security game (called qIND) which makes sense in respect to cryptographic schemes implemented by means of non-standard quantum unitaries. A full description of the qIND experiment can be found in [76], but it basically boils down to an IND game where also the challenger $\mathcal{C}$ and the challenge query are quantum: the adversary sends to the challenger two quantum states φ_0, φ_1, which the challenger interprets as superpositions of messages, and then selects one of the two, encrypts it, and sends it back to the adversary. The resulting security notions are still in QS2, but are strictly stronger.

Definition 7 (Indistinguishability in QS2 (Quantum Challenge)) *A SKES $\mathcal{E}$ has* quantum indistinguishable ciphertexts under chosen-plaintext attack *(or, it is qIND-CPA secure) iff, for all QPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds:*

$$\left| \Pr[\mathcal{A}^{|\mathsf{Enc_k}\rangle} \textit{ wins the qIND game}] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Moreover, $\mathcal{E}$ has quantum indistinguishable ciphertexts under non-adaptive chosen-ciphertext attack *(or, it is qIND-CCA1 secure) iff, in addition to qIND-CPA, $\mathcal{A}_1$ has (quantum) oracle access to $|\mathsf{Dec_k}\rangle$.*

The above security notions do not extend naturally to the CCA2 case, for technical reasons discussed, e.g., in [3, 77]. However, recently, [4] introduced new results (given in the "fully quantum domain", or QS3, which we do not address in this document) which extend naturally to the QS2 case, and allow to overcome the above difficulties by defining a quantum extension of qIND-CCA1 to the CCA2 case. The resulting notion is called qIND-CCA2.
The relations between all these indistinguishability notions are summarized in Figure 4.1 and 4.2. The weakest form of quantum security for the indistinguishability of ciphertexts for SKES is therefore pqIND-CPA, while the strongest one is qIND-CCA2.

$$\begin{tikzcd}
\text{qIND-CCA2} \arrow[r] \arrow[d] & \text{qIND-CCA1} \arrow[r] \arrow[d] & \text{qIND-CPA} \arrow[r] \arrow[d] & \text{qIND} \arrow[ddd] \\
\text{IND-qCCA2} \arrow[r] \arrow[d] & \text{IND-qCCA1} \arrow[r] \arrow[d] & \text{IND-qCPA} \arrow[d] & \\
\text{pqIND-CCA2} \arrow[r] \arrow[d] & \text{pqIND-CCA1} \arrow[r] \arrow[d] & \text{pqIND-CPA} \arrow[d] & \\
\text{IND-CCA2} \arrow[r] & \text{IND-CCA1} \arrow[r] & \text{IND-CPA} \arrow[r] & \text{IND}
\end{tikzcd}$$

Figure 4.2: quantum security notion hierarchy for SKES. An arrow means "strictly implies".

4.1.2 Integrity

The notion of *integrity* requires that a malicious adversary, even if unable to decrypt ciphertexts, should not be able even just to manipulate or create valid encrypted data. This notion comes in two flavors.

Integrity of Plaintexts.

In this case it is required that no adversary, even if able to see encryptions of plaintexts of his choice, can generate another ciphertext that successfully decrypts to a *fresh plaintext*. More formally, in the INT-PTXT experiment, an adversary is given access to an encryption oracle (initialized with a fresh key), and eventually has to output a ciphertext. We say that the adversary *wins* the INT-PTXT game if that ciphertexts correctly decrypts to a fresh plaintext (i.e., one which was not queried before to the encryption oracle).

Definition 8 (Integrity of Plaintexts in QS0) *A SKES $\mathcal{E}$ has* integrity of plaintexts *(or, it is INT-PTXT secure) iff, for all PPT adversaries $\mathcal{A}$, it holds:*

$$\left|\Pr[\mathcal{A}^{\mathsf{Enc_k}} \textit{ wins the INT-PTXT game }]\right| \leq \mathsf{negl}(\lambda).$$

In the QS1 setting, the PPT adversary is replaced by a QPT adversary, as usual, but the structure of the experiment and the oracle access remain the same.

Definition 9 (Integrity of Plaintexts in QS1) *A SKES $\mathcal{E}$ has* post-quantum integrity of plaintexts *(or, it is pqINT-PTXT secure) iff, for all QPT adversaries $\mathcal{A}$, it holds:*

$$\left|\Pr[\mathcal{A}^{\mathsf{Enc_k}} \textit{ wins the INT-PTXT game }]\right| \leq \mathsf{negl}(\lambda).$$

The QS2 setting is more involved. The reason is that here the adversary has quantum access to the encryption oracle, and then it becomes hard to define what a "fresh" plaintext should be. We will follow the approach of [43] for MACs, and define a different "quantum forgery" game. In this game (which we call INT-qPTXT) the adversary has quantum access to $\mathsf{Enc_k}$, as from the QS2 model, and he is allowed to perform an arbitrary (but polynomial in λ) number q of quantum queries to such oracle. However, the goal of the adversary this time is to produce $q+1$ valid (classical) ciphertexts, such that they decrypt to $q+1$ distinct (classical) plaintexts. It is easy to see that if we re-adapt this experiment to the QS0 and QS1 cases, the resulting security definitions are actually equivalent to INT-PTXT and pqINT-PTXT respectively. However, this "one-more forgery" approach has the advantage that it is also easy to translate to the QS2 setting.

Definition 10 (Integrity of Plaintexts in QS2) *A SKES $\mathcal{E}$ has* integrity of quantum plaintexts *(or, it is INT-qPTXT secure) iff, for all QPT adversaries $\mathcal{A}$, it holds:*

$$\left|\Pr[\mathcal{A}^{|\mathsf{Enc}_\mathsf{k}\rangle} \textit{ wins the INT-qPTXT game }]\right| \leq \mathsf{negl}(\lambda).$$

Integrity of Ciphertexts.

A stronger integrity notion is *integrity of ciphertexts* (sometimes also called *ciphertext unforgeability*). The INT-CTXT experiment here works exactly like the INT-PTXT one, except that this time the adversary is required to output *any fresh ciphertext that decrypts correctly*, regardless of the plaintext produced. It is a stronger notion because here the adversary is allowed to exploit some form of "malleability" in the ciphertexts, where it could be possible to alter an original ciphertext without changing the underlying plaintext. Think as an example an INT-PTXT encryption scheme which always appends a random bit to the ciphertexts produced, and such bit is discarded during decryption. Clearly such scheme cannot be INT-CTXT, because an adversary could always manipulate a valid ciphertext by flipping the redundant bit.
More formally, in the INT-CTXT experiment, an adversary is given access to an encryption oracle (initialized with a fresh key), and eventually has to output a ciphertext. We say that the adversary *wins* the INT-CTXT game if that ciphertexts correctly decrypts, and it is a fresh one (i.e., one which was not produced before by the encryption oracle).

Definition 11 (Integrity of Ciphertexts in QS0) *A SKES $\mathcal{E}$ has* integrity of ciphertexts *(or, it is INT-CTXT secure) iff, for all PPT adversaries $\mathcal{A}$, it holds:*

$$\left|\Pr[\mathcal{A}^{\mathsf{Enc}_\mathsf{k}} \textit{ wins the INT-CTXT game }]\right| \leq \mathsf{negl}(\lambda).$$

In the QS1 setting, the PPT adversary is replaced by a QPT adversary, as usual, but the structure of the experiment and the oracle access remain the same.

Definition 12 (Integrity of Ciphertexts in QS1) *A SKES $\mathcal{E}$ has* post-quantum integrity of ciphertexts *(or, it is pqINT-CTXT secure) iff, for all QPT adversaries $\mathcal{A}$, it holds:*

$$\left|\Pr[\mathcal{A}^{\mathsf{Enc}_\mathsf{k}} \textit{ wins the INT-CTXT game }]\right| \leq \mathsf{negl}(\lambda).$$

In the QS2 setting we again adopt the "one-more forgery" paradigm. We define a new security game (which we call INT-qCTXT) where the adversary has quantum access to Enc_k, as from the QS2 model, and he is allowed to perform an arbitrary (but polynomial in λ) number q of quantum queries to such oracle. However, the goal of the adversary this time is to produce $q + 1$ *valid and distinct* (classical) ciphertexts.

Definition 13 (Integrity of Ciphertexts in QS2) *A SKES $\mathcal{E}$ has* integrity of quantum ciphertexts *(or, it is INT-qCTXT secure) iff, for all QPT adversaries $\mathcal{A}$, it holds:*

$$\left|\Pr[\mathcal{A}^{|\mathsf{Enc}_\mathsf{k}\rangle} \textit{ wins the INT-qCTXT game }]\right| \leq \mathsf{negl}(\lambda).$$

It is easy to see that if we re-adapt this experiment to the QS0 and QS1 cases, the resulting security definitions are actually equivalent to INT-CTXT and pqINT-CTXT respectively. The relations between all these integrity notions is summarized in Figure 4.3 and 4.4. The weakest form of quantum security for the integrity of SKES is therefore pqINT-PTXT, while the strongest one is INT-qCTXT.

quantum security domain	integrity of plaintexts	integrity of ciphertexts
QS0	INT-PTXT	INT-CTXT
QS1	pqINT-PTXT	pqINT-CTXT
QS2	INT-qPTXT	INT-qCTXT

Figure 4.3: security domain classification of integrity for SKES.

$$\begin{tikzcd}
\text{INT-qCTXT} \arrow[r] \arrow[d] & \text{INT-qPTXT} \arrow[d] \\
\text{pqINT-CTXT} \arrow[r] \arrow[d] & \text{pqINT-PTXT} \arrow[d] \\
\text{INT-CTXT} \arrow[r] & \text{INT-PTXT}
\end{tikzcd}$$

Figure 4.4: quantum security notion hierarchy of integrity for SKES. An arrow means “strictly implies”.

4.1.3 Pseudorandomness

A natural property to be expected by most block ciphers (but technically speaking not necessary for arbitrary SKES) is that of *pseudorandomness*. That is: it must be hard for an adversary to distinguish correctly generated ciphertexts from random strings. It turns out that such definition is *exactly equivalent* to the definition of pseudorandom function (in QS0, QS1, and QS2) when viewing the encryption procedure of the SKES as an arbitrary function. Therefore we will use the latter terminology.

Definition 14 (Pseudorandomness of Encryption in QS0) *A SKES $\mathcal{E}$ has* pseudorandom ciphertexts *(or, it is PRF secure) iff, for all PPT adversaries $\mathcal{A}$, it holds:*

$$\left| \Pr_{\mathsf{k} \leftarrow \mathsf{Kgen}(1^\lambda)} [\mathcal{A}^{\mathsf{Enc}_\mathsf{k}} \to 1] - \Pr_{\mathcal{O} \xleftarrow{\$} C^M} [\mathcal{A}^{\mathcal{O}} \to 1] \right| \leq \mathsf{negl}(\lambda),$$

where C^M is the space of all functions mapping bitstrings in M to bitstrings in C

As usual, in the QS1 setting, the PPT adversary is replaced by a QPT adversary, but the structure of the experiment and the oracle access remain the same.

Definition 15 (Pseudorandomness of Encryption in QS1) *A SKES $\mathcal{E}$ has* post-quantum pseudorandom ciphertexts *(or, it is pqPRF secure) iff, for all QPT adversaries $\mathcal{A}$, it holds:*

$$\left| \Pr_{\mathsf{k} \leftarrow \mathsf{Kgen}(1^\lambda)} [\mathcal{A}^{\mathsf{Enc}_\mathsf{k}} \to 1] - \Pr_{\mathcal{O} \xleftarrow{\$} C^M} [\mathcal{A}^{\mathcal{O}} \to 1] \right| \leq \mathsf{negl}(\lambda),$$

where C^M is the space of all functions mapping bitstrings in M to bitstrings in C (that is, $\mathcal{O}$ is a random oracle).

In the QS2 setting, as usual, the access to the oracle becomes quantum.

Definition 16 (Pseudorandomness of Encryption in QS2) *A SKES $\mathcal{E}$ has* pseudorandom quantum ciphertexts *(or, it is qPRF secure) iff, for all QPT adversaries $\mathcal{A}$, it holds:*

$$\left| \Pr_{\mathsf{k} \leftarrow \mathsf{Kgen}(1^\lambda)} [\mathcal{A}^{|\mathsf{Enc_k}\rangle} \to 1] - \Pr_{\mathcal{O} \xleftarrow{\$} C^M} [\mathcal{A}^{|\mathcal{O}\rangle} \to 1] \right| \leq \mathsf{negl}(\lambda),$$

where $|\mathsf{Enc_k}\rangle$ is the canonical (type-1) quantum encryption oracle, and C^M is the space of all functions mapping bitstrings in M to bitstrings in C (that is, $|\mathcal{O}\rangle$ is a quantum random oracle).

As usual, qPRF implies pqPRF, which in turn implies PRF. Pseudorandomness is a very strong property, as it implies any other form of secrecy of ciphertext (IND-CCA2 and related quantum security notions).

4.1.4 Authenticated Encryption

Authenticated encryption (AE) is normally considered one of the strongest security notions for a symmetric-key encryption scheme. It is usually defined as IND-CPA plus INT-CTXT plus PRF. We define naturally pqAE in QS1 (defined as pqIND-CPA plus pqINT-CTXT plus pqPRF). In QS2 there are many different combinations on how to define "quantum AE", we will define explicitly what we mean whenever necessary.

4.2 Modes of Operation

Block ciphers, in their natural definition, are just invertible, deterministic, length-preserving, fixed message-bitsize keyed maps. Security requires them to be pseudorandom (in the correct quantum security domain). In order to obtain a SKES, block ciphers must be combined with a *mode of operation (MoO)*. A MoO is a set of rules specifying how to apply the block cipher to messages of arbitrary size, often exceeding the base input size of the block cipher, and also how to introduce randomization (necessary for achieving higher security notions for the SKES), usually by specifying the management of a *nonce* or *inizialization vector (IV)*.
The definition of a SKES is then given by specifying: the underlying block cipher, the MoO, and some form of security parameter. For example, using AES in CBC mode with a 256-bit key would result in the AES-CBC-256 encryption scheme.
As mentioned, the security of these MoO (and hence the resulting SKES) relies on the pseudorandomness of the underlying block cipher, which is usually a stronger property than simple indistinguishability of ciphertexts. Therefore, it is important to analyze how the quantum security of the underlying block cipher affects the quantum security of the final SKES, depending on the chosen mode of operation. Some of these MoO provide better security guarantees, while others have better performance. We will start by analyzing the 5 most used MoO for block ciphers. In all these cases, security of the underlying block cipher in QS0 (e.g., PRF) implies the corresponding security property for the resulting SKES. This means that SKES built using block ciphers classically inherit "natively" pseudorandomness, and hence all of the other secrecy guarantees. The same holds for QS1, but not necessary for QS2 as we will see.
Finally, there are other MoO that are especially tailored for producing authenticated encrypting (AE) block ciphers. We will analyze some of the most famous ones. In this case care must be taken, as pseudorandomness of the underlying block cipher does not necessarily imply unforgeability of the resulting SKES.

4.2.1 ECB Mode

This is the simplest form of MoO. The plaintext is split in blocks of the same size, and each of them is encrypted separately using the SKES with the same secret key. Identical blocks of plaintexts are mapped to identical blocks of ciphertexts, this MoO is therefore not considered secure even in QS0 and should be avoided when possible. It has the advantage of being fast, simple to implement, parallelizable, and allows selective decryptions of single ciphertext blocks.

4.2.2 CBC Mode

One of the most widely used MoO. Every block of plaintext is first XORed with the previous block of ciphertext, and then encrypted using the block cipher. The first block of ciphertext is XORed with a fresh randomly generated inizialization vector (IV) instead, which is also attached to the ciphertext, in order to allow decryption.
In this MoO identical blocks of plaintexts are usually mapped to different blocks of ciphertexts, and bit-errors on a single ciphertext block only affect the decryption of that block and the next one, but do not propagate to the rest of the ciphertext. This allows the decryption process to be parallelizable (although the encryption must still be sequential), but makes it vulnerable to oracle padding attacks.
From the point of view of QS2 security, CBC mode requires the underlying block cipher to be qPRF (QS2) secure in order to yield an IND-qCPA (QS2) secure SKES [13].

4.2.3 CFB Mode

Another widely used MoO, similar to CBC, but this time every block of plaintext is directly XORed with the encryption (through the SKES) of the previous block of ciphertext (where a fresh IV is used as a “zero-th” ciphertext block). So, in a sense, CFB mode turns the block cipher into a stream cipher, where the seeds of the stream cipher are taken from previous ciphertext blocks.
As in CBC, bit-errors on a single ciphertext block only affect the decryption of that block and the next one, but do not propagate to the rest of the ciphertext. This allows the decryption process to be parallelizable (although the encryption must still be sequential). However, unlike CBC, CFB decryption can be performed by using the underlying block cipher only in encryption mode. This has potential implementation advantages, because it does not require to code the decryption procedure of the SKES. Moreover, if a single block of ciphertext is lost during transmission, only two blocks of plaintexts are affected during decryption.
From the point of view of QS2 quantum security, CFB mode requires the underlying block cipher to be qPRF (QS2) secure in order to yield an IND-qCPA (QS2) secure SKES [13].

4.2.4 OFB Mode

In this mode, the freshly generated IV is iteratively encrypted over and over again (with the same key) in order to obtain a stream of data to XOR the plaintext into the ciphertext blocks, again like in a stream cipher. Decryption is not parallelizable, because decrypting a single block requires processing the IV up to the correct iteration for that block. However, like in CFB, the block cipher is used only in the encryption direction.
From the point of view of QS2 quantum security, OFB mode has the advantage that the underlying block cipher needs only to be pqPRF (QS1) secure in order to yield an IND-qCPA (QS2) secure SKES [13].

4.2.5 CTR Mode

In this mode, a pair IV/counter is encrypted with the underlying SKES, and the output used to XOR a single block of plaintext into ciphertext. Then the counter is increased by 1 when passing to the next block, and so on. Like other MoO mentioned, CTR basically turns the underlying block cipher into a stream cipher. The advantage in respect to other modes of operations is that both encryption and decryption are parallelizable, and it is possible to decrypt blocks at random positions without decrypting other blocks: it is sufficient to compute the right counter for the desired block position and recover the stream key from that counter. This is very useful for applications such as filesystem encryption, where random access to encrypted data is crucial. It is also possible to decrypt data by only relying on the encryption direction of the block cipher.
From the point of view of QS2 quantum security, CTR mode has the advantage that the underlying block cipher needs only to be pqPRF (QS1) secure in order to yield an IND-qCPA (QS2) secure SKES [13].

4.2.6 XTS Mode

This is a NIST-recommended MoO which, however, lacks a formal security analysis even in QS0. Moreover, it suffers from other known classical vulnerabilities, and an explicit attack in QS2 is known if the underlying block cipher is only pqPRF (QS1) secure [13].

4.2.7 CCM Mode

This is a MoO for authenticated encryption (AE), and no quantum security analysis has been performed on it so far. According to [82], this MoO is a very elementary one, suffering from poor performance and having just a few desirable properties.

4.2.8 GCM Mode

This is a MoO for authenticated encryption (AE), and no quantum security analysis has been performed on it so far. According to [82], this MoO is a widely used, standard one. It has decent performance and it is royalty-free, but it is sometimes tricky to implement correctly.

4.2.9 OCB Mode

This is a MoO for authenticated encryption (AE), and it is shown to be insecure in QS2 [13]. According to [82], this MoO has the best performance and security properties in QS0. However it is patent-encumbered, and its adoption for a project such as FutureTPM has to be evaluated with care.

4.2.10 AEX Mode

This is a MoO for authenticated encryption (AE), and no quantum security analysis has been performed on it so far. According to [82], this MoO does not exhibit good performance properties, but it is royalty-free, lightweight on resources, and easy to implement.

4.3 Proposed Candidates

In this section we consider some well-known block ciphers for possible adoption into FutureTPM.

4.3.1 AES Family

The Advanced Encryption Standard (AES), whose original name is Rijndael, was developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, who submitted a proposal to the NIST during the AES selection process that was announced in January 1997. After a five-year standardization process, in which 15 competing designs were presented and evaluated, the NIST announced in November 2001 that Rijndael was approved as AES and published its specification as Federal Information Processing Standard (FIPS) 197.
The AES is based on a design principle known as Substitution-Permutation Network (SPN). It has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. The encryption and decryption process consists of 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. In NIST Special Publication 800-38A to 800-38G, a set of standardized modes of operation for the AES are recommended, including ECB, CBC, CFB, OFB, CTR, CMAC, CCM, GCM, XTS, KW, FF1 and FF3.

QS0 Security

Classical Attacks: In [37], a Related-Key Attack was discovered that exploits the simplicity of AES's key schedule and has a complexity of $2^{99.5}$. In [36], a Key Recovery Attack is described that uses only two related keys and 2^{39} time to recover the complete 256-bit key of a 9-round version, or 2^{45} time for a 10-round version with a stronger type of related subkey attack, or 2^{70} time for an 11-round version. 256-bit AES uses 14 rounds. These attacks are not effective against full AES.
In [79], the first known-key distinguishing attack against a reduced 8-round version of AES-128 was proposed. This known-key distinguishing attack is an improvement of the rebound, or the start-from-the-middle attack, against AES-like permutations. It works on the 8-round version of AES-128, with a time complexity of 2^{48}, and a memory complexity of 2^{32}. As mentioned above, the 128-bit AES uses 10 rounds. This attack is not effective against full AES-128.
In [41], the first key-recovery attack on the full AES was proposed; it is a biclique attack and is faster than brute force by a factor of about four. It requires $2^{126.2}$ operations to recover an AES-128 key. For AES-192 and AES-256, $2^{190.2}$ and $2^{254.6}$ operations are needed, respectively. This result has been further improved to $2^{126.0}$ for AES-128, $2^{189.9}$ for AES-192 and $2^{254.3}$ for AES-256 [153], which are the currently best results for key recovery attacks against the AES.
Side-Channel Attacks: In [84], a cache attack on AES was proposed. It has a "near real time" recovery of secret keys from AES-128 without the need for either cipher text or plaintext. In [18], a very efficient side-channel attack on AES was described that can recover the complete 128-bit AES key and requires just 6-7 blocks of plaintext/ciphertext.

QS1 Security

The Grover algorithm [83] finds with high probability the unique input to a black box function that produces a particular output value, using just $O(\sqrt{n})$ evaluations of the function, where n is the size of the function's domain. Grover's algorithm can be used to achieve a quadratic speedup in the brute force key search, given enough available plaintext-ciphertext pairs. Cryptographers then commonly consider that doubling the length of the keys (or hash lengths) would be enough

to continue having secure symmetric algorithms. In [62], the PQCRYPTO project recommends thoroughly analyzed ciphers with 256-bit keys to achieve 128-bit post-quantum security. In [29], Bernstein et al summarize the effects of Grover's algorithms on typical cryptosystems including AES.
In [24], combining Grover's algorithm and a previously known classical parallel search method, Banegas and Bernstein introduce a quantum algorithm using p small parallel quantum processors connected by a two-dimensional mesh that finds a t-target preimage in roughly $\sqrt{n/pt^{1/2}}$ steps. Here, t different keys are used to encrypt the same plaintext to t different ciphertexts, and the algorithm finds one of the t keys at random. Since NIST's post-quantum security claims for AES assume that Grover's algorithm is the optimal attack strategy, the new bounds mean that in the class QS1 these claims need to be revised.

QS2 Security

Grover's algorithm can be applied to perform a preimage search on a target ciphertext (therefore breaking security properties such as IND-CPA), provided the adversary has quantum oracle access to the encryption procedure with the right key, which is exactly the QS2 scenario. This means that in QS2 an adversary could successfully attack AES (and other symmetric-key ciphers) *without* actually having to perform an exhaustive key search. The complexity of this attack depends *only* on the input blocksize, *regardless* of the keysize used. This is a serious problem for AES and similar ciphers, because with a blocksize of only 128 bit, Grover's algorithm can perform an exhaustive search with a complexity in the order of only 2^{64} operations, and this attack cannot be mitigated by increasing the keysize.
In [81], Grassl et al. also use Grover's algorithm and mention that a quantum version of related-key attacks would be a threat if the attackers have the ability to generate quantum superpositions of related keys. This means that the AES may not be secure in class QS2. Grassl et al. provide reversible circuits that implement the full AES for each standardized key size (i.e., $k = 128, 192, 256$) and establish resource estimates for the number of qubits and the number of Toffoli gates, controlled NOT gates, and NOT gates (see Table 4.1). They find that the number of logical qubits required to implement a Grover attack on the AES is relatively low, but it seems challenging to implement this algorithm on an actual physical quantum computer due to the circuit depth.
Summarizing, in class QS2 the security of the AES and similar block ciphers is seriously affected by Grover's algorithm in theory, although in reality the QS2 attacks might be challenging to mount.

Table 4.1: Quantum resource estimates for Grover's algorithm to attack AES-k, where $k \in \{128, 192, 256\}$

	gates		depth		
k	T	Clifford	T	overall	qubits
128	$1.19 \cdot 2^{86}$	$1.55 \cdot 2^{86}$	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	2953
192	$1.81 \cdot 2^{118}$	$1.17 \cdot 2^{119}$	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	4449
256	$1.41 \cdot 2^{151}$	$1.83 \cdot 2^{151}$	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	6681

4.3.2 Camellia Family

Camellia was developed by Mitsubishi Electric and NTT (Nippon Telegraph and Telephone). It was first published in 2000. It is a symmetric key block cipher with a block size of 128 bits.

Padding must be added such that the data to be encrypted has a length that is a multiple of 16 octets. Its key sizes are 128, 192 and 256 bits.
Camellia is a Feistel cipher with either 18 rounds (when using 128-bit keys) or 24 rounds (when using 192- or 256-bit keys). Every six rounds, a logical transformation layer is applied: the so-called “FL-function” or its inverse. The description of Camellia can be found in RFC 3713. According to RFC 3713, Camellia is characterized by its suitability for both software and hardware implementations as well as its high level of security. From a practical viewpoint, it is designed to enable flexibility in software and hardware implementations on 32-bit processors widely used over the Internet and many applications, and 8-bit processors used in smart cards, cryptographic hardware and embedded systems.
Camellia uses four 8x8-bit S-boxes with input and output affine transformations and logical operations. The cipher also uses input and output key whitening. The diffusion layer uses a linear transformation based on a matrix with a branch number of 5.
This cipher has been approved for use by the ISO/IEC (ISO/IEC 18033-3, “Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers”). It has also been certified by the European Union’s NESSIE project and the Japanese CRYPTREC project. Although Camellia is patented, it is available under a royalty-free license. This has allowed the Camellia cipher to become part of various popular security libraries, such as Crypto++, GnuTLS, mbed TLS and OpenSSL. IETF has published specifications for the use of Camellia with IPsec (RFC 4312), TLS (RFC 4132), Secure/Multipurpose Internet Mail Extensions (S/MIME) (RFC 3657), and XML Securiy (RFC 4051).
Camellia can be used in Counter mode (CTR) and Counter with Cipher Block Chaining MAC mode (CCM) (see RFC 5528).

Security Analysis. Camellia is a block cipher that can be completely defined by minimal systems of multivariate polynomials. In Camellia-128, the key schedule can be described by 1120 equations in 768 variables using 2816 linear and quadratic terms, while the block cipher can be described by 5104 equations in 2816 variables using 12288 linear and quadratic terms [35]. In total, 6224 equations in 3584 variables using 15104 linear and quadratic terms are required [35]. The number of free terms is 8880. Theoretically, such properties might make it possible to break Camellia by using an algebraic attack, such as extended sparse linearisation, if in the future such an attack becomes feasible.
Several cryptanalysis results on Camellia have been published, but none of them implies a practical attack on Camellia. Li et al [105] use truncated differential cryptanalysis to attack 11/12-round Camellia-128/192 with $2^{121.3}$ and $2^{185.3}$ encryptions. Blondeau [39] presents the first attack on 13 rounds of Camellia-192 and an attack on 14 rounds of Camellia-256 requiring less complexity than the previous impossible differential attacks. Liu et al [106] present an 8-round zero-correlation linear distinguisher of Camellia, which they use to launch key recovery attacks on 13-round Camellia-192 and 14-round Camellia-256. Jia and Wang [99] build on the results by Liu et al to give an impossible differential attack on 14-round Camellia-192 with $2^{126.5}$ known plaintexts and $2^{189.32}$ encryptions. Koie et al [104] use truncated differential cryptanalysis to give low data complexity attacks on 4 to 7 rounds of Camellia.
No specific quantum attacks are known for Camellia. However, all the known attacks existing for AES and similar generic block ciphers apply, both in QS1 and QS2.

4.3.3 Serpent Family

Serpent is a block cipher that was designed by Ross Anderson, Eli Biham, and Lars Knudsen [14]. It was first published in 1998. Serpent has a block size of 128 bits. Its key sizes are 128, 192 or 256 bits.
Serpent is a 32-round substitution-permutation network operating on a block of four 32-bit words, plus an initial and a final permutation to simplify an optimized implementation. Each round applies one of eight 4-bit to 4-bit S-boxes 32 times in parallel. More in detail, the round function in Serpent consists of key-mixing XOR, thirty-two parallel applications of the same 4x4 S-box, and a linear transformation, except in the last round, wherein another key-mixing XOR replaces the linear transformation. Therefore, each S-box is used in 4 rounds. Serpent was designed so that all operations can be executed in parallel, using 32 bit slices. This maximizes parallelism.
The S-boxes of Serpent are 4-bit permutations with the following properties:

- Each differential characteristic has a probability of at most 1/4, and a one-bit input difference will never lead to a one-bit output difference.
- Each linear characteristic has a probability in the range $1/2 \pm 1/4$, and a linear relation between one single bit in the input and one single bit in the output has a probability in the range $1/2 \pm 1/8$.
- The nonlinear order of the output bits as a function of the input bits is the maximum, namely 3. However, Singh et al [147] show that there are some output bits whose nonlinear order as a function of the input bits is only 2.

The Serpent cipher algorithm is in the public domain and has not been patented. The reference code is public domain software and the optimized code is under GPL. Serpent was ranked second in the Advanced Encryption Standard contest.
Serpent can be used in various modes of operation. For example, RFC 4344 on the Secure Shell (SSH) transport layer encryption modes lists as optional the use of Serpent-(128, 192, 256) in CTR mode, while RFC 4253 on the Secure Shell (SSH) transport layer protocol lists as optional the use of Serpent-(128, 192, 256) in CBC mode.

Security Analysis. Serpent is a block cipher that can be completely defined by minimal systems of multivariate polynomials. In Serpent-128, the key schedule can be described by 8848 equations in 8448 variables using 15048 linear and quadratic terms, while the block cipher can be described by 8832 equations in 8192 variables using 14592 linear and quadratic terms [35]. In total, 17680 equations in 16640 variables using 29640 linear and quadratic terms are required [35]. The number of free terms is 11960. Theoretically, such properties might make it possible to break Serpent by using an algebraic attack, such as extended sparse linearisation, if in the future such an attack becomes feasible.
Several cryptanalysis results on Serpent have been published, but none of them implies a practical attack on Serpent. In 2012, Wang et al [166] use structures in differential attacks, i.e. the use of multiple input and one output difference, to improve the previous structure attacks on 7-round and 8-round Serpent. In 2014, Tezcan et al [161] analyze the security of Serpent against impossible and improbable differential cryptanalysis and provide a 7-round improbable differential attack by using undisturbed bits of its S-boxes. They conclude that Serpent is secure against these kind of attacks. In 2015, Tezcan uses differential factors to reduce the data and time complexity of the previous differential-linear attacks on 11-round Serpent-192 and 12-round Serpent-256. The resistance of Serpent S-boxes to differential power analysis has also been analyzed [125].

No specific quantum attacks are known for Serpent. However, all the known attacks existing for AES and similar generic block ciphers apply, both in QS1 and QS2.

4.3.4 Twofish Family

Twofish was designed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson [143]. It was first published in 1998. Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes of up to 256 bits.
Twofish uses a Feistel structure with 16 rounds. Its distinctive features are the use of pre-computed key-dependent S-boxes, and a relatively complex key schedule. One half of an n-bit key is used as the actual encryption key and the other half of the n-bit key is used to modify the encryption algorithm (key-dependent S-boxes). Twofish borrows some elements from other designs, e.g. the pseudo-Hadamard transform. It is derived from the previous block ciphers Blowfish, SAFER and Square. Twofish also employs a Maximum Distance Separable (MDS) matrix.
More in detail, Twofish uses a 16-round Feistel-like structure with additional whitening of the input and output. The only non-Feistel elements are the 1-bit rotates. The rotations can be moved into the F function, which is a key-dependent permutation on 64-bit values, to create a pure Feistel structure, but this requires an additional rotation of the words just before the output whitening step. The plaintext is split into four 32-bit words. In the input whitening step, these are XORed with four key words. This is followed by sixteen rounds. In each round, the two words on the left are used as input to the g functions. (One of them is rotated by 8 bits first.) The g function consists of four byte-wide key-dependent S-boxes, followed by a linear mixing step based on an MDS matrix. The results of the two g functions are combined using a Pseudo-Hadamard Transform (PHT), and two key words are added. These two results are then XORed into the words on the right (one of which is rotated left by 1 bit first, the other is rotated right afterwards). The left and right halves are then swapped for the next round. After all the rounds, the swap of the last round is reversed, and the four words are XORed with four more key words to produce the ciphertext.
The key dependent S-boxes in the function g are defined by $g(X) = h(X, S)$. The function h is a function that takes two inputs, a 32-bit word X and a list $L = (L_0, \ldots, L_{k-1})$ of 32-bit words of length k, and produces one word of output. This function works in k stages. In each stage, the four bytes are each passed through a fixed S-box, and XORed with a byte derived from the list. Finally, the bytes are once again passed through a fixed S-box, and the four bytes are multiplied by the MDS matrix just as in g. In other words, for $i = 0, \ldots, 3$, the key-dependent S-box s_i is formed by the mapping from x_i to y_i in the h function, where the list L is equal to the vector S derived from the key.
The Twofish cipher has not been patented and the reference implementation has been placed in the public domain. It is one of the ciphers included in the OpenPGP standard (RFC 4880). It was one of the five finalists of the Advanced Encryption Standard contest.
Twofish can be used in various modes of operation, e.g. ECB, CBC, CFB, OFB and CTR. For example, RFC 4344 on the Secure Shell (SSH) transport layer encryption modes lists as optional the use of Twofish-(128, 192, 256) in CTR mode, while RFC 4253 on the Secure Shell (SSH) transport layer protocol lists as optional the use of Twofish-(128, 192, 256) in CBC mode.

Security Analysis. Several cryptanalysis results on Twofish have been published, but none of them implies a practical attack on Twofish. In 1999, Ferguson [73] shows an impossible-differential attack against 6-round Twofish-256. In 2000, Murphy and Robshaw [120] show five- and six-round characteristics that improve the attacks in [73]. In 2001, Lucks [107] introduces saturation attacks and applies them to reduced-round variants of the Twofish block cipher with

up to seven rounds with full whitening or eight rounds without whitening at the end. Those attacks take up to 2^{127} chosen plaintexts and are 2 to 4 times faster than exhaustive search. In 2005, Macchetti [110] analyzes and discusses the cryptographic robustness of key-dependent substitution boxes (KDSBs) by deriving the expressions of their linear and differential characteristics. Macchetti shows that Twofish KDSBs, although very efficient, can be easily distinguished from truly randomly built KDSBs. The resistance of Twofish to differential fault analysis [8] and to algebraic side-channel attacks [109] has also been analyzed.
No specific quantum attacks are known for Twofish. However, all the known attacks existing for AES and similar generic block ciphers apply, both in QS1 and QS2.

4.4 Candidates Comparison

In terms of quantum security, all of the schemes presented in the previous section are suitable for use in FutureTPM if used with a minimum keysize of 256 bits. As discussed in the previous sections, doubling the keysize is sufficient for the QS1 scenario, but for the more challenging QS2 scenario it is also necessary to increase the blocksize. The following tables summarize the equivalent bit-security level for 128 and 256 bit blocksize.

Table 4.2: Quantum security levels for block ciphers: 128 bit blocksize.

	equivalent bit-security		
key bitsize	QS0	QS1	QS2
128	128	64	64
192	192	96	64
256	256	128	64

Table 4.3: Quantum security levels for block ciphers: 256 bit blocksize.

	equivalent bit-security		
key bitsize	QS0	QS1	QS2
128	128	64	64
192	192	96	96
256	256	128	128

Regarding modes of operation, we make a distinction between modes offering encryption only, or authenticated encryption (AE). In the former case, we notice that ECB and XTS do not offer strong security guarantees. CBC and CFB are suitable, but we recommend using OFB and CTR for improved security, especially in the more challenging QS2 scenario. In the case of AE modes, we do not recommend the adoption of OCB, for security and IP reasons. This might pose as a challenge for the design of FutureTPM, as OCB is considered very appealing in terms of performance.

4.5 Open Issues

The study of quantum security of block ciphers is a much more complex topic than commonly assumed by a large part of the cryptographic community. Although conceivably less affected by quantum computing than other public-key primitives, many questions about the security of block

ciphers in a quantum world remain unanswered. The recommendation and guidelines proposed in this evaluation are based on the state-of-the-art in the academic literature. Future deliverables of the current working package of FutureTPM might change the evaluation as scientific discussion on these topics advances.
We stress the difference between security in QS1 and QS2. The latter is a much more challenging scenario for block ciphers, and attacks cannot be mitigated by just increasing the keysize. The limited blocksize of the currently considered ciphers (128 bit) is particularly troublesome. Future deliverables might focus on alternative block ciphers offering larger blocksize.
We notice the need for more studies in quantum cryptanalysis. The existing literature mostly focuses on general improvements of Grover's search algorithm, but very few works address the algebraic structure of the underlying ciphers, and those wo do mostly focus on AES. Advanced attacks in QS2, such as bad randomness and weak key attacks, are still poorly understood, and mitigation techniques against other attacks (e.g., quantum key-related attacks) are unavailable. At the moment of writing, it is not even clear how to efficiently build a fundamental object such as *quantum-secure pseudorandom permutations (qPRP)*[169, 77], although it is conceivable to believe that Feistel or Even-Mansour ciphers with a large number of rounds should suffice. More study is also needed regarding the quantum security of block ciphers' modes of operations, especially in terms of AE modes.
In the QS2 setting, achieving qIND-qCPA security or above is considered tricky. None of the MoO proposed so far achieves such a level of security, and the only known method achieving that is unpractical for use in FutureTPM (see [77]), because it would incur in a large performance penalty. Better modes of operation for qIND-qCPA (and above) are required.
Finally, we notice that correctly defining (and achieving) *quantum indifferentiability* for symmetric-key encryption is an open problem, and currently unclear whether possible at all. In the setting of FutureTPM, indifferentiability against quantum adversaries would be a very desirable property, as it would imply good composition properties of the underlying SKES with the more complex protocols and primitives offered by the TPM architechture.

Chapter 5

Digital Signatures

Digital signature schemes (DSS) allow (using the public key of a user) to verify that a *digital signature* on a particular message was produced by the owner of the corresponding secret key. Formally, a DSS is defined as follows.

Definition 17 (Digital Signature Scheme) *A (stateful) digital signature scheme is a tuple of PPT algorithms:*

- $\mathsf{DS.Kgen}$ *takes as input a (unary representation of) security parameter* 1^λ *and outputs a signing-verifying keypair* $(\mathsf{sk}, \mathsf{vk})$ *and (optionally) an (initial) signature state* $\mathsf{state}_{\mathsf{DS}}$*. Without loss of generality, we assume that the secret signing key* sk *includes the public verification key* vk*, and* vk *includes the security parameter* λ*.*
- $\mathsf{DS.Sign}$ *takes as input a message* msg*, a signing key* sk*, and a state* $\mathsf{state}_{\mathsf{DS}}$*, and outputs a signature* σ *and an updated state* $\mathsf{state}_{\mathsf{DS}}'$*.*
- $\mathsf{DS.Verify}$ *takes as input a message* msg*, a verification key* vk*, and a signature* σ*, and outputs an acceptance/rejectal bit* OK *or* rej*.*

In many DSS the state can be omitted. In this case we call the DSS *stateless*. We will only consider stateless schemes here, although stateful schemes might be considered in future versions of this document.

5.1 Security Models

Intuitively, for a DSS to be secure, one does not want an adversary to be able to produce a valid signature for a certain public key without having the corresponding secret key. However, there are also certain subtleties to consider when the DSS is dependent on, or interacts with, other primitives, especially hash functions.

5.1.1 Unforgeability

Unforgeability of a DSS requires that no adversary can efficiently generate a valid signature for a given public key without having the corresponding secret key. This notion can come in different flavors (existential, strong, weak, etc.) according to how much constraints are put on an adversary to be considered successful. Moreover, each of these flavors can be analyzed in each one of the quantum security domain we consider.

Existential Unforgeability

In the classical (QS0) domain, a common security notion is the adaptive version (chosen-message attack) of existential unforgeability, or *EUF-CMA* in short. The non-adaptive version (EUF) requires that no PPT adversary is able, given as input a freshly generated public key, to output a valid pair (message, signature) for that key without having the corresponding secret key.
Such notion is usually augmented by giving the adversary the possibility of seeing, adaptively, valid signatures on messages of his choice. Formally, this is done by giving the adversary access to a *signing oracle* $\mathcal{O}_{\mathsf{DS.Sign}_{\mathsf{sk}}}$ for the correct signing key sk. The requirement for a successful adversary in this case is that he must produce a valid (message, signature) pair for a fresh message, i.e., one which was never sent to the oracle before. The resulting security notion is called *existential unforgeability under chosen message attack (EUF-CMA)*, and it is the minimum meaningful security guarantee for a DSS.
In the QS1 (post-quantum) setting, EUF-CMA is modified by granting the adversary local quantum computing power. That is, PPT adversaries are replaced by QPT adversaries. The resulting QS1 notion is called *post-quantum existential unforgeability under chosen message attack (pqEUF-CMA).* Notice how, in this case, the access to the signing oracle is still classical. This models the natural real-world consideration, where the adversary has access to signatures previously generated by an honest (and hence classical) party.
However, for the consideration explained in Section 2.1, it is possible to strengthen the above notions to the QS2 setting, by giving the adversary quantum access to the signing oracle as well. That is, the signing oracle $\mathcal{O}_{\mathsf{DS.Sign}_{\mathsf{sk}}}$ is replaced by a quantum signing oracle $|\mathcal{O}_{\mathsf{DS.Sign}_{\mathsf{sk}}}\rangle$, acting as:

$$|\mathsf{msg}, y\rangle \mapsto |\mathsf{msg}, y \oplus \mathsf{DS.Sign}_{\mathsf{sk}}(\mathsf{msg})\rangle$$

In this case, the signing oracle can produce signatures on a quantum superposition of messages. This means that the notion of "fresh message" is ill-defined. To circumvent such problem, the winning probability of the adversary is modified using a "one-more approach" in the following way: it is required that an adversary performing q quantum queries to the signing oracle must output $q + 1$ valid (message, signature) pairs, such that all $q + 1$ messages are different from each other. The resulting QS2 notion is called *existential unforgeability under quantum chosen message attack (EUF-qCMA).*

Strong Unforgeability

EUF-CMA can be strengthened in a further, subtle way. Instead of requiring a successful adversary to output a valid forgery for a *fresh message*, one can require merely that the *signature* must be fresh. That is, the adversary could ask the signing oracle to provide a bunch of valid signatures for a certain message of his choice, and then output a new signature for that same message. Intuitively, this is a potentially easier task for the adversary, because he can exploit the *malleability* of the signatures: indeed there do exist schemes such that by combining, e.g., two valid signatures for a certain message, one can find another valid signature (this has often to do with the homomorphic properties of the signature, and happens for example in RSA and related schemes). This means that the resulting security notion is stronger. In the QS0 setting, this is called *strong unforgeability under chosen message attack (SUF-CMA).*
The corresponding QS1 and QS2 variants are called *post-quantum strong unforgeability under chosen message attack (pqSUF-CMA)*, and *strong unforgeability under quantum chosen message attack (SUF-qCMA)*, respectively.

	EUF-CMA	pqEUF-CMA	EUF-qCMA
ROM	QS0	QS0.5	invalid[1]
QROM	invalid[2]	QS1	QS2

Figure 5.1: security domain classification for Existential Unforgeability for DSS.

One can show that the strong unforgeability notions are always strictly stronger than the analogous existential unforgeability notions. Also, any QS(x) notion is always weaker than the corresponding QS$(x+1)$ notion. However, SUF in QS(x) is usually uncomparable to EUF in QS$(x+1)$. This means that the strongest security notion one can achieve is SUF-qCMA.

5.1.2 ROM VS QROM

Security proofs for DSS are often given in the Random Oracle Model. It is important to relate these models to the quantum security setting. We recall that:

- in the (classical) ROM, hash functions and other similar primitives are modeled as a random oracle $\mathcal{O}$, which is a black-box random function accessed classically (for any kind of adversaries, even the quantum ones); while
- in the QROM, the black-box random function is accessed in a quantum way (possibly on a superposition of inputs) as a unitary gate $|\mathcal{O}\rangle$.

As discussed in Section 2.1, the QROM is the model most suited to the case of quantum adversaries with access to a public-code primitive. In the case of signatures, it is common to combine standard security definitions with the ROM. In the quantum security settings, one has to distinguish whether such definitions are given in the ROM or QROM, and which security domain the resulting notions belong to. Starting from the EUF-CMA notion, the situation is summarized in Figure 5.1. The related possibilities are:

- EUF-CMA-ROM: this would be the classical EUF-CMA definition, where the adversary also has (classical) access to the random oracle $\mathcal{O}$. This is the traditional QS0 definition in the ROM.
- pqEUF-CMA-ROM: this notion takes into account existential unforgeability against quantum adversaries with classical access to both signature and random oracle. Despite this notion being (implicitly) widely used in post-quantum signatures, we only classify it as a QS0.5 security notion, because the most natural model for quantum adversaries is the QROM.
- pqEUF-CMA-QROM: existential unforgeability against quantum adversaries with quantum access to the random oracle, but only classical access to the signing oracle. This is a perfectly natural assumption, because the signing oracle depends on the secret key of a (classical) honest party, while the random oracle usually does not. Therefore this is the canonical QS1 security notion for DSS.
- EUF-qCMA-QROM: this is existential unforgeability against quantum adversaries with quantum access both to the random and signing oracles. This very strong security notion is hence classified as QS2.

	SUF-CMA	pqSUF-CMA	SUF-qCMA
ROM	QS0	QS0.5	invalid[3]
QROM	invalid[4]	QS1	QS2

Figure 5.2: security domain classification for Strong Unforgeability for DSS.

$$\begin{tikzcd}
\text{SUF-qCMA-QROM} \arrow[r] \arrow[d] & \text{pqSUF-CMA-QROM} \arrow[r] \arrow[d] & \text{pqSUF-CMA-ROM} \arrow[r] \arrow[d] & \text{SUF-CMA-ROM} \arrow[d] \\
\text{EUF-qCMA-QROM} \arrow[r] & \text{pqEUF-CMA-QROM} \arrow[r] & \text{pqEUF-CMA-ROM} \arrow[r] & \text{EUF-CMA-ROM}
\end{tikzcd}$$

Figure 5.3: quantum security notion hierarchy for DSS. An arrow means “stricty implies”.

The same classification and discussion applies when starting from SUF-CMA instead, with the resulting security notions in QS0, QS0.5, QS1, and QS2 respectively. The situation is summarized in Figure 5.2.
Recalling that SUF-CMA is generally stronger than EUF-CMA, and that QS(x) is usually weaker than QS(y) for $x < y$, we can identify a hierarchy of security notions for DSS summarized in Figure 5.3. The weakest form of quantum security for DSS is hence pqEUF-CMA-ROM, while the strongest is SUF-qCMA-QROM.

5.1.3 Transformations

A few useful generic transformations are known which can strengthen the security properties of quantum-secure signature schemes under additional assumptions. We recall here two important ones.
The first one, due to Boneh and Zhandry[44], allows to go from CMA to qCMA security in the QROM by using a family of pairwise independent functions.

Theorem 1 ([44]) *Let $\Sigma := (\mathsf{DS.Kgen}, \mathsf{DS.Sign}, \mathsf{DS.Verify})$ be a DSS, $\mathcal{O}$ a random oracle, and $\mathcal{Q}$ a family of pairwise independent functions. Consider the new DSS $\Sigma' := (\mathsf{DS.Kgen}', \mathsf{DS.Sign}', \mathsf{DS.Verify}')$ defined as follows:*

- $\mathsf{DS.Kgen}' := \mathsf{DS.Kgen}$
- $\mathsf{DS.Sign}'_{\mathsf{sk}}(m) \leftarrow \mathsf{DS.Sign}_{\mathsf{sk}}(\mathcal{O}(m\|r); Q(m), r)$, *where* $r \xleftarrow{\$} \mathcal{R}$ *and* $Q \xleftarrow{\$} \mathcal{Q}$;
- $\mathsf{DS.Verify}'_{\mathsf{vk}}(m, \sigma') := \mathsf{DS.Verify}_{\mathsf{vk}}(\mathcal{O}(m\|r), \sigma)$, *where* $\sigma' := (\sigma, r)$.

If Σ is pqEUF-CMA (resp., pqSUF-CMA), then Σ' is EUF-qCMA-QROM (resp., SUF-qCMA-QROM).

In practical applications, one instantiates $\mathcal{O}$ with a hash function, as usual, and $\mathcal{Q}$ can be a family of linear functions.
The other useful transformation is the Teranishi-Oyama-Ogata (TOO) [160], proven secure in the QROM by Eaton and Song [70]. It allows to strengthen security from EUF to SUF by using chameleon hash functions. Chameleon hash functions are a special type of hash functions which only allow to find collisions if one knows a secret trapdoor.

Definition 18 (Chameleon Hash Function) *A* chameleon hash function *with range $\mathcal{H}$ is a tuple of PPT algorithms:*

- $\mathsf{Chml.Kgen}$ *takes as input a (unary representation of) security parameter 1^λ and outputs a public-private keypair $(\mathsf{pk_{Ch}}, \mathsf{sk_{Ch}})$. Without loss of generality, we assume that the secret key $\mathsf{sk_{Ch}}$ includes the public key $\mathsf{pk_{Ch}}$, and $\mathsf{pk_{Ch}}$ includes the security parameter λ.*
- $\mathsf{Chml.Hash}$ *takes as input a message msg, a public key $\mathsf{pk_{Ch}}$, and a randomness r, and outputs a hash value $h \in \mathcal{H}$.*
- $\mathsf{Chml.Sample}$ *takes as input a (unary representation of) security parameter 1^λ and outputs a randomness r with the property that $\mathsf{Chml.Hash}(\mathsf{msg}, \mathsf{pk_{Ch}}, r)$ is uniform over the output of $\mathsf{Chml.Sample}$, for every message msg and public key $\mathsf{pk_{Ch}}$.*
- $\mathsf{Chml.Invert}$ *takes as input a secret key $\mathsf{sk_{Ch}}$, a message msg, and a hash value $h \in \mathcal{H}$, and outputs a randomness r.*

A chameleon hash function is post-quantum if the following hold:

1. (collision resistance): *for any QPT adversary, averaged over $(\mathsf{pk_{Ch}}, \mathsf{sk_{Ch}}) \leftarrow \mathsf{Chml.Kgen}$, it is unfeasible to find $\mathsf{msg}, \mathsf{msg}', r, r'$ with $(\mathsf{msg}, r) \neq (\mathsf{msg}', r')$ such that $\mathsf{Chml.Hash}_{\mathsf{pk_{Ch}}}(\mathsf{msg}, r) = \mathsf{Chml.Hash}_{\mathsf{pk_{Ch}}}(\mathsf{msg}', r')$*
2. (chameleon property): *for any $(\mathsf{pk_{Ch}}, \mathsf{sk_{Ch}}) \leftarrow \mathsf{Chml.Kgen}(1^\lambda)$, for any message msg and hash value $h \in \mathcal{H}$, it holds that $\mathsf{Chml.Hash}_{\mathsf{pk_{Ch}}}(\mathsf{msg}, r) := h$, where $r \leftarrow \mathsf{Chml.Invert}_{\mathsf{sk_{Ch}}}(\mathsf{msg}, h)$.*
3. (uniformity): *the distributions of randomness produced by $\mathsf{Chml.Sample}$ and $\mathsf{Chml.Invert}$ (on messages chosen by the adversary) are (quantumly) computationally indistinguishable.*

Practically, post-quantum chameleon hash functions can be constructed from many quantum-resistant trapdoor permutations (e.g, based on lattice problems such as SIS).

Theorem 2 ([70]) *Let $\Sigma := (\mathsf{DS.Kgen}, \mathsf{DS.Sign}, \mathsf{DS.Verify})$ be a DSS, $\mathcal{O}$ a random oracle, and $\Phi := (\mathsf{Chml.Kgen}, \mathsf{Chml.Sample}, \mathsf{Chml.Hash}, \mathsf{Chml.Invert})$ a post-quantum chameleon hash function with range $\mathcal{H}$. Consider the new DSS $\Sigma' := (\mathsf{DS.Kgen}', \mathsf{DS.Sign}', \mathsf{DS.Verify}')$ defined as follows:*

- $\mathsf{DS.Kgen}'(1^\lambda) \rightarrow (\mathsf{pk}', \mathsf{sk}')$, *where:* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS.Kgen}(1^\lambda)$,
  $(\mathsf{pk_{Ch}}, \mathsf{sk_{Ch}}) \leftarrow \mathsf{Chml.Kgen}(1^\lambda), \mathsf{pk}' := (\mathsf{vk}, \mathsf{pk_{Ch}}), \mathsf{sk}' := (\mathsf{sk}, \mathsf{sk_{Ch}})$.
- $\mathsf{DS.Sign}'_{\mathsf{sk}'}(m) \rightarrow (\sigma, r)$, *where:* $h \xleftarrow{\$} \mathcal{H}, \sigma \leftarrow \mathsf{DS.Sign}_{\mathsf{sk}}(h)$,
  $M := \mathcal{O}(m \| \sigma), r \leftarrow \mathsf{Chml.Invert}_{\mathsf{sk_{Ch}}}(M, h)$.
- $\mathsf{DS.Verify}'_{\mathsf{pk}'}(m, \sigma') := \mathsf{DS.Verify}_{\mathsf{vk}}(\mathsf{Chml.Hash}_{\mathsf{pk_{Ch}}}(\mathcal{O}(m \| \sigma), r), \sigma)$, *where* $\sigma' := (\sigma, r)$.

If Σ if pqEUF-CMA-QROM, then Σ' is pqSUF-CMA-QROM.

5.1.4 Constructions

There are many ways of constructing digital signature schemes. Some construction types exhibit more desirable quantum security properties, while others offer better performance. In this section we discuss some of the common constructions.

Hash-And-Sign

Hash-and-sign constructions of DSS are relatively simple: first the message is hashed, and then the hash is signed by applying the inverse (which depends on the secret key) of a one-way trapdoor permutation (OWTP). The verifier can apply the trapdoor permutation in the one-way direction using the signer's public key, and verify that what is obtained is actually the hash of the message. This is a canonical construction, used e.g. in RSA signatures.
These constructions require security in the QROM in order to provide at least QS1 security. The security proof often involves careful reprogramming of the random oracle, but usually goes through without big issues (see for example [60, 44]).
The disadvantage is that a one-way trapdoor permutation is required, which is a relatively strong computational assumption. Candidates for quantum-resistant OWTPs (based, e.g., on SIS and other lattice problems) exist, however the resulting keysize and signatures are usually quite large.

Fiat-Shamir

Fiat-Shamir (F-S) signatures are constructed starting from any 3-step, 2-party interactive identification scheme (also called a Σ-protocol) by removing the interactivity using a hash function, and binding the message to the resulting transcript. These signatures have numerous advantages:

- they can be built from any Σ-protocol: this is a weaker assumption than the existence of OWTP, and many post-quantum candidates are known;
- the same construction can be used in general to turn any (interactive) zero-knowledge proof of knowledge protocol into a non-interactive one (NIZKPoK); this often results in desirable properties, such as the possibility of proving efficiently in zero-knowledge a certain signature, useful for many privacy applications;
- the resulting signatures are usually very short.

However, F-S signatures also have disadvantages:

- the security proof requires rewinding: this results in non-tight bounds, therefore making difficult to give reasonable keysize estimate for desired security parameters;
- rewinding in the QROM is an infamously tricky technique, which usually does not work without additional assumptions on the underlying Σ-protocol. The most useful of these additional assumptions, "perfectly unique responses", is generally difficult to achieve in lattice-based and other candidate quantum-secure problems, due to the intrinsic presence of errors in the exchanged values. Other useful properties are often expensive to achieve. With only few exceptions, F-S signatures are thus only proven secure in the classical ROM, achieving only QS0.5.

Unruh's Transform

Unruh's Transform (UT) is a modification of F-S signatures made in order to achieve security on the QROM without additional assumptions. In a nutshell, a UT signature is a vector of many parallel F-S signatures where every signature includes additional "secret information". The knowledge of all this information would allow an adversary to recover the secret key by observing a single valid signature, therefore this information is hidden in a commitment, and only part of it is revealed in the signature, using a hash-dependant cut-and-choose technique.

The main disadvantage of UT signatures is the overhead of data necessary, which blows up signature sizes considerably. However, in many practical scenarios, this blowup is mitigated by the fact that the underlying FS signature is small, and should be anyway repeated many times for achieving a secure level of soundness. In many schemes this results in "practical" overhead of roughly 2 times compared to the corresponding FS signature, but at the advantage of achieving QROM security, and maintaining the "NIZK-friendliness" of F-S.

Merkle Trees

This construction is at the core of many hash-based signature schemes. Usually, hash-based signatures start with a one-time variant (e.g., Lamport signatures), where the secret key is a bunch of random values, and the public key is the corresponding hashes. A signature is generated by revealing some of the hash preimages, in a pattern determined by the message. Clearly, this degrades the security of a given keypair after every signature.
A Merkle tree is a binary tree where every node stores the hash value of the parent nodes. By combining one-time hash-based signatures in a Merkle tree structure, a single public key can be reused to sign many different messages without substantially degrading security.
The disadvantage of these constructions is that usually signature and public-key size are very large compared to other schemes, and moreover they are usually stateful, a rather undesirable property for many practical applications. The presence of the state can be eliminated in some constructions, at the expense of a further increase in key and signature size.
The advantages of this kind of constructions is that they are very fast, require minimum code to run, are usually secure in the QROM, and they are only based on very minimal hardness assumptions.

5.2 Proposed Candidates

5.2.1 Dilithium Family

In this subsection, we given an overview of the CRYSTALS-Dilithium scheme, which was submitted to the NIST Post-Quantum Standardization process [68]. Dilithium is built over lattices, and its security is based on the Module-LWE and Modulee-SIS hardness assumptions, which are more general versions of the Ring-LWE and Ring-SIS respectively.
The design of the scheme is based on the Fiat-Shamir with Aborts approach and the scheme works as follows:
Key Generation: The key generation algorithm generates a $k \times \ell$ matrix A whose entries are elements of the ring $\mathcal{R} = \mathbb{Z}_q/(x^n + 1)$ with $q = 2^{23} - 2^{13} + 1$ and $n = 256$. The secret consists of two random vectors $\mathbf{s}_1 \in \mathcal{R}^\ell$ and $\mathbf{s}_2 \in \mathcal{R}^k$, and with small coefficients of size at most η. Finally, the public key is computed as $\mathbf{t} = A\mathbf{s}_1 + \mathbf{s}_2$.

Sign: The signing algorithm generates a masking vector of polynomials $\mathbf{y}$ with coefficients less than γ_1, where γ_1 is large enough so that the final signature doesn't reveal the secret key (the signing algorithm is zero-knowledge), yet γ_1 is small enough so that the signature is not easily forged. Let $\mathbf{w} = A\mathbf{y}$, then every coefficient w of $\mathbf{w}$ can be written in a canonical way as $w = w_1 \cdot 2\gamma_2 + w_0$, where $|w_0| \leq \gamma_2$. Let $\mathbf{w}_1$ denote the vector comprising all w_1's. The signature is then computed as $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$, where c is a polynomial created as a hash of the message and $\mathbf{w}_1$, i.e. $c = H(M|\mathbf{w}_1)$, that belongs to $\mathcal{R}$ with exactly 60 ± 1 and the rest are 0's. Let β be the parameter corresponding to the maximum possible coefficient of $|c\mathbf{s}_i|$, clearly $\beta \leq 60\eta$. The scheme uses the rejection sampling to avoid the dependency of $\mathbf{z}$ on the secret key, if any of the coefficents of $\mathbf{z}$ is larger than $\gamma_1 - \beta$, then $\mathbf{z}$ is rejected and the signing proceedure is restarted. For correctness of the scheme, the signing is also restarted if any coefficient of the low-order bits of $A\mathbf{z} - c\mathbf{t}$ is greater than $\gamma_2 - \beta$. The signature is $\sigma = (\mathbf{z}, c)$.

Lemma 1 *[68] : If $\|\mathbf{s}\|_\infty \leq \beta$ and $\|LowBits_q(\mathbf{r}, \alpha)\|_\infty < \alpha/2 - \beta$, then:*

$$HighBits_q(\mathbf{r}, \alpha) = HighBits_q(\mathbf{r} + \mathbf{s}, \alpha)$$

Verification: The verifier first computes $\mathbf{w}_1'$ to be the high order bits of $A\mathbf{z} - c\mathbf{t}$, accepts if all the coefficients of $\mathbf{z}$ are less than $\gamma_1 - \beta$ and c is that hash of the message and $\mathbf{w}_1'$. In particular we have

$$HighBits(A\mathbf{z} - c\mathbf{t}, 2\gamma_2) = HighBits(A\mathbf{y}, 2\gamma_2)$$

This is true since $A\mathbf{z} - c\mathbf{t} = A\mathbf{y} - c\mathbf{s}_2$, and we have $\|LowBits(A\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta$, and $\|c\mathbf{s}_2\|_\infty < \beta$, therefore it follows from Lemma 1 the equality:

$$HighBits(A\mathbf{y}, 2\gamma_2) = HighBits(A\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)$$

5.2.2 Tesla Family

In this subsection, we give an overview of the qTESLA scheme, which was submitted to the NIST Post-Quantum Standardization process [34]. qTESLA is a signature scheme based on lattices whose security is based on the standard LWE assumption.

Definition 19 *Let w be an integer polynomial, we define the following:*

- *$[.]_L$ is the value represented by the d least significant bit of w.*
- *$[.]_m$ is the value represented by the d most significant bit of w.*
- *w is well-rounded if $\|w\|_\infty \leq \lfloor q/2 \rfloor - L_E$ and $\|[w]_L\|_\infty \leq 2^d - L_E$*
- *We define the hash oracle H: $\{0,1\}^* \to \mathbb{H}$, where $\mathbb{H}$ denotes the set of polynomials $c \in \mathcal{R}$ with coefficients in $\{-1, 0, 1\}$ with excatly h nonzero entries.*

Key Generation: Let $\mathcal{R} = \mathbb{Z}_q/(x^n + 1)$, and $(n, q, \gamma, L_E, L_S, B, d)$ and h be the system parameters.

- The algorithm samples a uniformly randon invertible ring element a from the ring $\mathcal{R}$.
- Choose two short polynomials e and s from some distribution χ_γ.
- If the h largest entries of e sum to L_E, then sample new polynomial e.
- If the h largest entries of s sum to L_S, then sample new polynomial s.
- Let $t = as + e \;\in \mathcal{R}$.
- The secret key is the pair(s, e), and the public key is (a, t).

Sign: Given a message M and a secret key (s, e, a), the algorithm outputs a signature σ as follows:

- Choose a uniformly random polynomial y in $\mathcal{R}$, $\|y\|_\infty < B$.
- Let $c = H([ay]_m, M)$.
- $z = y + sc$.
- If $\|z\|_\infty > B - L_S$, then resample y.
- If $ay - ec$ is not well rounded, then resample y.
- Return the siganture $\sigma = (z, c)$.

Verify: Starting with a message M, public key (a, t) and a signature σ, the algorithm outputs 'Accepted' if the signature is valid, otherwise 'Rejected'.

- If $\|z\|_\infty > B - L_S$, then return reject.
- Let $w = az - tc \mod q$, if $H([w]_m, M) \neq c$, return reject.
- Return accept.

5.2.3 FALCON Family

In this subsection, we given an overview of the FALCON scheme, which was submitted to the NIST Post-Quantum Standardization process [74]. FALCON is a lattice-based signature scheme from NTRU assumptions. It stands for Fast Fourier lattice-based compact signatures over NTRU.

Key Generation: Let Φ is a cyclotomic polynomial which is monic and irreducible, and q be a modulus that can be either:

- Binary case: $\Phi = x^n + 1$ such that n is a power of 2, $q = 12289$.
- Ternary case: $\Phi = x^n - x^{n/2} + 1$, such that n is 3 times a power of 2, and $q = 18433$.

Let $\beta > 0$ be a real bound. The FALCON private key consists of four short polynomials $f,\ g,\ F,\ G \in \mathbb{Z}/(\Phi)$, verifying the NTRU equation:

$$fG - gF = q \mod \Phi$$

The FFT representation of $f,\ g,\ F$ and G, ordered in the form of the following matrix:

$$\hat{B} = \begin{bmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{bmatrix}$$

The FALCON public key is a polynomial $h \in \mathbb{Z}/(\Phi)$ such that:

$$h = gf^{-1} \mod (\Phi, q)$$

Sign: It takes a secret key and a message M, the signer uses his secret key to sign M as follows:

- Generate a random salt r uniformly in $\{0, 1\}^{320}$.
- Compute a hash value $c \in \mathbb{Z}/(\Phi)$ from a message M and a salt r.

- Using the knowledge of the secret key $(f,\ g,\ F,\ G)$, the signer computes two short vectors s_1 and s_2 in $\mathbb{Z}/(\Phi)$ such that $s_1 + s_2 h = c \mod q$.
- s_2 is compressed to a bitstring s as specified in [74].
- The signature σ consists of the pair $(r,\ s)$.

Verify: The signature verification procedure is much simpler than the key pair generation and the signature generation. Starting with a public key h, a message M, a signature $\sigma(r, s)$ and an acceptance bound β, the verifier checks that the signature σ is a valid signature for the message M using the following steps:

- Compute a hash value $c \in \mathbb{Z}/(\Phi)$ from a message M and a salt r.
- s is decompressed to a polynomial $s_2 \in \mathbb{Z}/(\Phi)$.
- The value $s_1 = c - s_2 h \mod q$ is computed.
- If $\|(s_1, s_2)\| \leq \beta$, the the signature is accepted. Otherwise, rejected.

5.2.4 pqNTRUSign Family

In this subsection, we given an overview of the pqNTRUSign scheme, which was submitted to the NIST Post-Quantum Standardization process [171].
Key Generation: Let $\mathcal{R} = \mathbb{Z}_q/(x^N \pm 1)$, where N and q are public parameters. Let $f,\ g$ and h be 3 polynomials in $\mathcal{R}$, where f and g are invertible polynomials with small coefficients (less than some real number B_k); $h = p^{-1} g f^{-1}$ for some integer p. The secret key is (f, g), and the public key is the polynomial h.

Sign: Takes a message M, public key h, secret key (f, g) and a distribution χ_t, and outputs a signature as follows:

- Calculate $\mathsf{H}(M|h) = (u_p, v_p) \in \mathcal{R}^2$.
- Sample a polynomial r from the distribution χ_t, and a random bit b.
- Let $u_1 = p \cdot r + u_p$; $v_1 = u_1 \cdot h \mod q$
- Let $a = (v_p - v_1)/g \mod p$
- Check that the norms of $a \cdot f$ and $a \cdot g$ are bounded by some small parameters B_s and B_t respectively, if not then resample r and b.
- Let $v = v_1 + (-1)^b a \cdot g$. If $\|v\|_\infty > q/2 - B_t$, then resample r and b.
- Return the signature $\sigma = (r + (-1)^b a f)$.

Verify: Takes the public parameters, a message M, the public key h and a signature σ as inputs. It accepts or rejects the signature using the following steps:

- Check that $\mathsf{H}(M|h) = (u_p, v_p)$
- Set $u = p\sigma + u_p$

- if $\|u\|^2 > p^2t^2N$, reject.
- $v = u \cdot h \mod q$.
- If $v \not\equiv v_p \mod p$ or $\|v\|_\infty > q/2 - B_t$, reject, otherwise accept.

5.2.5 SPHINCS Family

SPHINCS is a hash-based signature scheme that was submitted to the NIST PQC standardization initiative by Dan Bernstein et al. A major advantage of hash-based signature schemes is that their security relies solely on the properties of the underlying cryptographic hash function. However, most hash-based signature schemes, such as XMSS and LMS, are stateful, which means the signer has to update the secret key with every signature generation. SPHINCS, on the other hand, is *stateless* and can, therefore, serve as a drop-in replacement for currently-used signature schemes like RSA and ECDSA.
The hash tree used by SPHINCS is a hypertree that is composed of several layers of hash trees. Winternitz One-Time Signature (WOTS$^+$) keys are contained in the leaf nodes of top layers in the hypertree. These keys are used for signing the root nodes of trees on lower levels. The Few-Time Signature (FTS) scheme HORS with Trees (HORST) is used for signing messages, whereby the HORST keys are stored in the leaf nodes of the lowermost layer of the trees. Stateful hash-based signature schemes based on Merkle trees store a leaf index counter as part of the private key. In order to prevent reusing the same key pair, it is necessary that this counter is updated for every new signature generation. SPHINCS is a stateless signature scheme because it picks the leaf index belonging to a key pair randomly and does not take into account whether the key pair has been used before. However, this randomized leaf index selection carries the risk that the same key pair is used twice or even several times. Using a One-Time Signature scheme would lead to a complete compromise of the system, but SPHINCS minimizes this threat by employing an FTS scheme.

Keypair Generation: In order to generate a SPHINCS key pair, one has to first sample two secret values $(\mathsf{SK}_1, \mathsf{SK}_2) \in \{0,1\}^n \times \{0,1\}^n$, where n is the main security parameter. SK_1 serves as input for a pseudo-random key generation, while SK_2 is used to generate an unpredictable index and to randomize the message hash. Furthermore, a few n-bit bitmasks $\mathbf{Q}$ are generated, which are used for all WOTS$^+$ and HORST instances as well as for the hash trees. Next, the root node has to be generated, which requires the generation of the WOTS$^+$ key pairs for the topmost tree. All these keys are generated with the help of the pseudo-random key generator initialized with SK_1. The final step is to build the hash tree and calculate the root node value PK_1 using the leaf nodes, which contain WOTS$^+$ public keys. The private key consists of $(\mathsf{SK}_1, \mathsf{SK}_2, \mathbf{Q})$ and the public key consists of $(\mathsf{PK}_1, \mathbf{Q})$.

Signature Generation: Let $M \in \{0,1\}^*$ denote the message to be signed. At first, a $2n$-bit pseudo-random value $R = (R_1, R_2)$ is generated using a pseudo-random function that takes M and SK_2 as inputs. Next, a randomized message digest D is derived from M whereby R_1 serves as source of randomness. In order to generate a signature for M, the signer chooses a HORST key pair using an index i, which is derived from R_2 and determines both the tree and the leaf index inside the chosen tree. The SPHINCS signature for M consists of the index i, the randomness R_1, and a HORST signature σ_H. In addition, a WOTS$^+$ signature and an authentication path per layer of trees is included, both of which are necessary to ensure that the signature can be verified. They are obtained as part of the signing process by generating a binary hash tree for

Scheme	n	k	q	NIST	Assumption	Φ	pk size (bytes)	sign size (bytes)
Dilithium	768	3	8380417	1	MLWE	$x^{n/k}+1$	1184	2044
Dilithium	1024	4	8380417	2	MLWE	$x^{n/k}+1$	1472	2701
Dilithium	1280	5	8380417	3	MLWE	$x^{n/k}+1$	1760	3366
qTESLA	1024	-	8058881	1	RLWE	x^n+1	2720	2976
qTESLA	2048	-	12681217	3	RLWE	x^n+1	5664	6176
qTESLA	2048	-	27627521	5	RLWE	x^n+1	5920	6432

Table 5.1: Parameter sets for LWE-based signature schemes with dimension n, modulo q, (rank k in case of MLWE) over the ring $\mathbb{Z}_q/(\Phi(x))$. The NIST column indicates the NIST security category aimed at. The table also compares the public key and signatures sizes in both schemes.

Scheme	n	q	$\|f\|$	$\|g\|$	NIST	Assumption	Φ	pk size (bytes)	sign size (bytes)
FALCON	512	12289	91.71	91.71	1	NTRU	x^n+1	897	618
FALCON	768	18433	112.32	112.32	2,3	NTRU	$x^n - x^{n/2}+1$	-	-
FALCON	1024	12289	91.71	91.71	4,5	NTRU	x^n+1	1793	1233
pqNTRU-sign	1024	65537	22.38	22.38	1,2,3, 4,5	NTRU	x^n-1	2048	1408

Table 5.2: Parameter sets for NTRU-based signature schemes with dimension n, modulo q, small polynomials f and g, and ring $\mathbb{Z}_q/(\Phi(x))$. The table also compares the public key and signatures sizes in both schemes.

each layer of the SPHINCS hypertree. The signature generation is a fully deterministic process since all required randomness is generated with the help of a pseudo-random function.

Signature Verification: The verification of the SPHINCS signature consists of (i) the verification of the HORST signature σ_H and (ii) the verification of a WOTS$^+$ signature and authentication path per layer of trees. In this way, the verifier will obtain a value for the root node, which serves as proof for the validity of the signature. Namely, the signature is valid if the obtained value equals PK$_1$ of the public key.

5.3 Candidates Comparison

We present a comparison among the four lattice-based signature schemes, introduced in the previous subsection. Table 5.1 shows a comparison between two LWE-based schemes, Dilithium and qTESLA, while Table 5.2 shows a comparison between two NTRU-based schemes, FALCON and pqNTRUSign.

5.4 Open Issues

The following open issues will be considered in the FutureTPM project:

- It is not clear yet whether the proposed candidates of the signature schemes are efficient enough for inclusion in a TPM. This will be further investigated in the FutureTPM project.

- There is a trade-off between the quantum-resistance of digital signature schemes and performance of the scheme implementation. Finding a right balance on this trade-off is not trivial.

- Researchers are working on side-channel attacks and fault attacks to quantum-resistant signature schemes, including some of the proposed signature candidates listed in this section. How to integrate solutions from this research into this project and how to identify an appropriate balance between algorithm robustness against these attacks and algorithm performance suitable for inclusion in a TPM is challenging.

Chapter 6

Public-Key Encryption and Key Exchange

Public Key Encryption (PKE) is the core of secure communication, as it allows any party to use a publicly available key to safely send messages to the owner of the corresponding secret key. Formally, PKE can be defined as it follows.

Definition 20 *A Public Key Encryption scheme (PKE) is a triple of PPT algorithms* $(\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ *such that*

- *On input the security parameter* 1^λ *the key generation algorithm* Kgen *outputs a pair of keys* $(\mathsf{sk}, \mathsf{pk})$*. Without loss of generality we can assume that the secret key* sk *includes both the security parameter* 1^λ *and the public key* pk*, and* pk *includes also* 1^λ*.*
- *The encryption algorithm* Enc *takes as input the public key* pk *and a plaintext* msg *and outputs a ciphertext* c *that is an encryption of the message* msg *under* pk*.*
- *The decryption algorithm* Dec *takes as input a ciphertext* c *and the secret key* sk *and outputs a plaintext* msg*.*

In the following section we will discuss the security definitions of the PKE.

6.1 Security Models

Security of a PKE essentially requires that the encryptions of two messages be indistinguishable. Sometimes it is required for the ciphertext to be also non-malleable, i.e., that it should be impossible transform the encryption of a message msg_1 in the encryption of the message msg_2 without knowing the secret key.

6.1.1 Indistinguishability

Analogously to block ciphers, a PKE should not leak any information about the plaintext. This property can be formulated as indistinguishability of ciphertexts (IND). The definition is given with respect to an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ modeled as a pair of Turing machines that are allowed to share a state. In QS0 these algorithm are modeled as Probabilistic Polynomial Time (PPT) Turing machines. Security is defined as a game between the adversary and a challenger $\mathcal{C}$. The challenger generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\lambda)$ and sends pk to $\mathcal{A}_1$. Upon receiving pk, the algorithm $\mathcal{A}_1$ outputs a pair of challenge messages $\mathsf{msg}_1, \mathsf{msg}_2$ and sends them to $\mathcal{C}$. The challenger samples a random bit $b \xleftarrow{\$} \{0, 1\}$, encrypts the message msg_b running $\mathsf{c}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_b)$, and sends c_b to $\mathcal{A}_2$. On input the challenge ciphertext c_b (and, possibly, a state generated from $\mathcal{A}_1$), the algorithm $\mathcal{A}_2$ outputs a guess b' on the bit b. The adversary wins the game if its guess on the bit is right. A PKE scheme is secure if the adversary has no better strategy than randomly guessing the bit.

Definition 21 (Indistinguishability in QS0) *A PKE scheme is IND secure iff, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of the adversary in winning the IND game is negligible, i.e.,*

$$\left|\Pr(\mathcal{A} \textit{ wins the IND game}) - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda) \ .$$

In particular, in this case the scheme is said to be indistinguishable under chosen-plaintext attacks (IND-CPA), as giving the public key as input to $\mathcal{A}_1$ is equivalent to give it access to an encryption oracle. If $\mathcal{A}_1$ is also given oracle access to the decryption $\mathsf{Dec}(\mathsf{sk}, \cdot)$, the scheme is said to be indistinguishable under non-adaptive chosen-ciphertext attacks (IND-CCA1). Finally, if also $\mathcal{A}_2$ is given oracle access to a decryption oracle $\mathsf{Dec}^*(\mathsf{sk}, \cdot)$, the PKE is said to be indistinguishable under adaptive chosen-ciphertext attacks (IND-CCA2). Caution has to be taken in this last case, as, for the definition to be meaningful, $\mathcal{A}_2$ should not be allowed to query the challenge ciphertext c_b to the decryption oracle. Hence, the decryption oracle is defined as:

$$\mathsf{Dec}^*(\mathsf{sk}, \mathsf{c}) := \begin{cases} \perp, & \textit{if } \mathsf{c} = \mathsf{c}_b \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{c}), & \textit{otherwise} \end{cases} \ . \tag{6.1}$$

The definition of indistinguishability against a quantum adversary can be done in different ways, depending on how the adversary is modeled. In QS1 scenario, the adversary is modeled as a quantum Turing machine, but still has classical access to the oracles. The definition of indistinguishability in QS1 is therefore straightforward.

Definition 22 (Indistinguishability in QS1) *A PKE scheme is post-quantum IND secure (pqIND) iff, for all QPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of the adversary in winning the IND game is negligible, i.e.,*

$$\left|\Pr(\mathcal{A} \textit{ wins the IND game}) - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda) \ .$$

In this case the scheme is said to be post-quantum indistinguishable under chosen-plaintext attacks (pqIND-CPA). If $\mathcal{A}_1$ is also given oracle access to the decryption $\mathsf{Dec}(\mathsf{sk}, \cdot)$, the scheme is said to be post-quantum indistinguishable under non-adaptive chosen-ciphertext attacks (pqIND-CCA1). Finally, if also $\mathcal{A}_2$ is given oracle access to the decryption oracle $\mathsf{Dec}^*(\mathsf{sk}, \cdot)$ defined in Equation (6.1), the PKE is said to be post-quantum indistinguishable under adaptive chosen-ciphertext attacks (pqIND-CCA2).

A more powerful definition of the adversary allows $\mathcal{A}$ to have quantum oracle access to the decryption oracle. To model this scenario, the decryption oracle is modeled as the following unitary gate (cf. []):

$$U_{\mathsf{Dec}(\mathsf{sk}, \cdot)} \left|x, y\right\rangle := \left|x, y \oplus \mathsf{Dec}(\mathsf{sk}, x)\right\rangle \ .$$

Analogously, the second decryption oracle Dec^* is defined as

$$U_{\mathsf{Dec}^*(\mathsf{sk}, \cdot)} \left|x, y\right\rangle := \begin{cases} \left|\perp\right\rangle, & \text{if } x = \mathsf{c}_b \\ \left|x, y \oplus \mathsf{Dec}(\mathsf{sk}, x)\right\rangle, & \text{otherwise} \end{cases}$$

where again the equality check is performed in quantum superposition. The structure of the IND game remains the same.
Remark that the definition of indistinguishability under quantum chosen-plaintext attacks (IND-qCPA) is equal to pqIND-CPA, as there is no encryption oracle in the game. Hence, we only give the definitions of indistinguishability under chosen-ciphertext attacks in QS2.

Definition 23 (Indistinguishability under chosen-ciphertext attacks in QS2) *A PKE scheme is IND secure under non-adaptive quantum chosen-ciphertext attacks (IND-q) iff, for all QPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of the adversary in winning the IND game is negligible, i.e.,*

$$\left| \Pr(\mathcal{A} \textit{ wins the IND game}) - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda) \ ,$$

where $\mathcal{A}_1$ is given access to the quantum decryption oracle $U_{\mathsf{Dec}(\mathsf{sk},\cdot)}$ If also $\mathcal{A}_2$ is given oracle access to the decryption oracle $U_{\mathsf{Dec}^*(\mathsf{sk},\cdot)}$, the PKE is said to be indistinguishable under adaptive quantum chosen-ciphertext attacks (IND-qCCA2).

6.1.2 Non-Malleability

Non-malleability is a stronger security requirement for encryption first defined by Dolev, Dwork and Naor [67]. In the following we give the definition by Pass et al. [131]. The high-level idea is to prevent an adversary to tamper with the ciphertext so that it will not decrypt to the original plaintext anymore. The formal definition of non-malleability is based on a game between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ modeled as a pair of Turing machines and a challenger $\mathcal{C}$. The challenger generates the key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\lambda)$ and sends pk to $\mathcal{A}_1$. The adversarial algorithm $\mathcal{A}_1$ sends back to $\mathcal{C}$ a pair of messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger samples a uniform random bit $b \xleftarrow{\$} \{0,1\}$ and sends to $\mathcal{A}_2$ the encryption of msg_b generated as $\mathsf{c}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_b)$. Upon receiving c_b (and possibly a state from $\mathcal{A}_1$), $\mathcal{A}_2$ outputs a tuple of ciphertexts $(\mathsf{c}'_1, \dots, \mathsf{c}'_\ell)$. The challenger outputs $(d_1, \dots, d_\ell)$ such that

$$(d_1, \dots, d_\ell) = \begin{cases} \bot, & \text{if } \mathsf{c}'_i = \mathsf{c}_b \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{c}'_i), & \text{otherwise} \end{cases} .$$

Depending on the value of the random bit b, we call this game $\mathsf{NME}(\ell, b, \mathcal{A})$. Non-malleability requires the game $\mathsf{NME}(\ell, 1, \mathcal{A})$ to be indistinguishable from $\mathsf{NME}(\ell, 0, \mathcal{A})$ for a distinguisher $\mathcal{D}$ that on input the game NME_b, outputs a guess on b.

Definition 24 (Non-Malleability in QS0) *A PKE scheme is non-malleable if for all PPT adversaries $\mathcal{A}$ and non-uniform PPT distinguisher $\mathcal{D}$*

$$|\Pr(\mathcal{D}(\textit{NME}(\ell, 1, \mathcal{A})) = 1) - \Pr(\mathcal{D}(\textit{NME}(\ell, 0, \mathcal{A})) = 1)| \leq \mathsf{negl}(\lambda) \ .$$

Non-malleability in QS1 can be obtained from the definition in QS0 by modeling the adversary as a quantum Turing machine.

Definition 25 (Non-Malleability in QS1) *A PKE scheme is post-quantum non-malleable if for all QPT adversaries $\mathcal{A}$ and non-uniform PPT distinguisher $\mathcal{D}$*

$$|\Pr(\mathcal{D}(\textit{NME}(\ell, 1, \mathcal{A})) = 1) - \Pr(\mathcal{D}(\textit{NME}(\ell, 0, \mathcal{A})) = 1)| \leq \mathsf{negl}(\lambda) \ .$$

Given that the adversary is not given access to any oracle in the security game, the definition of non-malleability in QS1 can be used also in a QS2 scenario. This does not take into account more complex scenarios, like the case in which the adversary is allowed to send as challenge plaintexts superimpositions of messages, that are still open problems.

```
$\mathsf{Kgen}_{\mathrm{FO}}(1^\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\lambda)$
$\mathsf{pk}^* = \mathsf{pk}$
$\mathsf{sk}^* = \mathsf{sk}$

$\mathsf{Enc}_{\mathrm{FO}}(\mathsf{pk}^*, \mathsf{msg})$:
$\sigma \xleftarrow{\$} \{0,1\}^\ell$
$\mathsf{c}_1 = \mathsf{Enc}(\mathsf{pk}, \sigma; \mathsf{H}(\sigma))$
$\mathsf{c}_2 = \mathsf{Enc}_{\mathcal{S}}(\mathsf{H}_{\mathcal{S}}(\sigma), \mathsf{msg})$
$\mathsf{c}^* = (\mathsf{c}_1, \mathsf{c}_2)$

$\mathsf{Dec}_{\mathrm{FO}}(\mathsf{sk}^*, \mathsf{c}^*)$:
Compute $\sigma = \mathsf{Dec}(\mathsf{sk}, \mathsf{c}_1)$.
If $\mathsf{c}_1 \neq \mathsf{Enc}(\mathsf{pk}, \sigma; \mathsf{H}(\sigma))$ then
  output $\perp$
else
  output $\mathsf{Dec}_{\mathcal{S}}(\mathsf{H}_{\mathcal{S}}(\sigma), \mathsf{c}_2)$
```

Protocol 6.1: Fujisaki-Okamoto Construction.

6.1.3 Transformations

A public key encryption that is only IND-CPA can be transformed into an IND-CCA2 PKE when combined with other cryptographic primitives. In this section we present three different IND-CPA to IND-CCA2 transforms from literature.

Fujisaki-Okamoto (FO) Transformation

The Fujisaki-Okamoto [75] construction allows to obtain an IND-CCA2 encryption scheme combining an IND-CPA encryption scheme $(\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ with a CCA secure symmetric encryption scheme $(\mathsf{Kgen}_{\mathcal{S}}, \mathsf{Enc}_{\mathcal{S}}, \mathsf{Dec}_{\mathcal{S}})$ and two hash functions H, $\mathsf{H}_{\mathcal{S}}$ that map bit strings to the random coin space of $(\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ and to the key space of $(\mathsf{Kgen}_{\mathcal{S}}, \mathsf{Enc}_{\mathcal{S}}, \mathsf{Dec}_{\mathcal{S}})$ respectively. The construction is shown in Figure 6.1. Remark that with $\mathsf{Enc}(\mathsf{pk}, \sigma; \mathsf{H}(\sigma))$ we denote the encryption of σ with randomness $\mathsf{H}(\sigma)$ w.r.t the public key pk.
A variant of the FO transform was proved secure in the Quantum Random Oracle Model (QS2) by Targhi and Unruh [154].

Naor-Yung (NY) Transformation

The construction by Naor and Yung [121] allows to obtain IND-CCA1 security combining the PKE with a non-interactive zero-knowledge proof (NIZK). A NIZK is a protocol that allows a prover to prove that she knows a secret to a verifier, without leaking information about the secret.

Definition 26 (NIZK) *A NIZK proof is a pair of algorithms $(\mathcal{P}, \mathcal{V})$ where $\mathcal{P}$ is a probabilistic algorithm and $\mathcal{V}$ is a deterministic algorithm such that, given a relation $\mathcal{R}$, on input a public x and a secret w such that (x, w) satisfies $\mathcal{R}$, $\mathcal{P}$ produces a proof Π that she knows w. On input x and Π, the verifier outputs 1 if the proof Π is accepted, 0 otherwise. In particular, the proof has to have the following property:*

- *Completeness: a honestly generated proof is always accepted by the verifier.*
- *Soundness: a prover cannot generate a valid proof without knowing a valid witness w.*
- *Zero-Knowledge: there exists a polynomial time simulator $\mathcal{S}$ that on input x outputs a valid proof Π^* such that a PPT adversary cannot distinguish Π^* from an honestly generated proof.*

The construction is extremely easy and it is shown in Figure 6.2.
This construction can be modified to obtain IND-CCA2 security in different ways. In particular, Sahai [139] proved that one condition for the NY construction to be IND-CCA2 is for the NIZKs to be also *simulation sound* (this construction was proved to be IND-CCA2 secure by Faust et

```
$\mathsf{Kgen}_{\mathrm{NY}}(1^\lambda)$:                 $\mathsf{Enc}_{\mathrm{NY}}(\mathsf{pk}^*, \mathsf{msg})$:          $\mathsf{Dec}_{\mathrm{NY}}(\mathsf{sk}^*, \mathsf{c}^*)$:
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Kgen}(1^\lambda)$   $\mathsf{c}_1 = \mathsf{Enc}(pk_1, \mathsf{msg})$    If $\mathcal{V}((\mathsf{c}_1, \mathsf{c}_2), \Pi) = 0$ then
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{Kgen}(1^\lambda)$   $\mathsf{c}_2 = \mathsf{Enc}(pk_2, \mathsf{msg})$        output $\perp$
$\mathsf{pk}^* = (\mathsf{pk}_1, \mathsf{pk}_2)$              $\Pi = \mathcal{P}((\mathsf{c}_1, \mathsf{c}_2), \mathsf{msg})$  else
$\mathsf{sk}^* = \mathsf{sk}_1$                       $\mathsf{c}^* = (\mathsf{c}_1, \mathsf{c}_2, \Pi)$           output $\mathsf{Dec}(\mathsf{sk}_1, \mathsf{c}_1)$
```

Protocol 6.2: Naor-Yung Construction.

```
$\mathsf{Kgen}(1^\lambda)$:                 $\mathsf{Enc}(\mathsf{pk}^*, \mathsf{msg})$:                $\mathsf{Dec}(\mathsf{sk}^*, \mathsf{c}^*)$:
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda)$   $(\mathsf{c}_k, \mathsf{sk}) = \mathsf{KE}(pk_1, \mathsf{msg})$   $\mathsf{sk} \leftarrow \mathsf{KD}(\mathsf{dk}, \mathsf{c}_k)$
$\mathsf{pk}^* = \mathsf{ek}$                   $\mathsf{c} = \mathcal{E}(\mathsf{sk}, \mathsf{msg})$               $\mathsf{msg} \leftarrow \mathcal{D}(\mathsf{sk}, \mathsf{c})$
$\mathsf{sk}^* = \mathsf{dk}$                   $\mathsf{c}^* = (\mathsf{c}_k, \mathsf{c})$
```

Protocol 6.3: PKE from KEM Construction.

al. [72] in the Random Oracle Model). Otherwise, Dolev-Dwork-Naor [66] obtained IND-CCA2 security adding to the construction a *one-time strong signature scheme*, i.e., a signature scheme such that an adversary seeing only a signature on a message msg cannot come up with a new signature on msg. There are no security proofs of the construction in the Quantum Random Oracle Model.

6.1.4 Construction

As we have seen in the previous section, it is possible to build an IND-CCA2 PKE starting from an IND-CPA PKE. In this section we explore how to construct a basic (IND-CPA) PKE.
PKE can be constructed from a more general primitive called *Key Encapsulation Mechanism* (KEM). KEMs allow to publish a public key such that anybody can safely transmit to the owner of the public key an encryption of a secret key. This primitive is more general than PKE, and it is used in the setup phases of protocols for secure communication, such as TLS.

Definition 27 *A* Key Encapsulation Mechanism *(KEM) is a triple of PPT algorithms* $(\mathsf{KG}, \mathsf{KE}, \mathsf{KD})$ *such that*

- *The key generation algorithm* KG *takes as input the security parameter* 1^λ *and generates two keys, an encapsulation key* ek *and a decapsulation key* dk.
- *The key encapsulation mechanism* KE *takes as input the public encapsulation key* ek *and outputs a pair* $(\mathsf{c}, \mathsf{sk})$ *where* c *is the encryption of* sk.
- *The key decapsulation mechanism* KD *takes as input the decapsulation key* dk *and the ciphertext* c *and outputs the secret key* sk.

Security definitions for KEMs can be obtained adapting the indistinguishability definitions of PKEs in a straightforward way.
Blum and Goldwasser [40] showed that it is possible to obtain a PKE from a KEM combining it with a symmetric key encryption (SKE) scheme $(\mathcal{E}, \mathcal{D})$. The construction is shown in Figure 6.3
In particular, an IND-CPA PKE can be obtained from an IND-CPA KEM and a SKE that has indistinguishable encryptions in the presence of an eavesdropper (IND secure, cf. Definition 4 in Section 4.1.1).

6.2 Proposed Candidates

In this section we list promising candidates for post-quantum key exchange and public-key encryption from the NIST standardization process [129].

6.2.1 NewHope Family

The NewHope cryptosystem [9] is a suite of key encapsulation mechanisms (KEM) denoted as NewHope-CPA-KEM and NewHope-CCA-KEM that are based on the conjectured quantum hardness of the RLWE problem [108]. Both schemes are based on a variant of the previously proposed NewHope-Simple [12] scheme modeled as semantically secure public-key encryption (PKE) scheme with respect to adaptive chosen plaintext attacks (CPA) that is called NewHope-CPA-PKE. However, NewHope-CPA-PKE is only used inside of NewHope-CPA-KEM and NewHope-CCA-KEM and not intended to be an independent CPA-secure PKE scheme, in part because it does not accept arbitrary length messages. For NewHope-CPA-KEM a transformation of NewHope-CPA-PKE into a passively secure KEM is provided. For NewHope-CCA-KEM it is shown how to realize a semantically secure key encapsulation with respect to adaptive chosen ciphertext attacks (CCA) based on NewHope-CPA-PKE. The CCA transformation in HewHope is based on the Fujisaki-Okamoto transformation [75] which is a standard technique in cryptography and has recently been adopted to the quantum setting by Targhi and Unruh [154] (see Section 6.1.3). The general idea to prevent ciphertext malleability is to check during decryption that the ciphertext is a valid and correctly generated ciphertext by performing a re-encryption. This is possible when the underlying CPA-secure PKE scheme is used to encrypt the random seed used by all PRNGs during encryption. Moreover, the ability of an attacker to influence the choice of a seed is restricted by using several instantiations of a random oracle. We refer to Algorithms 7, 8, 9 that specify the CCA transformation for the Kyber scheme. For NewHope the Kyber.CPA primitive can basically be replaced by the NewHope.CPA primitive.
To show how NewHope works we describe the NewHope-Simple protocol in the following as a key exchange scheme where it allows two entities (Alice and Bob) to agree on a 256-bit shared key μ that is selected by Bob. Let $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ be a ring of integer polynomials. All elements of the ring $\mathcal{R}_q$ can be written in the form $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$, where the integer coefficients $a_0, a_1, \ldots, a_{n-1}$ are reduced modulo q. Let Ψ_k be a binomial distribution with parameter k. The distribution is determined by $\Psi_k = \sum_{i=0}^{k-1} b_i - b'_i$, where $b_i, b'_i \in \{0, 1\}$ are uniform independent bits. The binomial distribution is centered with a zero mean, approximates a discrete Gaussian, has variance $k/2$, and gives a standard deviation of $\psi = \sqrt{k/2}$.
Protocol 6.4 shows the underlying algorithm of NewHope Simple where we highlight important steps.

1. Alice samples the seed from a random number generator. The seed is expanded with the SHAKE-128 extendable-output function. The expanded seed is used to generate the uniformly random public polynomial a using the Parse function.

2. Alice and Bob randomly sample the coefficients of the secret polynomials s and s', and the error polynomials e, e' and e'' according to the error distribution Ψ_k, which is denoted by $\xleftarrow{\$} \Psi_k$.

3. Alice calculates $b = as + e$ and sends it together with the seed to Bob. Extraction of the secret s from b is hard due to the error term e and because b is exactly an RLWE instance. Similar to Alice, Bob can use the seed to generate the public polynomial a.

Alice (server):		**Bob (client):**
① seed $\xleftarrow{\$} \{0,1\}^{256}$		
$a \leftarrow$ Parse(SHAKE(seed))		
② $s, e \xleftarrow{\$} \Psi_{16}^{n}$		② $s', e', e'' \xleftarrow{\$} \Psi_{16}^{n}$
③ $b \leftarrow as + e$	$\xrightarrow{m_a = \text{encodeA}(b,\text{seed})}$	③ $(b,\text{seed}) \leftarrow$ decodeA(m_a)
		$a \leftarrow$Parse(SHAKE(seed))
		④ $v \xleftarrow{\$} \{0,1\}^{256}$
		$d \leftarrow$ NHSEncode(v)
		⑤ $u \leftarrow as' + e'$
		$c \leftarrow bs' + e'' + d$
⑥ $(u, \overline{c}) \leftarrow$ decodeB(m_b)	$\xleftarrow{m_b = \text{encodeB}(u, \overline{c})}$	⑥ $\overline{c} \leftarrow$ NHSCompress(c)
$c' \leftarrow$ NHSDecompress$(\overline{c})$		⑧ $\mu \leftarrow$ SHA3-256(v)
⑦ $d' \leftarrow c' - us$		
$v' \leftarrow$ NHSDecode(d')		
⑧ $\mu \leftarrow$ SHA3-256(v')		

Protocol 6.4: NewHope Simple protocol. All polynomials are elements of the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, where $n = 1024$ and $q = 12289$ [12].

4. Bob samples 256 bits from a random number generator and assigns them to the secret key vector v. Then, Bob encodes v into the most significant bit of the coefficients of polynomial d = NHSEncode(v). The functions NHSEncode and NHSDecode of NewHope Simple build an error-correcting code, which is used to remove small errors and to increase the probability that Alice and Bob share a similar key. The function NHSEncode, maps one bit of v into two or four coefficients of d. This redundancy is used by the NHSDecode function in Step 7 to average out small errors.

5. Bob calculates $u = as' + e'$ and hides the secret key polynomial d in $c = bs' + e'' + d = ass' + es' + e'' + d$. The polynomials u and c are again instances of the RLWE problem.

6. Bob sends to Alice the polynomial u and the compressed polynomial $\overline{c}$. The goal of the compression of polynomial c is the reduction of the communication overhead between Alice and Bob.

7. Alice removes the large noise term ass' from the decompressed polynomial c' by calculating $d' = c' - us \approx bs' + e'' + d - (as' + e')s = ass' + es' + e'' + d - ass' - e's = (es' - e's) + e'' + d$. Alice obtains the term v' after decoding d', using the function NHSDecode.

8. After the decoding, Alice and Bob can use v' and v, respectively, as input for the SHA3-256 function to obtain the shared key.

The security level of NewHope depends on three parameters: the dimension n of the ring, the modulus q, and the parameter k that determines the standard deviation of the noise distribution Ψ_k. In Table 6.1 we provide an overview over the various NewHope instantiations and their parameters. We also list the claimed security level/category according to the methodology outlined by NIST in their submission guidelines for the standardization process [128].
An advantage of NewHope is that it achieves high performance on a wide range of platforms, is memory efficient and relatively easy to implement. The design can be considered conservative

Parameter Set	n	q	k	q. b.-sec.	NIST cat.	fail	$\|pk\|$	$\|sk\|$	—$ctxt$—
NEWHOPE-USENIX									
NEWHOPE-SIMPLE									
NEWHOPE512-CPA-KEM	512	12289	8	101	1	2^{-213}	928	869	1088
NEWHOPE1024-CPA-KEM	1024	12289	8	233	5	2^{-216}	1824	1792	2176
NEWHOPE512-CCA-KEM	512	12289	8	101	1	2^{-213}	928	1888	1120
NEWHOPE1024-CCA-KEM	1024	12289	8	233	5	2^{-216}	1824	3680	2208

Table 6.1: Parameters proposed for instantiating NewHope in the submission to the NIST process.

as there is a considerable security margin over the claimed 233-bits or 101-bits of security and as RLWE is a standard problem. The main disadvantages are limits in parametrization as with the current structure of NewHope it is not straightforward to construct a scheme that achieves NIST security category 2, 3, or 4 as either ring dimension n=512 or n=1024 has to be used. Moreover, NewHope explicitly specified how polynomial multiplication has to be implemented using the Number Theoretic Transform (NTT) to optimize performance. While the NTT has in general proven to be the method of choice for fast polynomial multiplication, this certainly restricts implementers in their choice to use an optimal multiplication algorithm.

6.2.2 Frodo Family

The FrodoKem [10] is a submission to the NIST PQC process and the Leaning with Errors (LWE) analogue of RLWE-based key exchange scheme introduced by Bos, Costello, Naehrig and Stebila in 2015 [46]. The FrodoKEM scheme has been modified for the NIST process and is an updated version of the Frodo scheme [45]. FrodoKEM only supports IND-CCA security and uses the Fujisaki-Okamoto transformation to transform a CPA-secure PKE scheme into a CCA-secure PKE scheme. The security of FrodoKEM is based on the LWE problem, which is related to the hardness of lattice problems. For positive integers n, q, an (error) distribution χ over $\mathbb{Z}$, and $s \in \mathbb{Z}_q^n$ the LWE distribution is defined for $a \in \mathbb{Z}_q^n$ sampled uniformly random and $e \in \mathbb{Z}$ sampled from χ as the pair $(a, \langle a, s\rangle + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.
To optimize the performance of FrodoKem, the authors suggested four different distributions for the error distribution χ, all faster to sample than a discrete Gaussian distribution. Let χ be a centered error distribution with support $\{-\eta, .., \eta\}$ over $\mathbb{Z}_q^{n\times\overline{n}}$, $q \in \mathbb{Z}, n, \overline{n}, m, \overline{m} \in \mathbb{Z}$, $B \in \{0, .., \log_2(q) - 2\}$ and $\overline{B} = \log_2(q) - B$. For a matrix V over $\mathbb{Z}_q$ define the following rounding functions entry-wise:

$$\lfloor v \rceil 2^B = \lfloor 2^{-\overline{B}} v \rfloor \mod 2^B$$
$$\langle v \rangle_{2^B} = \lfloor 2^{-\overline{B}+1} v \rfloor \mod 2^B$$

For two matrices entry-wise define the reconciliation rec for $w, b \in \mathbb{Z}_q$ as $\lfloor v \rceil 2^B$, where v is the closed element to w fulfilling $\langle v \rangle_{2^B} = b$. If two entries v, w have a distance $|v - w| < 2^{\overline{B}-2}$, then $rec(w, \langle v \rangle_{2^B}) = \lfloor v \rceil_{2^B}$. Let Gen denote a pseudo-random function generating an $n \times n$ matrix over $\mathbb{Z}_q$.

The core of the CCA-secure FrodoKEM is the CPA-secure scheme FrodoPKE (closely related to Frodo [45]), whose algorithms are defined in a simplified manner as follows:
In Table 6.2 we list parameters of FrodoKEM. The biggest difference between NewHope and

```
1 $seed_A \xleftarrow{\$} \{0,1\}^s$
2 $A \leftarrow Gen(seed_A)$
3 $E, S \xleftarrow{\$} \chi(\mathbb{Z}_q^{n\times\overline{n}})$
4 $B = AS + E$
5 return $pk := (seed_A, B), sk := S$
```

Algorithm 1: FRODOPKE.KEYGEN

```
  Input: $pk = (seed_A, B)$
1 $A \leftarrow Gen(seed_A)$
2 $E', S' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\overline{m}\times n})$
3 $B' = S'A + E'$
4 $E'' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\overline{m}\times\overline{n}})$
5 $V = S'B + E''$
6 $C = \langle V \rangle_{2^B}$
7 $K = \lfloor V \rceil_{2^B}$
8 return $key := K, rec\_info := C, blinded\_key := B'$
```

Algorithm 2: FRODOPKE.ENC

```
  Input: $rec\_info = C, blinded\_key = B'$
1 $V' = B'S$
2 $K = rec(V', C)$
3 return $key = K$
```

Algorithm 3: FRODOPKE.DEC

FrodoKEM is the choice of the underlying lattice-problem. FrodoKEM explicitly uses the LWE problem which is based on a less structured problem and which is considered a weaker assumption. Currently, no attacks are known that can significantly exploit the additional structure of RLWE but this may change in the future. Additionally, FrodoKEM does not need a error reconcilliation function and can use a very simple encoding and decoding mechanism. This due to the authors' design rationale to put simplicity and security over performance and optimization. The biggest disadvantage of FrodoKEM are its big public-key and ciphertext sizes. In this regard, when comparing NIST level 1 security, the difference between FrodoKEM and NewHope is roughly a factor of 10.

6.2.3 Kyber Family

A recent construction relying on the so-called Modular LWE Problem (MLWE) is the CCA-secure Kyber Key Encapsulation Mechanism. The MLWE problem is a generalization of both RLWE and LWE. Given a tuple $(a, \langle a, s\rangle + e)$ with a, s, e being matrices over $\mathcal{R}_q$, where e is small, the challenge is to find s. In practice it allows easier scaling of security parameters as security levels can be achieved that are hard to reach using RLWE. Kyber has been submitted to the NIST PQC standardization process [144] and a variant is also published as an academic paper [47]. It is defined by an intermediate IND-CPA secure Public-Key Encryption (PKE) scheme which is then transformed to an IND-CCA secure KEM using a generic transform based on the Fujisaki-Okamoto transfom [91][1]. Kyber unambiguously refers to the IND-CCA secure KEM, i.e. [144]

[1]We note that [144] does not include the Targhi-Unruh tag [154].

Parameter Set	n	q	$\overline{n} = \overline{m}$	η	q. bit-sec.	NIST cat.	failure	$\|pk\|$	$\|sk\|$	—$ctxt$—
Frodo-640	640	2^{15}	8	11	103	1	$2^{-148.8}$	9616	19872	9736
Frodo-976	976	2^{16}	8	10	150	3	$2^{-199.6}$	15632	31272	15768

Table 6.2: Parameters proposed for instantiating FrodoKEM in the submission to the NIST process.

does not formally propose a public-key encryption scheme nor a KEM which only claims IND-CPA security. For Kyber we extend $\xleftarrow{\$}$ notation and use $x \xleftarrow{y} z$ to describe that element x was sampled according to distribution z using a binary seed y that is used as initial randomness by the sampler of the distribution. This way the sampler can be made deterministic depending on the seed, which is required for the CCA transformation.

Definition 28 (Simplified Kyber.CPA following [47]; c.f. [144]) *Let k, n, q, η, d_t, d_u, d_v be positive integers, where $n = 256$. Let $\mathcal{M} = \{0,1\}^n$ be the plaintext space, where each message $m \in \mathcal{M}$ can be seen as a polynomial in $\mathcal{R}$ with coefficients in $\{0,1\}$. Define the functions*

$$\textsc{Compress}_q(x,d) := \lceil (2^d/q) \cdot x \rfloor \bmod^{(+)} 2^d\,,$$
$$\textsc{Decompress}_q(x,d) := \lceil (q/2^d) \cdot x \rfloor,$$

let χ a centered binomial distribution with support $\{-\eta,\ldots,\eta\}$, and let χ_n be the distribution of polynomials of degree n with entries independently sampled from χ. Define the public-key encryption scheme KYBER.CPA = *(*KYBER.CPA.GEN*,* KYBER.CPA.ENC*,* KYBER.CPA.DEC*) as follows:*

```
1 $(\rho,\sigma) \xleftarrow{\$} \{0,1\}^{256} \times \{0,1\}^{256}$ ;
2 $\vec{A} \xleftarrow{\rho} \mathcal{R}_q^{k\times k}$ ;
3 $(\vec{s},\vec{e}) \xleftarrow{\sigma} \chi_n^k \times \chi_n^k$ ;
4 $\vec{t} \leftarrow \textsc{Compress}_q(\vec{A}\vec{s} + \vec{e}, d_t)$ ;
5 return $pk_{CPA} := (\vec{t},\rho)$, $sk_{CPA} := \vec{s}$ ;
```

Algorithm 4: KYBER.CPA.GEN.

```
  Input: $pk_{CPA} = (\vec{t},\rho)$
  Input: $m \in \mathcal{M}$
  Input: $r \xleftarrow{\$} \{0,1\}^{256}$
1 $\vec{t} \leftarrow \textsc{Decompress}_q(\vec{t}, d_t)$ ;
2 $\vec{A} \xleftarrow{\rho} \mathcal{R}_q^{k\times k}$ ;
3 $(\vec{r},\vec{e}_1,e_2) \xleftarrow{r} \chi_n^k \times \chi_n^k \times \chi_n$ ;
4 $\vec{u} \leftarrow \textsc{Compress}_q(\vec{A}^T\vec{r} + \vec{e}_1, d_u)$ ;
5 $v \leftarrow \textsc{Compress}_q(\langle \vec{t},\vec{r}\rangle + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$ ;
6 return $c := (\vec{u}, v)$ ;
```

Algorithm 5: KYBER.CPA.ENC.

In Kyber, the parameters that define the base ring $\mathcal{R}_q$ are fixed at $n = 256$ and $q = 7681$. The parameters that define key and ciphertext compression are also fixed and set to $d_u = 11, d_v = 3$

```
    Input: $sk_{CPA} = \vec{s}$
    Input: $c = (\vec{u}, v)$
1 $\vec{u} \leftarrow$ DECOMPRESS$_q(\vec{u}, d_u)$ ;
2 $v \leftarrow$ DECOMPRESS$_q(v, d_v)$ ;
3 return COMPRESS$_q(v - \langle \vec{s}, \vec{u} \rangle, 1)$ ;
```

Algorithm 6: KYBER.CPA.DEC.

Parameter Set	n	q	k	η	q. bit-sec.	NIST cat.	failure	$\|pk\|$	$\|sk\|$	—$ctxt$—
Kyber512	256	7681	2	5	102	1	2^{-145}	736	1632	800
Kyber768	256	7681	3	4	161	3	2^{-142}	1088	2400	1152
Kyber1024	256	7681	4	3	218	5	2^{-169}	1440	3168	1504

Table 6.3: Parameters proposed for instantiating Kyber in the submission to the NIST process.

and $d_t = 11$. The three different security levels are obtained by different choices of k and η. All relevant Kyber parameters are summarized in Table 6.3.
The performance of an implementation of Kyber depends highly on the speed of the polynomial multiplication algorithm and the performance of the PRNG instantiations as a large number of pseudo random data is required when generating $\vec{A} \xleftarrow{\rho} \mathcal{R}_q^{k \times k}$ or when sampling noise from χ_n^k. Regarding operation in $\mathcal{R}_q$, KYBER.CPA.GEN needs k^2 multiplications and $(k-1)k$ additions. For encryption as defined in KYBER.CPA.ENC, k^2 multiplications and $(k-1)k$ additions as well as k multiplications and $k-1$ additions are needed. The decryption routine KYBER.CPA.DEC can be implemented with k multiplications and $k-1$ additions. Note that Kyber specifies a Number Theoretic Transform (NTT). The NTT allows to implement a fast polynomial multiplication by computing $c = \mathrm{NTT}^{-1}(\mathrm{NTT}(a) \circ \mathrm{NTT}(b))$ for $a, b, c \in R_q$, where $\circ$ denotes coefficient-wise multiplication. Kyber exploits that the NTT is a one-to-one map and assumes that randomly sampled polynomials in $\vec{A}$ are already in the transformed domain. Thus, an implementation using a different multiplication algorithm than the NTT would have to apply an inverse transformation first and then use the polynomial multiplication algorithm of its choice to stay compatible with the original specification.
Given $G\colon \{0,1\}^* \to \{0,1\}^{2\times 256}$ and $H\colon \{0,1\}^* \to \{0,1\}^{256}$ two hash functions, Kyber is obtained from KYBER.CPA using a Fujisaki-Okamoto style transform from [91] as shown in Algorithms 7, 8, 9. In KYBER.DECAPS a re-encryption has to be computed whose result is compared to the received ciphertext. Thus Kyber specifies exactly how to generate the uniformly random matrix $\vec{A}$ as well as polynomials from the error distribution χ_n from a seed[2]. For this the authors of Kyber have chosen different instantiations from the SHA3 family (SHAKE-128, SHAKE-256, SHA3-256 and SHA3-512).

```
1 $((\vec{t}, \rho), \vec{s}) \leftarrow$ KYBER.CPA.GEN() ;
2 $z \xleftarrow{\$} \{0,1\}^{256}$ ;
3 $h \leftarrow H((\vec{t}, \rho))$ ;
4 return $pk := (\vec{t}, \rho),\ sk := (\vec{s}, \vec{t}, \rho, h, z)$ ;
```

Algorithm 7: KYBER.GEN.

[2]This also applies to Newhope and FrodoKEM when used in the CCA setting but was omitted for simplicity in the respective descriptions.

```
  Input: $pk = (\vec{t}, \rho)$
1 $m \xleftarrow{\$} \{0,1\}^{256}$;
2 $m \leftarrow H(m)$;
3 $(\hat{K}, r) \leftarrow G(m, H(pk))$ ;
4 $(\vec{u}, v) \leftarrow$ KYBER.CPA.ENC$(pk, m; r)$;
5 $c \leftarrow (\vec{u}, v)$ ;
6 $K \leftarrow H(\hat{K}, H(c))$;
7 return $(c, K)$ ;
```

Algorithm 8: KYBER.ENCAPS.

```
  Input: $sk = (\vec{s}, \vec{t}, \rho, h, z)$
  Input: $c = (\vec{u}, v)$
1 $m' \leftarrow$ KYBER.CPA.DEC$(\vec{s}, (\vec{u}, v))$;
2 $(\hat{K}', r') \leftarrow G(m', h)$ ;
3 $(\vec{u}', v') \leftarrow$ KYBER.CPA.ENC$(pk, m'; r')$;
4 if $(\vec{u}', v') = (\vec{u}, v)$ then
5     $K \leftarrow H(\hat{K}', H(c))$;
6 else
7     $K \leftarrow H(z, H(c))$;
8 end
9 return $K$ ;
```

Algorithm 9: KYBER.DECAPS.

6.2.4 BIKE

Code-based cryptography relies on the hardness of decoding (random) linear codes. In communication, error correcting codes are used to detect or correct errors in the transmission of data over an unreliable channel. In 1978 McEliece showed that codes can also be used to realize asymmetric cryptography, or more specifically, public-key encryption and he proposed the McEliece cryptosystem. A variant of the McEliece scheme is the Niederreiter cryptosystem which was proposed in 1986. Both schemes are related to computationally hard problems in coding theory and are well understood and considered appropriately secure when the so-called binary Goppa codes are used. However, with binary Goppa codes the public keys get very large. As an example, the IND-CCA2 secure Classic McEliece [26] submission to the NIST process requires a 1.3 Megabyte public key for NIST category 5 security while the ciphertext itself is rather small with only 240 byte.

The original McEliece cryptosystem can also be instantiated with more structured codes to reduce these sizes. A promising family of codes are Quasi Cyclic Moderate Density Parity Check (QC-MDPC) codes. The submission BIKE [16] is an example of a suite of CPA-secure KEMs with various trade-offs with regard to security and performance. However, BIKE only works with ephemeral keys as it is currently not clear how to protect the QC-MPDC decoder required in decryption against chosen ciphertext attacks [86]. The BIKE suite consists of three variants that can be parametrized to achieve NIST category 1,3, or 5. The BIKE-1 variant supports fast key generation without inversion but consequently needs larger public keys (2 blocks). In BIKE-2 the Niederreiter approach and framework is used and key generation is more expensive but smaller public keys can be obtained. BIKE-3 works on a so-called noisy syndrome when performing decapsulation and features similar sizes for public keys than BIKE-1.

Let r be a prime, d_v be an odd integer, t an integer, all depending on the target quantum security level λ, such that $(X^r - 1)/(X - 1)$ is irreducible over $\mathbb{F}_2$. Let $\boldsymbol{K} : \{0,1\}^n \to \{0,1\}^{l_K}$ be a hash function. And let $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$.

```
1 $h_0, h_1 \xleftarrow{\$} \mathcal{R}$ of odd weight $|h_0| = |h_1| = \frac{w}{2}$ ;
2 $g \xleftarrow{\$} \mathcal{R}$ of odd weight $|g| \approx \frac{r}{2}$ ;
3 $(f_0, f_1) \leftarrow (gh_1, gh_0)$ ;
4 return $sk := (h_0, h_1), pk := (f_0, f_1)$ ;
```

Algorithm 10: BIKE-1.GEN.

```
  Input: $pk = (f_0, f_1)$
1 Sample $(e_0, e_1) \in \mathcal{R}^2$ with $|e_0| + |e_1| = t$;
2 $m \xrightarrow{\$} \mathcal{R}$ ;
3 $c = (c_0, c_1) \to (mf_0 + e_0, mf_1 + e_1)$ ;
4 $K \to \boldsymbol{K}(e_0, e_1)$ ;
5 return Ecapsulated key $K$, ciphertext $c$ ;
```

Algorithm 11: BIKE-1.ENCAPS.

```
  Input: $sk = (h_0, h_1)$, ciphertext $c$
1 $s \to c_0h_0 + c_1h_1$ ;
2 Try to decode $s$ and recover to error verctor $(e'_0, e'_1)$;
3 If $|(e'_0, e'_1)| \neq t$ or decoding fails, return $\perp$ and halt;
4 $K \to \boldsymbol{K}(e'_0, e'_1)$ ;
5 return $K$ ;
```

Algorithm 12: BIKE-1.DECAPS.

6.2.5 Additional Primitives

In the previous sections we have exemplarily described candidate schemes based on RLWE (NewHope), MLWE (Kyber), LWE (Frodo), or QC-MDPC (BIKE). However, also other submissions to the NIST process are very promising.

One example of an efficient post-quantum lattice-based cryptosystem is NTRU [90], which has already been around for nearly 20 years. It supports fast encryption and decryption operations and leads to relatively short public keys and ciphertest that are comparable in size to RLWE or MLWE-based schemes. NTRU variants submitted to the NIST competition are NTRUEncrypt [172], NTRU-HRSS-KEM [141], as well as NTRU Prime [27] which claims to use a particularly safe instantiation of the underlying NTRU problem. Another interesting scheme to consider is the CPA or CCA-secure ThreeBears [87] cryptosystem. It is based on the integer module learning with errors (I-MLWE) problem that can be considered to be a variant of RLWE. From an implementation perspective, ThreeBears operates on large integers modulo a Mersenne prime and might thus also be a candidate where existing RSA big-number engines could be reused to accelerate computations. The CPA and CCA-secure SABER [63] PKE is similar to Kyber but uses the Module Learning With Rounding problem (Mod-LWR). It deviates from the popular approach of choosing parameters or designing the algorithm in a way that the properties of the NTT can directly be exploited. In SABER the modulus is a power of two and due to the usage of the Mod-LWR problem

no sampling of MLWE/RLWE error vectors is required. This significantly reduces the required pseudo random data. The secret information in a public key or ciphertext is instead disguised by rounding.
For code-based cryptography we discard schemes based on binary Goppa codes, e.g., Classic McEliece [26] or NTS-KEM [6] due to the size of their public keys. Proposals besides BIKE that can be considered as efficient are schemes like Ouroboros-R [114] based on (quasi-cyclic) Low Rank Parity Check (LRPC) codes that support public, private, and ciphertext sizes between one and two kilobytes. Another code-based candidate is LEDAkem [23] based on Quasi-Cyclic Low Density Parity Check (QC-LDPC), which features public key of sizes ranging from 3,480 bytes (NIST cat. 1) to 22,704 bytes (NIST cat. 5).
The Supersingular Isogeny Key Encapsulation (SIKE) [97] is the only submission to the NIST process based on supersingular isogeny cryptography. SIKE realizes Supersingular isogeny Diffie–Hellman key exchange (SIDH) with extremely small public keys (e.g., 564 bytes for NIST category 3 SIKEp751). However, the performance of SIKE is a big disadvantage with reported 289 million cycles for key generation, 468 million cycles for encapsulation and 503 million cycles for decapsulation on an Intel Core i7-6700. On embedded devices sufficient performance can only be expected with the usage of specialized hardware accelerators. Moreover, the underlying problem of SIKE is rather new and requires more analysis.

6.3 Candidates Comparison

In Table 6.4 we compare the underlying security assumption, security level, public-key, secret key, and ciphertext sizes of the candidate schemes discussed in the previous section. With a 736-byte public key the Kyber512 instance provides the smallest public key size while Frodo-640 is an order of magnitude larger with 9616 bytes due to the usage of the LWE assumption. In terms of public-key and ciphertext size Kyber is consistently smaller than NewHope for a comparable security level. The small size of the secret key of the CPA-secure NewHope instantiations can be explained by the fact that no re-encryption is required and thus the public key does not have to be included into the secret key.
In Table 6.5 we provide cycle counts for implementations of various candidate schemes on a Cortex-M microcontroller reported by the mupq project [164]. The cycle counts were measured on a STM32F4 Discovery board featuring an ARM Cortex-M4 CPU, 1MB of Flash, and 192KB of RAM. With less than three million cycles for their operations Kyber and NewHope are faster than a state of the art Curve25519 ECC scalar multiplication which is reported to cost 3,589,850 cycles [69] on a similar target device. The performance of Frodo on the Cortex-M is significantly slower than Kyber and NewHope and each operation would roughly take one second assuming a standard clock frequency of 100 MHz.
In Table 6.6 we provide cycle counts of various candidate schemes on Intel/AMD CPUs using vector instructions. The results are obtained taken from the NIST respective submissions and the general picture is similar to the results displayed in Table 6.5.

6.4 Open Issues

This first deliverable highlighted some open issues that will be addressed in this project:

- It seems that a trade-off has to be made between efficiency and security when choosing the encryption scheme. Indeed, schemes based on weak hardness assumptions tend to

Table 6.4: Comparison of candidate schemes and their public key, secret key, and ciphertext sizes in bytes

Parameter Set	Assumption	NIST cat.	Security	Sec. Model	$\|pk\|$	$\|sk\|$	$\|c\|$
NEWHOPE512-CPA-KEM	RLWE	1	101	CPA	928	869	1088
NEWHOPE1024-CPA-KEM	RLWE	5	233	CCA	1824	1792	2176
NEWHOPE512-CCA-KEM	RLWE	1	101	CPA	928	1888	1120
NEWHOPE1024-CCA-KEM	RLWE	5	233	CCA	1824	3680	2208
Kyber512	MLWE	1	102	CCA	736	1632	800
Kyber768	MLWE	3	161	CCA	1088	2400	1152
Kyber1024	MLWE	5	218	CCA	1440	3168	1504
Frodo-640	LWE	1	103	CCA	9616	19872	9736
Frodo-976	LWE	3	150	CCA	15632	31272	15768
BIKE-1-2542	QC-MDPC	1	-	CPA	2541	267	2541
BIKE-1-4964	QC-MDPC	3	-	CPA	5474	287	5474
BIKE-1-8188	QC-MDPC	5	-	CPA	8188	548	8188
NTRU-HRSS-KEM	NTRU	5	-	CCA	1138	1418	1278
sntrup4591761 (NTRU Prime)	NTRU	5	-	CCA	1218	1600	1047
ntrulpr4591761 (NTRU Prime)	NTRU	5	-	CCA	1047	1238	1175
ntru-kem-443 (NTRUEncrypt)	NTRU	1	84	CCA	611	701	611
ntru-kem-743 (NTRUEncrypt)	NTRU	5	159	CCA	1023	1173	1023
ntru-kem-1024 (NTRUEncrypt)	NTRU	5	198	CCA	4097	8194	4097
SIKEp751	Isogeny	3	191	CCA	564	24	596

Table 6.5: Cycle counts for implementations of various candidate schemes on a Cortex-M microcontroller

Parameter Set	key generation	encapsulation	decapsulation
NEWHOPE1024-CCA-KEM	1,502,435	2,370,157	2,517,215
Kyber768	1,200,351	1,497,789	1,526,564
Frodo-640	94,191,951	111,688,861	112,156,317
NTRU-HRSS-KEM	197,262,297	5,166,153	15,069,480
SABER	7,122,695	9,470,634	12,303,775
SIKEp751	3,508,587,555	5,685,591,898	6,109,763,845

Table 6.6: Cycle counts for implementations of various candidate schemes on CPUs.

Parameter Set	Optimization	key generation	encapsulation	decapsulation
NEWHOPE512-CPA-KEM	AVX	56,236	85,144	19,472
NEWHOPE1024-CPA-KEM	AVX	107,032	163,332	35,716
NEWHOPE512-CCA-KEM	AVX	68,080	109,836	114,176
NEWHOPE1024-CCA-KEM	AVX	129,670	210,092	220,864
Kyber512	AVX	55,160	75,680	74,428
Kyber768	AVX	85,472	112,660	108,904
Kyber1024	AVX	121,056	157,964	154,952
Frodo-640	AVX	1,288,000	1,834,000	1,837,000
Frodo-976	AVX	2,677,000	3,577,000	3,580,000
NTRU-HRSS-KEM	AVX	294,874	38,456	68,458
sntrup4591761 (NTRU Prime)	?	over 6,000,000	59456	97684
ntru-kem-443	?	1,144,000	213,200	283,400
ntru-kem-743	?	2,644,200	364,000	546,000
ntru-kem-1024	?	113,100,000	169,000,000	299,000,000
SABER	?	216,597	267,841	318,785

have rather large ciphertexts and public keys, while very efficient schemes usually need stronger assumptions. For example, when analyzing lattice-based schemes, on one hand schemes based on Ring-LWE have shorter keys and ciphertexts than schemes based on LWE (e.g., compare Frodo with New Hope), while on the other hand LWE is a weaker hardness assumption than Ring-LWE (even if, until now, no attacks exploiting the additional algebraic structure of Ring-LWE are known). The same observation can be done for codes.

- Like for signature schemes, also for encryption schemes there is ongoing cryptanalysis motivated by the NIST call. In particular, from a first analysis it seems that many schemes might be vulnerable to side-channel attacks. Overall, for code-based schemes it seems easier to prevent leakage as there are already standardized techniques that can be applied, even if these schemes are usually non-constant time and seems to be quite hard to secure. The effectiveness of leake-mitigation techniques for lattice-based schemes is still unclear.

- When choosing a scheme among the different candidates, memory requirements have to be taken into account too. From this first analysis, it seems that schemes based on codes guarantee small ciphertexts but rather large public keys, while in general lattice-based constructions guarantee smaller public keys.

- In this overview we focused mainly on schemes based on hardness assumptions from codes or lattices. Other schemes have been proposed that are based on assumptions that are either new or not well studied. Hence, their quantum security is hard to judge, and will be probably assessed after the Analysis Phase of the NIST call.

Chapter 7

Privacy-Supporting Primitives

The TPM-based trusted computing technology, specified by the Trusted Computing Group (TCG), provides a balance between security and privacy. Security means that the TPM supports various security services, such as platform authentication, secure storage, remote attestation, and only authorised entities can access these services. Privacy means that when the TPM provides a service, the identifier of the TPM is not revealed to any unauthorised entity, and that for two services supported by the TPM, the connection between them (meaning that they are provided by the same TPM) is also not revealed to any unauthorised entity.

7.1 Privacy Requirements

When a TPM communicates with an entity through a network to provide a TPM service, for security reasons, the entity usually requires verifying that the TPM is a genuine TPM. The successful verification will provide some assurance to the entity that the TPM will follow the TPM specifications, as specified in the TCG standard and the international standard ISO/IEC 11889. Each genuine TPM can be identified by their cryptographic keys that are unique to the TPM.
However, from the TPM side, usually interested by the TPM owner or the owner of the platform where the TPM is embedded, one or more the following privacy requirements should be held in the service:

- Anonymity. A service provided by a TPM is anonymous, meaning that it is computationally infeasible for any unauthorised entity to tell which TPM has provided this service. In other words, the service does not reveal the identity or cryptographic key of the TPM.

- Linkability/Unlinkability. Two services provided by TPMs are linkable, meaning that an authorised entity can tell whether these two services have been provided by the same TPM or not. Two services are unlinkable, meaning that it is computationally infeasible for any unauthorised entity to tell whether these two services have been provided by the same TPM or different TPMs. Here we talk about at least one service is anonymous, otherwise finding they are linked or not linked is trivial.

- Deniability. A service provided by a TPM is deniable, meaning that the entity received this service cannot convince a third party that the TPM has provided such a service.

7.2 (Interactive and Non-Interactive) Zero-Knowledge Proofs

We need to consider both security and privacy requirements. Again assume that a TPM provides a security service to an entity where the entity wants to authenticate the TPM by checking the validation of the TPM's cryptographic key, and the service itself does not reveal the key obeying the privacy requirement. A balance between security and privacy requirements can be achieved by using a zero-knowlege proof. Such a proof can let the TPM convince the verifying entity that the TPM knows the key but the entity cannot distribute this knowledge to a third party. A "real" zero-knowledge proof without setup assumptions requires interaction between the TPM and the verifier. The reason is that if the TPM produces a single, one-shot piece of information that convinces the verifier, then this information in theory could be used to convince anybody else, therefore making the proof transferable without actual knowledge of the secret key. Transferability can be avoided only by using additional setup assumptions (such as the Common Reference String or Random Oracle models). Hence, interaction generally offers stronger security guarantees for zero-knowledge proofs.
However, interaction in zero-knowledge proof is expensive. Luckily, for most of the TPM applications interaction it is not necessary. The reason is that it is acceptable to let a third party know that the verifier has talked to a TPM as long as the TPM's identifier is not revealed (see the discussion in the next subsection). In that case, a non-interactive zero-knowledge proof can be used. The typical technique for constructing such a proof is using a digital signature, which can be used to convince a verifier that the TPM knows a private signing key and has used it to sign a given message, without revealing the signing key. A non-interactive zero-knowledge proof based on digital signatures is referred to as a signature-based proof of knowledge.

7.3 Anonymous Signatures

A digital signature scheme enables a private key holder to digitally sign a message. The corresponding public verification key can be used to verify the validity of the signature on the message. Anonymous digital signatures are a special type of digital signatures. In an anonymous digital signature scheme, given a digital signature, an unauthorised entity, including the verifier, cannot discover the signer's identifier. However, such a scheme still has the property that only a legitimate signer can generate a valid signature. For authorised entities involved in an anonymous signature mechanism, there are four different cases:

1. A scheme involving an authorised entity that is capable of identifying the signer of a signature;
2. A scheme involving an authorised entity that is only capable of linking two signatures created by the same signer without identifying the signer;
3. A scheme involving both of the authorised entities in Cases 1) and 2);
4. A scheme involving neither of the authorised entities in Cases 1) and 2).

One of the major differences between a conventional digital signature and an anonymous digital signature is in the nature of the public keys used to perform the signature verification. To verify a conventional digital signature, the verifier makes use of a single public verification key which is bound to the signer's identifier. To verify an anonymous digital signature, the verifier makes use of either a group public key or multiple public keys, which are not bound to an individual signer. In

the literature, an anonymous signature using a group public key is commonly known as a group signature [56], and an anonymous signature using multiple public keys is commonly known as a ring signature [136]. The anonymity strength (i.e. degree of anonymity) provided by a mechanism depends upon the size of the group and the number of public keys.
A TPM supports a special type of group signature, namely Direct Anonymous Attestation (DAA). A traditional group signature scheme involves the first authorised entity case, i.e., a group manager can identify the signer of a group signature. However, in the trusted computing environment, it is not easy to find such a super user. Instead, DAA supports the Cases 2) or 4) and allows users (either a signer or verifier or both) to make a choice between these two cases.

7.4 DAA

7.4.1 General Concept

A DAA scheme involves a set of issuers, signers, and verifiers. An issuer is in charge of verifying the legitimation of signers and of issuing a DAA credential to each signer. A signer can prove the possession of the credential to a verifier by providing a DAA signature. The verifier can verify the membership credential from the signature but he cannot learn the identity of the signer. The following two unique properties make DAA attractive in practice.

- The first one is that the signer role of DAA is split between two entities, a principal signer with limited computational and storage capability, that can be a TPM, and an assistant signer with more computational power but less security tolerance, e.g. an ordinary computer platform (namely the host with the TPM embedded in). The TPM is the real signer and holds the secret signing key, whereas the host helps the TPM to compute the signature under the credential, but is not allowed to learn the secret signing key and to forge such a signature without the involvement of a TPM.

- The second one is to provide different degrees of privacy. As mentioned before, a DAA scheme can be seen as a special group signature scheme without the feature of opening the identity of the signer from its signature by the issuer. Interactions in DAA signing and verification are anonymous, meaning that neither verifier, the issuer, or even both of them colluding together, can discover the identity of the signer from a DAA signature. Instead of full-traceability as held in group signatures [56], DAA has user-controlled traceability, meaning that the DAA signer is able to control whether or not a verifier can determine if any two signatures have been produced by the same signer. Moreover, the signer and verifier may negotiate as to whether or not the verifier is able to link different signatures signed by the signer.

7.4.2 Security Models

There are a number of DAA security models in the literature, including the simulation-based model [50], the game-based model [57] and the Universal Composability (UC) model [53, 51, 52]. In this section, we follow the security model for DAA given by Camenish et al. in [53]. In the UC DAA model, an environment ε should not be able to distinguish with non negligible probability between two worlds:

1. The real world, where each part in the DAA protocol Π executes its assigned part of the protocol. The network is controlled by an adversary $\mathcal{A}$ that communicates with ε.

2. The ideal world, in which all parties forward their inputs to a trusted third party, called the ideal functionality F^l_{daa}, which internally performs all the required tasks and creates the party's outputs.

A protocol Π is said to securely realize F^l_{daa} if for every adversary $\mathcal{A}$ performing an attack in the real world, there is an ideal world adversary $\mathcal{S}$ that performs the same attack in the ideal world. More precisely, given a protocol Π, an ideal functionality F^l_{daa} and an environment ε, we say that Π securely realises F^l_{daa} if the real world in which Π is used is as secure as the ideal world in which F^l_{daa} is used. In other words, for any adversary $\mathcal{A}$ in the real world, there exists a simulator $\mathcal{S}$ in the ideal world such that $(\varepsilon, F^l_{daa}, \mathcal{S})$ is indistinguishable from $(\varepsilon, \Pi, \mathcal{A})$.
In general the security properties that a DAA scheme should enjoy are the following:

- *Unforgeability* This property requires that the issuer is honest and should hold even if the host is corrupt. If all the TPMs are honest, then no adversary can output a signature on a message M with respect to a basename (bsn). On the other hand, if not all the TPMs are honest, say n TPMs are corrupt, the adversary can at most output n unlinkable signatures with respect to the same basename.

- *Anonymity*: This property requires that the entire platform $(\mathsf{tpm_i} + \mathsf{host_j})$ is honest and should hold even if the issuer is corrupt. Starting from two valid signatures with respect to two different basenames, the adversary cannot tell whether these signatures were produced by one or two different honest platforms.

- *Non-frameability*: This requires that the entire platform $(\mathsf{tpm_i} + \mathsf{host_j})$ is honest and should hold even if the issuer is corrupt. It ensures that no adversary can produce a signature that links to signatures generated by an honest platform.

In the existing DAA schemes supported by the TPM (either the TPM Version 1.2 or the TPM Version 2.0), privacy was built on the honesty of the entire platform, i.e., both the TPM and the host are supposed to be honest. In [51, 52] it is considered that the TPM may be corrupt and privacy must hold whenever the host is honest, regardless of the corruption state of the TPM. However, this strong privacy notion has not been adopted by the TCG. To make the DAA security model compatible to the TCG TPM specifications, we do not consider this case in this document. We now formally define the ideal functionality F^l_{daa} under the assumption of static corruption, i.e., the adversary decides beforehand which parties are corrupt and informs F^l_{daa} about them. F^l_{daa} has five interfaces (SETUP, JOIN, SIGN, VERIFY, LINK) described below. In the UC model, several sessions of the protocol are allowed to run at the same time and each session will be given a global identifier sid that consists of an issuer I and a unique string sid', i.e. $\mathsf{sid} = (\mathsf{sid}', \mathsf{I})$. We also define the JOIN and SIGN sub-sessions by jsid and ssid. F^l_{daa} is parameterized by a leakage function $l : \{0,1\}^* \rightarrow \{0,1\}^*$, which models the information leakage that occurs in the communication between a host $\mathsf{host_j}$ and a TPM $\mathsf{tpm_i}$. We also define the algorithms that will be used inside the functionality as follows:

- $\mathsf{Kgen}(1^\lambda)$: A probabilistic algorithm that takes a security parameter λ and generates keys gsk for honest TPMs.

- $\mathsf{sig}(gsk,\ \mu,\ \mathsf{bsn})$: A probabilistic algorithm used for honest TPMs. On input of a key gsk, a message μ and a basename bsn, it outputs a signature σ.

- $\mathsf{ver}(\sigma,\ \mu,\ \mathsf{bsn})$: A deterministic algorithm that is used in the VERIFY interface. On input of a signature σ, a message μ and a basename bsn, it outputs $f = 1$ if the signature is valid, $f = 0$ otherwise.

- link($\sigma_1,\ \mu_1,\ \sigma_2,\ \mu_2,\ \mathsf{bsn}$): A deterministic algorithm that will be used in the LINK interface. It outputs 1 if both σ_1 and σ_2 were generated by the same TPM, 0 otherwise.
- identify($gsk, \sigma,\ \mu,\ \mathsf{bsn}$): A deterministic algorithm that will be used to ensure consistency with the ideal functionality F^l_{daa}'s internal records. It outputs 1 if a key gsk was used to produce a signature σ, 0 otherwise.

We now define useful functions to check whether or not a TPM key is consistent with the internal records of F^l_{daa}. We distinguish between the two cases whether a TPM is honest or corrupt as follows:

1. CkeckGskHonest(gsk): If the $\mathtt{tpm}_\mathsf{i}$ is honest, and no signatures in Signed or valid signatures in VerResults identify to be signed by gsk, then gsk is eligible and the function returns 1, otherwise it returns 0.
2. CkeckGskCorrupt(gsk): If the $\mathtt{tpm}_\mathsf{i}$ is corrupt and $\nexists gsk' \neq gsk$ and $(\mu,\ \sigma,\ \mathsf{bsn})$ such that both keys identify to be the owners of the same signature σ, then gsk is eligible and the function returns 1, otherwise it returns 0.

We refer to Appendix C for details on the interfaces of the ideal functionality F^l_{daa}.

7.4.3 The Existing Schemes

There are two lattice-based DAA schemes in the literature [101, 25]. Security of these two schemes is based on the Ring-SIS problem and Ring-LWE problem.
Analogously to other DAA schemes, such as the RSA-based DAA scheme in TPM version 1.2 and the ECC-based DAA schemes in TPM version 2.0, a lattice-based DAA scheme consists of five algorithms/protocols: SETUP, JOIN, SIGN, VERIFY and LINK. In the SETUP algorithm, the issuer creates a public and private key pair, which will be used for the issuer to create a DAA credential in the JOIN protocol. A DAA credential is a signature signed by the issuer. For a lattice-based DAA scheme, a DAA credential is a lattice-based signature. The JOIN protocol is run between the issuer and a signer, which is also referred to as a group member since DAA is a group-oriented signature. A DAA signer consists of a TPM and its host. During the JOIN protocol, the TPM creates a DAA private key and the corresponding public key. The issuer creates a DAA credential, which likes a certificate to the public key. The reason that this is called a credential rather than a certificate is that this certificate is not used to validate the public key by any DAA verifiers. At the end of the JOIN protocol, the host records the credential. The SIGN protocol is run between the TPM and its host, the host randomises the DAA credential and the TPM creates a signature that matches with the randomised credential. The TPM signature together with the randomised credential becomes a DAA signature, which is also a lattice-based signature. Like in a group signature scheme, a DAA verifier uses the issuer public key in the VERIFY algorithm to verify a DAA signature. Unlike a group signature, if two DAA signatures make use of the same base name, a DAA verifier in the LINK algorithm verifies whether these two signatures were signed by the same TPM or not.

The L-DAA Scheme [101]

The DAA credential in this scheme is a modification of the Boyen signature [48]. To analyse security of the scheme the following standard functionalities are required, as specified in [53].

- F_{ca} is a common certificate authority functionality that is available to all parties.

- $F^{\mathcal{D}}_{crs}$ is a common reference string functionality that provides participants with all system parameters.
- F_{auth^*} is a special authenticated communication functionality that provides an authenticated channel between the issuer and the TPM via the host.
- F^{l}_{smt} is a secure message transmission functionality that provides an authenticated and encrypted communication between the TPM and the host.

The L-DAA scheme includes the *SETUP*, *JOIN*, *SIGN*, *VERIFY*, and *LINK* processes as follows:

SETUP:
F_{crs} creates the system parameters: $\mathsf{sp} = (\lambda, q, n, m, \mathcal{R}_q, c, \beta, \beta', \ell)$, where λ and c are positive integer security parameters, β and β' are positive real numbers such that $\beta, \beta' < q$, and ℓ is the length of a message to be signed in the Boyen signature.
Upon input (SETUP, sid), where sid is a unique session identifier, the issuer first checks that $\mathsf{sid} = (\mathsf{I}, \mathsf{sid}')$ for some sid′, then creates its key pair. Issuer's public key is $\mathsf{pp} = (\mathsf{sp}, \hat{A}_t, \hat{A}_I, \hat{A}_0, \hat{A}_1, ..., \hat{A}_\ell, \mathbf{u}, \mathcal{H}, \mathcal{H}_0)$, where $\hat{A}_t, \hat{A}_I, \hat{A}_i (i = 0, 1, ..., \ell) \in \mathcal{R}^m_q$, $\mathbf{u} \in \mathcal{R}_q$, $\mathcal{H} : \{0,1\}^* \to \mathcal{R}_q$, and $\mathcal{H}_0 : \{0,1\}^* \to \{1,2,3\}^c$.
Issuer's private key is $\hat{T}_I$, which is the trapdoor of $\hat{A}_I$ and $\|\hat{T}_I\|_\infty \le \omega$, for some small real number ω .
The issuer initializes the list of joining members: Memebers $\leftarrow \emptyset$. The issuer proves that his secret key is well formed by generating a proof of knowledge π_I, and registers the key $(\hat{T}_I, \pi_I)$ with F_{ca} and outputs (SETUPDONE, sid).

JOIN: The Join process is a protocol running between the Issuer I and a platform, consisting of a TPM $\mathsf{tpm_i}$ and a Host $\mathsf{host_j}$ (with an identifier id). More than one Join session may run in parallel. A unique sub-session identifier jsid is used and this value is given to all parties.
The issuer I checks that the TPM-host is qualified to make the trusted computing attestation service, then issues a credential enabling the platform to create attestations. Via the unique session identifier jsid, the issuer can differentiate between various Join sessions that are executed simultaneously. A Join session works in two phases, Join request and Join proceed, as follows:

Join Request: On input query (JOIN, sid, jsid, $\mathsf{tpm_i}$), the host $\mathsf{host_j}$ forwards (JOIN, sid, jsid) to I, who replies by sending (sid, jsid, ρ, bsn_I) back to $\mathsf{host_j}$, where ρ is a uniform random nonce $\rho \xleftarrow{\$} \{0,1\}^\lambda$, and bsn_I is the Issuer's base name. This message is then forwarded to $\mathsf{tpm_i}$. The TPM proceeds as follows:

1. It checks that no such entry exists in its storage.
2. It samples a private key: $\hat{X}_t = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \xleftarrow{\$} \mathcal{R}^m_q$ with the condition $\|\hat{X}_t\|_\infty \le \beta$, and stores its key as (sid, $\mathsf{host_j}$, $\hat{X}_t$, id).
3. It computes the corresponding public key $\mathbf{u}_t = \hat{A}_t \cdot \hat{X}_t \mod q$, a link token $\mathsf{nym}_I = \mathcal{H}(\mathsf{bsn}_I) \cdot \mathbf{x}_1 + \mathbf{e}_I \mod q$ for some error $\mathbf{e}_I \leftarrow \mathcal{D}_{\mathbb{Z}^n, s'}$ such that $\|\mathbf{e}_I\|_\infty < \beta'$, and generates a signature based proof:

$$\begin{gathered}\pi_{\mathbf{u}_t} = \mathsf{SPK}\Big\{\text{public} := \{\mathsf{sp},\ \hat{A}_t,\ \mathbf{u}_t,\ \mathsf{bsn}_I,\ \mathsf{nym}_I\},\\ \text{witness} := \{\hat{X}_t = (\mathbf{x}_1, \ldots, \mathbf{x}_m),\ \mathbf{e}_I\} :\\ \mathbf{u}_t = \hat{A}_t \cdot \hat{X}_t \mod q \ \wedge\ \|\hat{X}_t\|_\infty \le \beta \ \wedge\ \mathsf{nym}_I = \mathcal{H}(\mathsf{bsn}_I) \cdot \mathbf{x}_1 + \mathbf{e}_I\\ \mod q \ \wedge\ \|\mathbf{e}_I\|_\infty \le \beta'\Big\}(\rho).\end{gathered}$$

4. It sends $(\mathsf{nym}_I,\ \mathsf{id},\ \mathbf{u_t},\ \pi_{\mathbf{u_t}})$ to the issuer I via the host by means of F_{auth^*}, i.e., it gives F_{auth^*} an input (SEND, $(\mathsf{nym}_I,\ \pi_{\mathbf{u}_t}), (\mathsf{sid},\ \mathsf{tpm_i},\ I),\ \mathsf{jsid},\ \mathsf{host_j}$).

The host, upon receiving (APPEND, $(\mathsf{nym}_I,\ \pi_{\mathbf{u}_t}), (\mathsf{sid},\ \mathsf{tpm_i}, I)$) from F_{auth^*}, forwards it to I by sending (APPEND, $(\mathsf{nym}_I,\ \pi_{\mathbf{u}_t}), (\mathsf{sid},\ \mathsf{tpm_i},\ I)$) to F_{auth^*} and keeps the state $(\mathsf{jsid},\ \mathbf{u_t},\ \mathsf{id})$. I upon receiving (SENT, $(\mathsf{nym}_I,\ \pi_{\mathbf{u}_t}), (\mathsf{sid},\ \mathsf{tpm_i},\ I),\ \mathsf{jsid},\ \mathsf{host_j}$) from F_{auth^*}, verifies the proof $\pi_{\mathbf{u_t}}$ to make sure that $\mathsf{tpm_i} \notin$ Members. I stores $(\mathsf{jsid},\ \mathsf{nym}_I,\ \pi_{\mathbf{u}_t},\ \mathsf{id}, \mathsf{tpm_i},\ \mathsf{host_j})$, and generates the message (JOINPROCEED, $\mathsf{sid},\ \mathsf{jsid},\ \mathsf{id},\ \pi_{\mathbf{u}_t}$).

Join Proceed: If the platform chooses to proceed with the Join session, the message (JOINPROCEED, $\mathsf{sid},\ \mathsf{jsid}$) is sent to the issuer, who performs as follows:

1. It checks the record $(\mathsf{jsid},\ \mathsf{nym}_I,\ \mathsf{id},\ \mathsf{tpm_i},\ \mathsf{host_j},\ \pi_{\mathbf{u}_t})$. For all nym'_I from the previous Join records, the issuer checks whether $\|\mathsf{nym}_I - \mathsf{nym}'_I\|_\infty \leq 2\beta'$ holds; if yes, the issuer treats this session as a rerun of the Join process; otherwise the issuer adds $\mathsf{tpm_i}$ to Members and goes to Step 2. If this is a rerun, the issuer will further check if $\mathbf{u}_t = \mathbf{u}'_t$; if not the issuer will abort; otherwise the issuer will jump to Step 4 returning $\hat{X}_h = \hat{X}'_h$. Note that this double check will make sure that any two DAA keys will not include the same $\mathbf{x}_1$ value.

2. It calculates the vector of polynomials $\hat{A}_h = [\hat{A}_I|\hat{A}_0 + \sum_{i=1}^{\ell} \mathsf{id}_i \cdot \hat{A}_i] \in \mathcal{R}_q^{2m}$.

3. It samples, using the issuer's private key $\hat{T}_I$, a preimage $\hat{X}_h = (\mathbf{x}_{m+1}, \ldots, \mathbf{x}_{3m})$ of $\mathbf{u} - \mathbf{u}_t$ such that: $\hat{A}_h \cdot \hat{X}_h = \mathbf{u}_h = \mathbf{u} - \mathbf{u}_t \mod q$ and $\|\hat{X}_h\|_\infty \leq \beta$.

4. It sends $(\mathsf{sid},\ \mathsf{jsid},\ \hat{X}_h)$ to $\mathsf{host_j}$ via F_{auth^*}.

When the host recieves the message $(\mathsf{sid},\ \mathsf{jsid},\ \hat{X}_h)$, it checks that the equations $\hat{A}_h \cdot \hat{X}_h = \mathbf{u}_h \mod q$ and $\mathbf{u} = \mathbf{u}_t + \mathbf{u}_h$ are satisfied with $\|\hat{X}_h\|_\infty \leq \beta$. If the checks are correct, then $\mathsf{host_j}$ stores $(\mathsf{sid},\ \mathsf{tpm_i},\ \mathsf{id},\ \hat{X}_h,\ \mathbf{u}_t)$ and outputs (JOINED, $\mathsf{sid},\ \mathsf{jsid}$).

SIGN: After obtaining the credential from the Join process, $\mathsf{tpm_i}$ and $\mathsf{host_j}$ can sign a message μ with respect to a basename bsn. We use a unique sub-session identifier ssid to allow multiple Sign sessions. Each session has two phases, Sign request and Sign proceed.
Sign request: Upon input (SIGN, $\mathsf{sid},\ \mathsf{ssid},\ \mathsf{tpm_i},\ \mathsf{bsn},\ \mu$), $\mathsf{host_j}$ looks up the record $(\mathsf{sid},\ \mathsf{tpm_i},\ \mathsf{id},\ \mathbf{u}_t, \hat{X}_h)$, and sends the message $(\mathsf{sid},\ \mathsf{ssid},\ \mathsf{bsn},\ \mu)$ to $\mathsf{tpm_i}$. The TPM then does the following:

1. It asks $\mathsf{host_j}$ for a permission to proceed.

2. It makes sure to have a Join record $(\mathsf{sid},\ \mathsf{id}, \hat{X}_t,\ \mathsf{host_j})$.

3. It generates a sign entry $(\mathsf{sid},\ \mathsf{ssid},\ \mathsf{bsn},\ \mu)$ in its record.

4. Finally it outputs (SIGNPROCEED, $\mathsf{sid},\ \mathsf{ssid},\ \mathsf{bsn},\ \mu$).

Sign Proceed: When $\mathsf{tpm_i}$ gets permission to proceed for ssid, the TPM proceeds as follows:

1. It retrieves the records $(\mathsf{sid},\ \mathsf{id},\ \mathsf{host_j},\ \pi_{\mathbf{u}_t})$ and $(\mathsf{sid},\ \mathsf{ssid},\ \mathsf{bsn},\ \mu)$.

2. Depending on the input bsn, there are two cases: If $\mathsf{bsn} \neq \perp$, the tpm computes the tag $\mathsf{nym} = \mathcal{H}(\mathsf{bsn}) \cdot \mathbf{x}_1 + \mathbf{e} \mod q$, for an error term $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n, s'}$ such that $\|\mathbf{e}\|_\infty < \beta'$ and generates a commitment:

$$\theta_t = \mathsf{COM}\Big\{\mathsf{public} := \{\mathsf{sp},\ \hat{A}_t,\ \mathsf{nym},\ \mathsf{bsn},\ \mathcal{H},\ \mathbf{u}_t\},\\ \mathsf{witness} := \{\hat{X}_t = (\mathbf{x}_1, \ldots, \mathbf{x}_m), \mathbf{e}\} :\\ \{\hat{A}_t \cdot \hat{X}_t = \mathbf{u}_t\ \wedge\ \|\hat{X}_t\|_\infty \le \beta\}\ \wedge\ \mathsf{nym} = \mathcal{H}(\mathsf{bsn}) \cdot \mathbf{x}_1 + \mathbf{e}\ \wedge\ \|\mathbf{e}\|_\infty \le \beta'\Big\}.$$

If $\mathsf{bsn}{=}\perp$, then $\mathsf{tpm_i}$ samples a random value $\mathsf{bsn} \leftarrow \{0,1\}^\lambda$, and then follows the previous case.

3. $\mathsf{tpm_i}$ sends $(\mathsf{sid},\ \mathsf{ssid},\ \theta_t,\ \mu)$ to $\mathsf{host_j}$.

4. When $\mathsf{host_j}$ recieves the message $(\mathsf{sid},\ \mathsf{ssid},\ \theta_t,\ \mu)$, it checks that the proof θ_t is valid, and subsequently generates a commitment again:

$$\theta_h = \mathsf{COM}\Big\{\mathsf{public} := \{\mathsf{sp},\ \hat{A}_h,\ \mathbf{u}_h,\ \mu,\ \theta_t\},\\ \mathsf{witness} := \{\hat{X}_h = (\mathbf{x}_{m+1}, \ldots, \mathbf{x}_{3m}),\ \mathsf{id}\} :\\ \{\hat{A}_h \cdot \hat{X}_h = \mathbf{u}_h\ \wedge\ \|\hat{X}_h\|_\infty \le \beta\}\Big\}.$$

The combination of these two commitments θ_t and θ_h follows the additional homomorphic property of the commitment scheme.

5. The TPM and Host run the standard Fiat-Shamir transformation, and the result is a signature based proof (signed on the message μ):

$$\pi = \mathsf{SPK}\Big\{\mathsf{public} := \{\mathsf{pp},\ \mathsf{nym},\ \mathsf{bsn}\},\\ \mathsf{witness} := \{\hat{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{3m}),\ \mathsf{id},\ \mathbf{e}\} :\\ [\hat{A}_t|\hat{A}_h] \cdot \hat{X} = \mathbf{u}\ \wedge\ \|\hat{X}\|_\infty \le \beta\ \wedge\ \mathsf{nym} = \mathcal{H}(\mathsf{bsn}) \cdot \mathbf{x}_1 + \mathbf{e} \mod q\ \wedge\ \|\mathbf{e}\|_\infty \le \beta'\Big\}(\mu).$$

For the details on how to compute θ_t, θ_h and π, we refer it to [101].

6. $\mathsf{host_j}$ outputs the L-DAA signature $\sigma = (\mathsf{nym}, \mathsf{bsn}, \pi)$.

VERIFY: The verify algorithm allows anyone to check whether a signature σ on a message μ with respect to a basename bsn is valid. Let RL denotes a revocation list with all the rogue TPM's secret keys. Upon input (VERIFY, $\mathsf{sid},\ \mathsf{bsn},\ \sigma,\ \mu,\ RL$), the verifier proceeds as follows:

1. It parses σ as $(\mathsf{nym},\ \mathsf{bsn},\ \pi)$, and checks SPK on π with respect to $\mathsf{bsn},\ \mathsf{nym},\ \mu$ and $\mathbf{u}$, then verifies the statement:

$$[\hat{A}_t|\hat{A}_h] \cdot \hat{X} = \mathbf{u}\ \wedge\ \|\hat{X}\|_\infty \le \beta\ \wedge\ \mathsf{nym} = \mathcal{H}(\mathsf{bsn}) \cdot \mathbf{x}_1 + \mathbf{e} \mod q\ \wedge\ \|\mathbf{e}\|_\infty \le \beta'.$$

2. It checks that the secret key $\hat{X}_t$ that was used to generate nym, doesn't belong to the revocation list RL. This is done by checking whether the following equation holds:

$$\forall \mathbf{x}_1 \in RL, \|\mathcal{H}(\mathsf{bsn}) \cdot \mathbf{x}_1 - \mathsf{nym}\|_\infty \le \beta'.$$

3. If all checks passed, the verifier outputs (VERIFIED, $\mathsf{ssid},\ 1$), and (VERIFIED, $\mathsf{ssid},\ 0$) otherwise.

LINK: The link algorithm allows anyone to check whether two signatures $(\sigma,\ \mu)$ and $(\sigma',\ \mu')$ that were generated for the same basename bsn stem from the same TPM. Upon input (LINK, $\mathsf{sid},\ \sigma,\ \mu,\ \sigma',\ \mu',\ \mathsf{bsn}$) the verifier follows the following steps:

1. Starting from $\sigma = (\mathsf{nym},\ \mathsf{bsn},\ \pi)$ and $\sigma' = (\mathsf{nym}',\ \mathsf{bsn},\ \pi')$, the verifier verifies σ and σ' individually.
2. If any of the signatures are invalid, the verifier outputs $\perp$.
3. Otherwise if $\|\mathsf{nym} - \mathsf{nym}'\|_\infty < 2\beta'$, the verifier outputs 1 (linked); otherwise 0 (not linked).

The El Bansarkhani et al Scheme [25]

This DAA scheme works as follows: The issuer's public key consists of $\ell + 2$ vectors in $\mathcal{R}_q^m$, namely $\hat{A}_I,\ \hat{A}_i$ for $i = 0, 1, \cdots \ell$, and 2 polynomials $\mathbf{u}$ and $\mathbf{b} \in \mathcal{R}_q$. The TPM generates a small secret $\hat{Z}_1 \in \mathcal{R}_q^{2m+1}$ such that $[\mathbf{b}|\hat{A}_{\mathsf{id}}][\hat{Z}_1] = \tilde{\mathbf{u}} \mod q$. The TPM sends $\tilde{\mathbf{u}}$ together with a proof of knowledge π_1 to the issuer, who registers both $\tilde{\mathbf{u}}$ and the corresponding TPM, and samples (using his secret key) a small credential $\hat{Z}_2$ such that $\hat{A}_{\mathsf{id}}\hat{Z}_2 = \mathbf{u} - \tilde{\mathbf{u}} \mod q$. The TPM and the host together combine their secret data to obtain a valid credential satisfying $\mathbf{u} = [\mathbf{b}|\hat{A}_{\mathsf{id}}][\hat{Z}_1 + (\mathbf{0}|\hat{Z}_2)]$. To create a signature, the TPM samples a small random vector $\hat{T} \in \mathcal{R}_q^{2m}$, such that $\hat{T}\hat{A}_{\mathsf{id}} \mod q$ is uniform, and shares it with the host in order to randomize the signature. The TPM and the host generate π_2 and π_3 separately, where π_2 proves $\mathbf{u}' = [\mathbf{b}|\hat{A}_{\mathsf{id}}][\hat{Z}_1 + (\mathbf{0}|\hat{T})]$ and π_3 proves $\mathbf{u} - \mathbf{u}' = \hat{A}_{\mathsf{id}}(\hat{Z}_2 - \hat{T})$. Finally, the host outputs the signature $\sigma = (\pi_2,\ \pi_3,\ \mathbf{u}',\ \mu)$.

Comparison Between These Two Schemes

Size Comparison. In the L-DAA scheme, the TPM's secret key size is reduced to $m' \leq m$ polynomials in $\mathcal{R}_q$, instead of $2m + 1$ polynomials in [25], while keeping the same credential size. Such a change has a significant contribution in reducing the TPM's computation costs in the join and sign interfaces, as well as reducing the TPM's key and the signature sizes. For instance, the host outputs the L-DAA signature after c rounds of the proof π, the size of the response for each round is bounded by $\mathcal{O}(n)km(2\ell + 2)$ elements in $\mathbb{Z}_q$ for the host, and $\mathcal{O}(n)k(m' + 1)$ for the TPM. In [25], the size of the response for each round is bounded by $\mathcal{O}(n)km(2\ell + 2)$ for the host, and $\mathcal{O}(n)km(2\ell + 2)$ for the TPM. Thus in the L-DAA scheme, the signature's size has been significantly reduced especially for large ℓ. The verification key set in [25] consists of the $\ell + 2$ vectors of polynomials $\hat{A}_I,\ \hat{A}_i$ for $i = 0, 1, \cdots \ell$ and two polynomials $\mathbf{u}$ and $\mathbf{b}$. In the L-DAA scheme, $\hat{A}_t$ is added to the verification key set resulting with $\ell + 2$ vectors of polynomials in $\mathcal{R}_q^m$, a vector of polynomials $\hat{A}_t \in \mathcal{R}_q^{m'}$ and a polynomial $\mathbf{u}$. Note that as m' is relatively small, then adding $\hat{A}_t$ may only have a slight impact on increasing the size of the verification key set. Table 7.1 compares the space efficiency between the L-DAA scheme and the scheme presented in [25].

Computation Costs. Table 7.2 compares the computation costs between the L-DAA scheme and the scheme presented in [25] in the join and sign interfaces. To calculate $\mathsf{nym}_\mathsf{I},\ \mathsf{nym},\ \mathbf{u_t}$ and generate one round of $\pi_{\mathbf{u}_t}$ and θ_t in the join and sign interfaces of the L-DAA scheme, the TPM has to perform at most $m' + 1$ polynomial multiplications. In [25], the TPM performs at most $2m + 2$ polynomial multiplications for calculating nym_I, $\tilde{\mathbf{u}}$ and generating each round of π_1 and π_2 in the join and sign interfaces respectively. The computation costs for the host in the join interface is $2m$ polynomial multiplications for checking the equality $\mathbf{u}_h = \hat{A}_h \cdot \hat{X}_h$ for both schemes. The Issuer verifies the reponses for each round of $\pi_{\mathbf{u}_t},\ \pi_1$ in both schemes in the join interface. Thus

Schemes	[101]	[25]
TPM's Secret key	$m'n$	$(2m+1)n$
Credential	$2mn$	$2mn$
Issuer's Secret Key	m^2n^2	m^2n^2
Signature	$c\mathcal{O}(n)[k(m'+1)+km(2\ell+2)]$	$2ckm\mathcal{O}(n)(2\ell+2)$
Verification key	$(\ell+2)mn+n(m'+1)$	$(\ell+2)mn+2n$

Table 7.1: Size comparison between these two schemes

	Join		Sign		Verify	
	[101]	[25]	[101]	[25]	[101]	[25]
TPM	$m'+1$	$2m+2$	$m'+1$	$2m+2$	-	-
Host	$2m$	$2m$	$2m$	$2m$	-	-
Issuer	$m'+1$	$2m+2$	-	-	-	-
Verifier	-	-	-	-	$2m+m'$	$4m+2$

Table 7.2: Computational cost comparison between these two schemes

the issuer's computation cost for each round is thus bounded by $m'+1$ for the L-DAA scheme and $2m+2$ in [25]. In both L-DAA and [25], the host performs $2m$ polynomial multiplications for generating one round of θ_h and π_3 in the sign interfaces.

7.5 Open Issues

Based on our best knowledge, the two quantum-resistant candidates DAA schemes described in this document are the only ones known in the scientific literature. Both schemes are not efficient enough for adoption in a TPM architecture. Designing a more efficient quantum-resistant DAA scheme is a challenge.

Chapter 8

Conclusions

In this section we first recap the quantum security assumptions we make and then we summarize the main recommendations regarding the quantum-safe algorithms to be adopted in FutureTPM.

Quantum Security Assumptions

- **QS0** assumes no quantum scenario is considered at all.
- **QS1** assumes that the honest parties are classical while the adversary are equipped with quantum computing capabilities. The interaction between them is still classical.
- **QS2** assumes that the honest parties are classical while the adversary are equipped with quantum computing capabilities. The interaction between them is quantum.

Hash Functions

Many standardized hash functions with large enough bitsize (minimum 384 bit) seem to guarantee the desired quantum security properties. In particular, SHA-3 seems to exhibit good properties for use in FutureTPM. Additionally, SM3 may be the only hash function that can be used as foreign encryption technology in China.

Proposed Candidates

Here, we provide a summary of the hash candidate families.

- **SHA-3** family is the outcome from the NIST competition started in 2006. It is based on a completely new approach compared with SHA-1 and SHA-2 and, even though some papers have presented a first attack on it, they are far from being practical.
- **BLAKE** is the runner-up of the NIST competition, while BLAKE2 is the improved version. As with SHA-3, even though there are academic papers that have attacked it on a reduced version, it is claimed to be as secure as SHA-3, and is currently adopted in many security projects.
- **SM3** is approved by the Chinese standardization organism (OSCCA). It may be the only hash function that can be used as foreign encryption technology allowed in China. The amount of cryptanalysis made on SM3 is limited and some theoretical attacks have been presented in the literature.

- **PHOTON** is a lightweight family of hash functions. The internal permutation can be seen as AES-like. However, its 100 and 144 variants do not meet the minimum security strength as required in ISO/IEC 29192-1 and they must not be used as generic purpose hash functions. Instead, the other variants can benefit from extensive cryptanalysis performed on AES-based hash functions.
- **Lesamnta-LW** is a lightweight family of hash functions. It has a low hardware footprint and low working memory requirements. Its underlying component is an AES-based block cipher. It was accepted for the Round One of the NIST SHA-3 competition and although it didn't pass the round, it has neither been conceded by submitters and no substantial cryptographic weaknesses found. The hash function consequences for its full internal block cipher are unclear.

Block Ciphers

Under the QS1 scenario, Grover's algorithm can be used to achieve a quadratic speedup in the brute force key search, given enough pairs plaintext-ciphertext necessary to uniquely identify the key. Therefore, keysizes of at least 256 bits are necessary in order to have secure symmetric algorithms.
Under the QS2 scenario, instead, much more serious attacks are possible. Mitigation techniques against these attacks are currently lacking, and even AES-256 is potentially vulnerable from a theoretical standpoint. Even if from a practical standpoint these attacks seem to be hard to implement, further research toward secure block ciphers and modes of operation is recommended.

Proposed Candidates

- **AES** is a NIST standard since 2001. Some attacks from literature have been presented but none of them are practical. Some side-channel attacks have been presented that seems very efficient. The security of AES seems affected by Grove's algorithm in theory but in reality the algorithm is difficult to implement.
- **Camellia** was developed by Mitsubishi Electric and NTT and was published in 2000. It can be completely defined by a system of multivariate polynomials therefore it might be theoretically broken by algebraic attack, if they become feasible in the future. Several cryptanalysis results have been published, but none of them implies a practical attack.
- **Serpent** was published in 1998. Similarly as Camellia, it can be completely defined by a system of multivariate polynomials and, therefore, it might be theoretically broken by algebraic attack, if they become feasible in the future. Several cryptanalysis results have been published but none of them imply a practical attack.
- **Twofish** was published in 1998. It was a finalist of the Advanced encryption Standard contest. Several cryptanalysis results on Twofish have been published, but none of them implies a practical attack.

Modes of Operation

To obtain encryption only, ECB and XTS do not offer strong security guarantees but CBC and CFB are suitable even though we recommend using OFB and CTR for improved security. To obtain authenticated encryption, OCB is not recommended even if it seems very appealing in terms of performance.

Digital Signature Schemes

All the proposed schemes have been submitted to the NIST Post-Quantum Standardization process.

Proposed Candidates

- **Dilithium** is a lattice-based signature from NTRU assumption. It is based on the Fiat-Shamir with Aborts approach which uses rejection sampling to make Fiat-Shamir schemes compact and secure. It can achieve 1,2 and 3 of NIST security categories.
- **Tesla** is based on the hardness of the decisional RLWE problem. It can achieve 1,3, and 5 level of NIST security categories.
- **pqNTRUSign** is a lattice-based signature scheme based on NTRU assumptions. It is based on hash-and-sign construction and it can achieve all 5 NIST security categories.
- **FALCON** is a lattice-based signature scheme from NTRU assumptions. It is based on the theoretical framework of Gentry, Peikert and Vaikuntanathan and it is underlying hard problem is the short integer solution problem (SIS) over NTRU lattices. It can achieve all 5 NIST security categories.
- **SPHINCS** is a hash-based algorithm that relies solely on the security of the underlying cryptographic hash function. It is a stateless protocol and can be a drop-in replacement for RSA and ECDSA.

Remarks: it is difficult to understand if the schemes are efficient enough to be included in FutureTPM. In addition, finding the right balance between quantum-resistant levels (QS1 or QS2) and performance is not trivial.

Public-Key Encryption and Key Exchange

In this section we summarize the promising candidates for post-quantum key exchange and public key encryption from NIST standardization process.

Proposed Candidates

- **NewHope** is a suite of key encapsulations mechanism that are based on the conjecture of quantum hardness of the RLWE problem. It can achieve high performance on a wide range of platforms and is memory efficient. It can achieve 1 and 5 of NIST security categories. There are no attacks that can significantly exploit RLWE but this may change in future.
- **Frodo** differs from NewHope in the choice of the underlying lattice-problems. Frodo uses LWE problem, which is considered a weaker assumption. It does not need an error reconciliation function and can use very simple encoding and decoding because it was designed

to prioritize simplicity and security over performance and optimization. The main disadvantage is the size of the public key and ciphertext. It can achieve 1 and 3 of NIST security categories.

- **Kyber** is a recent construction that, relying on MLWE, seems less vulnerable to attacks exploiting the algebraic structure of ideal lattices than RLWE. In practice it allows easier scaling of security parameters as security levels can be achieved that are hard to reach using RLWE. It can achieve 1,3 and 5 of NIST security categories.
- **BIKE** is code-based cryptography that relies on the hardness of decoding (random) linear codes. It is based on computationally hard problem in coding theory when the so-called binary Goppa codes are used. Binary Goppa codes make public keys very large. It works only with ephemeral keys as it is currently not clear how to protect it against chosen ciphertext attacks. It can achieve 1, 3 and 5 of NIST security categories.

Remarks: from a first analysis, it seems that many schemes might be vulnerable to side-channels attacks. While for code-based schemes it seems easier to prevent leakage, for lattice-based this is unclear. Other schemes have been proposed that are based on assumptions that are either new or not well studied.

References

[1] IEEE Standard Specification for Public Key Cryptographic Techniques based on Hard Problems over Lattices. *IEEE Std 1363.1-2008*, pages C1–69, March 2009.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108, 1996.

[3] Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardoni, Christian Schaffner, and Michael St. Jules. Computational security of quantum encryption. In *Information Theoretic Security - 9th International Conference, ICITS 2016, Tacoma, WA, USA, August 9-12, 2016, Revised Selected Papers*, pages 47–71, 2016.

[4] Gorjan Alagic, Tommaso Gagliardoni, and Christian Majenz. Unforgeable quantum encryption. 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 489–519, 2018.

[5] Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. Quantum-secure message authentication via blind-unforgeability. http://arxiv.org/abs/1803.03761v1.

[6] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, and Martin Tomlinson. NTS-KEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[7] Martin R Albrecht, Christian Hanser, Andrea Hoeller, Thomas Pöppelmann, Fernando Virdia, and Andreas Wallner. Learning with errors on rsa co-processors. https://eprint.iacr.org/2018/425.

[8] Sk Subidh Ali and Debdeep Mukhopadhyay. Differential fault analysis of Twofish. In *International Conference on Information Security and Cryptology*, pages 10–28. Springer, 2012.

[9] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe Thomas Pöppelmann, and Douglas Stebila. NewHope. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[10] Erdem Alkim, Joppe W. Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[11] Erdem Alkim, Leo Ducas, Thomas Poppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. Cryptology ePrint Archive, Report 2015/1092, 2015. https://eprint.iacr.org/2015/1092.

[12] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. NewHope without reconciliation. *IACR Cryptology ePrint Archive*, 2016:1157, 2016.

[13] Mayuresh Vivekanand Anand, Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation. In *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, pages 44–63, 2016.

[14] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A candidate block cipher for the Advanced Encryption Standard. *Página oficial do SERPENT, disponível em http://www. cl. cam. ac. uk/˜ rja14/serpent. html*, 2005.

[15] ANSI. Ansi x9.98-2010: Financial services - lattice-based polynomial public key establishment algorithm for the financial services industry. Technical report, 2010.

[16] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zemor. BIKE. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[17] Will Arthur, David Challener, and Kenneth Goldman. *Decrypt/Encrypt Sessions*, pages 271–287. Apress, Berkeley, CA, 2015.

[18] C Ashokkumar, Ravi Prakash Giri, and Bernard Menezes. Highly efficient algorithms for AES key retrieval in cache access attacks. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 261–275. IEEE, 2016.

[19] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2 - fast secure hashing. https://blake2.net/.

[20] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: simpler, smaller, fast as md5, January 2013. https://blake2.net/blake2.pdf.

[21] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2X, December 2016. https://blake2.net/blake2x.pdf.

[22] Josep Balasch, Baris Ege, Thomas Eisenbarth, Benoit Gérard, Zheng Gong, Tim Guneysu, Stefan Heyse, Stéphanie Kerckhof, François Koeune, Thomas Plos, Thomas Poppelmann, Francesco Regazzoni, François-Xavier Standaert, Gilles Van Assche, Ronny Van Keer, Loic van Oldeneel tot Oldenzeel, and Ingo von Maurich. Compact implementation and performance evaluation of hash functions in ATtiny devices. Cryptology ePrint Archive: Report 2012/507, 2012. https://eprint.iacr.org/2012/507.

[23] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAkem. Technical report, National Institute of Standards and Technology,

2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[24] Gustavo Banegas and Daniel J Bernstein. Low-communication parallel quantum multi-target preimage search. In *International Conference on Selected Areas in Cryptography*, pages 325–335. Springer, 2017.

[25] Rachid El Bansarkhani and Ali El Kaafarani. Direct anonymous attestation from lattices. cryptology eprint archive, report 2017/1022, 2017.

[26] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[27] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[28] Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hulsing, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. Sphincs+. Submission, November 2017.

[29] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography-dealing with the fallout of physics success. *IACR Cryptology ePrint Archive*, 2017:314, 2017.

[30] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the mceliece cryptosystem. In *http://cr.yp.to/papers.html#mceliece*, pages 31–46, 2008.

[31] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In *Advances in Cryptology (EUROCRYPT)*, volume 4965 of *LNCS*, pages 181–197, Istanbul, Turkey, April 2008. Springer.

[32] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Cryptographic sponge functions, January 2011. https://keccak.team/files/CSF-0.1.pdf.

[33] Eli Biham and Orr Dunkelman. A framework for iterative hash functions - HAIFA. Cryptology ePrint Archive: Report 2007/278, 2007. https://eprint.iacr.org/2007/278.

[34] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA.

[35] Alex Biryukov and Christophe De Cannière. Block ciphers and systems of quadratic equations. In *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, pages 274–289, 2003.

[36] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 299–319. Springer, 2010.

[37] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–18. Springer, 2009.

[38] Alex Biryukov, Mario Lamberger, Florian Mendel, and Ivica Nikolić. Second-order differential collisions for reduced SHA-256. In *Advances in Cryptology (ASIACRYPT)*, volume 7073 of *LNCS*, pages 270–287, Seoul, South Korea, December 2011. Springer.

[39] Céline Blondeau. Impossible differential attack on 13-round Camellia-192. *Information Processing Letters*, 115(9):660–666, 2015.

[40] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 289–302, 1984.

[41] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 344–371. Springer, 2011.

[42] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 41–69, 2011.

[43] Dan Boneh and Mark Zhandry. Quantum-secure Message Authentication Codes. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 592–608, 2013.

[44] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 361–379, 2013.

[45] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018, 2016.

[46] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570, 2015.

[47] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS - Kyber: a CCA-secure module-lattice-based KEM. *IACR Cryptology ePrint Archive*, 2017:634, 2017. to appear in IEEE European Symposium on Security and Privacy 2018, EuroS&P 2018.

[48] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *International Workshop on Public Key Cryptography*, pages 499–517. Springer, 2010.

[49] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Latin American Symposium on Theoretical Informatics (LATIN)*, volume 1380 of *LNCS*, pages 163–169, Campinas, Brazil, April 1998. Springer.

[50] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145. ACM, 2004.

[51] 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 *IEEE Security & Privacy – S&P 2017*. IEEE, 2017.

[52] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation with subverted TPMs. In *Crypto 2017*.

[53] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In *Public-Key Cryptography – PKC 2016*, volume LNCS 9615, pages 234–264. Springer, 2016.

[54] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

[55] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. On quantum indifferentiability. Cryptology ePrint Archive: Report 2018/257, 2018. https://eprint.iacr.org/2018/257.

[56] D. Chaum and E van Heyst. Group signatures. In *D.W. Davies (ed.) Advances in Cryptology - EUROCRYPT 1991*, pages 257–265. Springer, 1991).

[57] Liqun Chen. A DAA scheme requiring less tpm resources. In *Proceedings of the 5th China International Conference on Information Security and Cryptology (Inscrypt 2009)*, volume LNCS 6151, pages 350—-365. Springer, 2010.

[58] Craig Costello, Patrick Longa, and Michael Naehrig. Efficient algorithms for supersingular isogeny diffie-hellman. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 572–601, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[59] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. Post-quantum security of the Sponge construction. In *International Conference on Post-Quantum Cryptography (PQCrypto)*, volume 10786 of *LNCS*, pages 185–204, Fort Lauderdale, FL, USA, April 2018. Springer.

[60] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir transformation in a quantum world. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 62–81, 2013.

[61] Ivan Bjerre Damgård. A design principle for hash functions. In *Advances in Cryptology – CRYPTO)*, volume 435 of *LNCS*, pages 416–427, Santa Barbara, California, USA, August 1989. Springer.

[62] A Daniel, B Lejla, et al. Initial recommendations of long-term secure post-quantum systems. *PQCRYPTO. EU. Horizon*, 2020, 2015.

[63] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. SABER. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[64] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[65] Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In *Fast Software Encryption (FSE)*, volume 8424 of *LNCS*, pages 219–240, Singapore, March 2013. Springer.

[66] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552, 1991.

[67] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[68] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals–Dilithium: Digital signatures from module lattices.

[69] Michael Düll, Björn Haase, Gesine Hinterwälder, Michael Hutter, Christof Paar, Ana Helena Sánchez, and Peter Schwabe. High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers. *Des. Codes Cryptography*, 77(2-3):493–514, 2015.

[70] Edward Eaton and Fang Song. Making existential-unforgeable signatures strongly unforgeable in the quantum random-oracle model. In *10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2015, May 20-22, 2015, Brussels, Belgium*, pages 147–162, 2015.

[71] M-J. Saarinen (Ed.). The BLAKE2 cryptographic hash and message authentication code (MAC). RFC 7693, RFC Editor, November 2015.

[72] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 60–79, 2012.

[73] Niels Ferguson. Impossible differentials in Twofish. *Counterpane Systems. October*, 19, 1999.

[74] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU.

[75] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 537–554, 1999.

[76] Tommaso Gagliardoni. *Quantum Security of Cryptographic Primitives*. PhD thesis, Darmstadt University of Technology, Germany, 2017.

[77] Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 60–89, 2016.

[78] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[79] Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: improved attacks for AES-like permutations. In *International Workshop on Fast Software Encryption*, pages 365–383. Springer, 2010.

[80] Ken Goldman. IBM's Software TPM 2.0, 2018. https://sourceforge.net/projects/ibmswtpm2/.

[81] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying grover's algorithm to AES: quantum resource estimates. In *International Workshop on Post-Quantum Cryptography*, pages 29–43. Springer, 2016.

[82] Matthew Green. How to choose an authenticated encryption mode. https://blog.cryptographyengineering.com/2012/05/19/how-to-choose-authenticated-encryption/.

[83] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.

[84] David Gullasch, Endre Bangerter, and Stephan Krenn. Cache games–bringing access-based cache attacks on AES to practice. In *2011 IEEE Symposium on Security and Privacy – SP*, pages 490–505. IEEE, 2011.

[85] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 222–239, Santa Barbara, CA, USA, August 2011. Springer.

[86] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 789–815, 2016.

[87] Mike Hamburg. Three Bears. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[88] Shoichi Hirose, Kota Ideguchi, Hidenori Kuwakado, Toru Owada, Bart Preneel, and Hirotaka Yoshida. An AES based 256-bit hash function for lightweight applications: Lesamnta-LW. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E95.A(1):89–99, January 2012.

[89] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, pages 3–18, Cham, 2017. Springer International Publishing.

[90] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 267–288, 1998.

[91] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 341–371, 2017.

[92] A. Huelsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen. XMSS: Extended Hash-Based Signatures, 2018. https://datatracker.ietf.org/doc/draft-irtf-cfrg-xmss-hash-based-signatures/.

[93] A. Huelsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen. Xmss: extended merkle signature scheme. RFC 8391, RFC Editor, May 2018.

[94] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 8–26, 1988.

[95] Infineon. Infineon Chip Card & Security ICs Portfolio, 2017. https://www.infineon.com/dgdl/Infineon-Chip_Card_Security_ICs_Portfolio_2017-SG-v10_17-EN.pdf?fileId=5546d4624933b875014999016c6e2bde.

[96] International Organization for Standardization. ISO/IEC 29192-5:2016, information technology - security techniques - lightweight cryptography - part 5: Hash-functions, August 2016.

[97] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. SIKE. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[98] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, pages 19–34, 2011.

[99] Keting Jia and Ning Wang. Impossible differential cryptanalysis of 14-round Camellia-192. In *Australasian Conference on Information Security and Privacy*, pages 363–378. Springer, 2016.

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

[101] Nada EL Kassem, Liqun Chen, Rachid El Bansarkhani, Ali El Kaafarani, Jan Camenisch, and Patrick Hough. L-DAA: Lattice-based direct anonymous attestation. cryptology eprint archive, report 2018/401, 2018.

[102] Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for preimages: attacks on Skein-512 and the SHA-2 family. In *Fast Software Encryption (FSE)*, volume 7549 of *LNCS*, pages 244–263, Washington, DC, USA, March 2012. Springer.

[103] Aleksandar Kircanski, Yanzhao ShenGaoli Wan, and Amr M. Youssef. Boomerang and slide-rotational analysis of the SM3 hash function. In *Selected Areas in Cryptography (SAC)*, volume 7707 of *LNCS*, pages 304–320, Windsor, ON, Canada, August 2012. Springer.

[104] Takeru Koie, Takanori Isobe, Yosuke Todo, and Masakatu Morii. Low-data complexity attacks on Camellia. In *International Conference on Applications and Techniques in Information Security*, pages 128–140. Springer, 2017.

[105] Leibo Li, Keting Jia, Xiaoyun Wang, and Xiaoyang Dong. Meet-in-the-middle technique for truncated differential and its applications to CLEFIA and camellia. In *International Workshop on Fast Software Encryption*, pages 48–70. Springer, 2015.

[106] Zhiqiang Liu, Bing Sun, Qingju Wang, Kerem Varici, and Dawu Gu. Improved zero-correlation linear cryptanalysis of reduced-round camellia under weak keys. *IET Information Security*, 10(2):95–103, 2016.

[107] Stefan Lucks. The saturation attack—a bait for Twofish. In *International Workshop on Fast Software Encryption*, pages 1–15. Springer, 2001.

[108] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 1–23, 2010.

[109] Chujiao Ma, John Chandy, and Zhijie Shi. Algebraic side-channel attack on Twofish. *Journal of Internet Services and Information Security (JISIS)*, 7(2):32–43, 2017.

[110] Marco Macchetti. Characteristics of key-dependent s-boxes: the case of Twofish. *IACR Cryptology ePrint Archive*, 2005:115, 2005.

[111] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, pages 419–453, 1988.

[112] Ueli Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of Cryptography Conference (TCC)*, volume 2951 of *LNCS*, pages 21–39, Cambridge, MA, USA, February 2004. Springer.

[113] Robert J Mceliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[114] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Adrien Hauteville, and Gilles Zemor. Ouroboros-R. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[115] Carlos Aguilar Melchor, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 648–652, 2011.

[116] Florian Mendel, Tomislav Nad, and Martin Schlaffer. Finding collisions for round-reduced SM3. In *Topics in Cryptology (CT-RSA)*, volume 7779 of *LNCS*, pages 174–188, San Francisco,CA, USA, March 2013. Springer.

[117] Ralph C. Merkle. A certified digital signature. In *Advances in Cryptology (CRYPTO*, volume 435 of *LNCS*, pages 218–238, Santa Barbara, California, USA, August 1989. Springer.

[118] McGuire Michael. Into the web of profit. `https://www.surrey.ac.uk/news/new-report-reveals-cybercriminal-spending-behaviours`.

[119] Paweł Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced Keccak. In *Fast Software Encryption (FSE)*, volume 8424 of *LNCS*, pages 241–262, Singapore, March 2013. Springer.

[120] Sean Murphy and Matthew J. B. Robshaw. Key-dependent S-boxes and differential cryptanalysis. *Designs, Codes and Cryptography*, 27(3):229–255, 2002.

[121] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437, 1990.

[122] National Institute of Standards and Technology. FIPS PUB 180-1: Secure hash standard (SHS), April 1995.

[123] National Institute of Standards and Technology. FIPS PUB 180-4: Secure hash standard (SHS), August 2015.

[124] National Institute of Standards and Technology. FIPS PUB 202: SHA-3 standard: Permutation-based hash and extendable-output functions, August 2015.

[125] Cuong Nguyen, Lai Tran, and Khoa Nguyen. On the resistance of serpent-type 4 bit S-boxes against differential power attacks. In *Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on*, pages 542–547. IEEE, 2014.

[126] Ruben Niederhagen and Michael Waidner. Practical Post-Quantum Cryptography. Technical report, Fraunhofer Institute for Secure Information Technology, 8 2017.

[127] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[128] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, 2016. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[129] NIST. Post-quantum cryptography, 2017. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.

[130] Thomaz Oliveira, Julio López, Diego F. Aranha, and Francisco Rodríguez-Henríquez. Two is the fastest prime: lambda coordinates for binary elliptic curves. *Journal of Cryptographic Engineering*, 4(1):3–17, Apr 2014.

[131] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 271–289, 2006.

[132] Albrecht Petzoldt, Ming-Shing Chen, Bo-Yin Yang, Chengdong Tao, and Jintai Ding. Design principles for hfev- based multivariate signature schemes. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 311–334, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[133] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, 1979.

[134] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[135] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[136] R.L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Asiacrypt 2001*, pages 552–565. Springer, 2001.

[137] Phillip Rogaway. Formalizing human ignorance. In *Progress in Cryptology (VIETCRYPT)*, volume 4341 of *LNCS*, pages 211–228, Hanoi, Vietnam, September 2006. Springer.

[138] Phillip Rogaway and Tom Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption (FSE)*, volume 3017 of *LNCS*, pages 371–388, Delhi, India, February 2004. Springer.

[139] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553, 1999.

[140] Yu Sasaki and Kazumaro Aoki. Improved integral analysis on tweaked Lesamnta. In *International Conference on Information Security and Cryptology (ICISC)*, volume 7259 of *LNCS*, pages 1–17, Seoul, South Korea, December 2011. Springer.

[141] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. NTRU-HRSS-KEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[142] J. Schmitz, J. Loew, J. Elwell, D. Ponomarev, and N. Abu-Ghazaleh. TPM-SIM: A framework for performance evaluation of Trusted Platform Modules. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 236–241, June 2011.

[143] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15:23, 1998.

[144] Peter Schwabe, Roberto Avanzi, Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehle. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[145] Yaron Sella and Aviad Kipnis. Attack-resistant multivariate signature scheme, 2010. https://patents.google.com/patent/US8811608.

[146] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, page 289, 1994.

[147] Bhupendra Singh, Lexy Alexander, and Sanjay Burman. On algebraic relations of Serpent S-boxes. *IACR Cryptology ePrint Archive*, 2009:38, 2009.

[148] François-Xavier Standaert, Tal G. 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*, pages 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[149] Francois-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. Cryptology ePrint Archive, Report 2009/341, 2009. https://eprint.iacr.org/2009/341.

[150] Standardization Administration of the People's Republic of China. GB/T 32905-2016: Information security techniques - SM3 cryptographic hash algorithm, August 2016.

[151] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 570–596, 2017.

[152] STMICROELECTRONICS. Trusted Platform Module ST33TPHF20SPI, 2017. https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3065.pdf.

[153] Biaoshuai Tao and Hongjun Wu. Improving the biclique cryptanalysis of AES. In *Australasian Conference on Information Security and Privacy*, pages 39–56. Springer, 2015.

[154] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 192–216, 2016.

[155] TCG. TCG Algorithm Registry. Technical report, 2015. https://trustedcomputinggroup.org/wp-content/uploads/TCG-_Algorithm_Registry_Rev_1.27_FinalPublication.pdf.

[156] TCG. TPM 2.0 Mobile Common Profile. Technical report, 2015. https://www.trustedcomputinggroup.org/wp-content/uploads/TPM_2.0_Mobile_Common_Profile_v2r31_FINAL.pdf.

[157] TCG. Trusted Platform Module Library Part 1: Architecture. Technical report, 2016. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf.

[158] TCG. Trusted Platform Module Library Part 2: Structures. Technical report, 2016. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-2-Structures-01.38.pdf.

[159] TCG. TCG PC Clinet Platform TPM Profile (PTP) Specification. Technical report, TCG, 2017. https://trustedcomputinggroup.org/wp-content/uploads/PC-Client-Specific-Platform-TPM-Profile-for-TPM-2-0-v1-03-22-170516_final.pdf.

[160] Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. In *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, pages 191–205, 2006.

[161] Cihangir Tezcan, Halil Kemal Taşkın, and Murat Demircioğlu. Improbable differential attacks on SERPENT using undisturbed bits. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 145. ACM, 2014.

[162] The Crypto++ Project. Crypto++ 6.0.0 benchmarks. https://www.cryptopp.com/benchmarks.html.

[163] Vladimir Valyukh. Performance and comparison of postquantum cryptographic algorithms. Master's thesis, Link´oping University, Department of Electrical Engineering, 2017.

[164] various authors. Post-quantum crypto library for the ARM Cortex-M4. Website, 2018. https://github.com/mupq/pqm4, accessed 15 May 2015.

[165] Virtual Applications and Implementations Research Lab. eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to/results-hash.html.

[166] Meiqin Wang, Yue Sun, Elmar Tischhauser, and Bart Preneel. A model for structure attacks, with applications to PRESENT and Serpent. In *Fast Software Encryption*, pages 49–68. Springer, 2012.

[167] Xiaoyun Wang and Hongbo Yu. SM3 cryptographic hash algorithm. *Journal of Information Security Research*, 2(11):983–994, January 2016.

[168] Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A post-quantum digital signature scheme based on supersingular isogenies. In *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, pages 163–181, 2017.

[169] Mark Zhandry. A note on quantum-secure PRPs. *CoRR*, abs/1611.05564, 2016.

[170] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. Cryptology ePrint Archive: Report 2018/276, 2018. https://eprint.iacr.org/2018/276.

[171] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. pqNTRUSign.

[172] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. NTRUEncrypt. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[173] Jian Zou, Wenling Wu, Shuang Wu, Bozhan Su, and Le Dong. Preimage attacks on step-reduced sm3 hash function. In *Information Security and Cryptology (ICISC)*, volume 7259 of *LNCS*, pages 375–390, Seoul, Korea, November 2011. Springer.

Appendix A

List of Abbreviations

Abbreviation	Translation
AES	Advanced Encryption Standard
DAA	Direct Anonymous Attestation
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
EC	European Commission
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve DSA
QR	Quantum-Resistant
QROM	Quantum Random Oracle Model
ROM	Random Oracle Model
SHA	Secure Hash Algorithm
TBD	To Be Determined
TCG	Trusted Computing Group
TDES	Triple-DES
TPM	Trusted Plaform Module

Appendix B

List of Algorithms Supported by TPM V2.0

The TCG Algorithm Registry enlists all possible algorithms that could be used within TCG for other specifications. It is not specifically dedicated to TPM 2.0, this is the reason why it should be combined with other documents.

B.1 Hash Functions

As mentioned in the TCG specification [159], it is observed that SHA1 is required for legacy support, but may be removed in future. The following tables are the supported hash functions for TPM2.0:

Algorithm Name	Type	Classification	Reference	Comments
TPM_ALG_SHA1	H		ISO/IEC 10118-3	redefinition for documentation consistency
TPM_ALG_SHA256	H		ISO/IEC 10118-3	the SHA 256 algorithm
TPM_ALG_SHA384	H	A	ISO/IEC 10118-3	the SHA 384 algorithm
TPM_ALG_SHA512	H	A	ISO/IEC 10118-3	the SHA 512 algorithm
TPM_ALG_SM3_256	H	A	GM/T 0004-2012	SM3 hash algorithm

Algorithms that support the implementation of the hash function:

Algorithm Name	Type	Classification	Reference	Comments
TPM_ALG_HMAC	H X		ISO/IEC 9797-2	Hash Message Authentication Code (HMAC) algorithm
TPM_ALG_MGF1	H M		IEEE Std 1363TM-2000 IEEE Std 1363aTM- 2004	hash-based mask-generation function
TPM_ALG_KEYEDHASH	H O E X S		TCG TPM 2.0 library specification	an encryption or signing algorithm using a keyed hash. May also refer to a data object that is neither signing nor encrypting
TPM_ALG_XOR	H S		TCG TPM 2.0 library specification	the XOR encryption algorithm
TPM_ALG_OEAP	A E H		IETF RFC 8017	a padding algorithm defined in section 7.1 (RSAES_OAEP)
TPM_ALG_KDF1_SP800_56A	H M		NIST SP800-56A	concatenation key derivation function
TPM_ALG_KDF2	H M	A	IEEE Std 1363a-2004	key derivation function KDF2
TPM_ALG_KDF1_SP800_108	H M		NIST SP800-108	a key derivation method

The Algorithm Registry Specification [155] also determines the following hash algorithms which are not, yet, defined in the TPM2.0 Library specification [158].

Algorithm Name	Type	Classification	Reference	Comments
TPM_ALG_SHA3_256	H	A	NIST PUB FIPS 202	Hash algorithm producing a 256-bit digest
TPM_ALG_SHA3_384	H	A	NIST PUB FIPS 202	Hash algorithm producing a 384-bit digest
TPM_ALG_SHA3_512	H	A	NIST PUB FIPS 202	Hash algorithm producing a 512-bit digest

Type:
$\{$**A**:*asymmetric algorithm*, **X**: *signing algorithm*, **M**: *a method such as a mask generation function*, **N**: *an anonymous signing algorithm*, **H**: *hash algorithm*, **O**: *object type*, **E**: *encryption mode*, **S**: *symmetric algorithm* $\}$

C:
$\{$**A:** *assigned, not TCG Standard*$\}$

B.2 Block Ciphers

Although Algorithm Registry [155], defines TDES, it is stated that TCG compliant device shall not allow a triple DES key to be used if two from the three keys are the same (i.e., K1 = K2, or K2

= K3).
In general, TDES is not recommended for usage after 2014. Three-key TDES is recommended only for three different keys [159]. It worth mentioning that the TPM2.0 Library Specification [158] does not define TDES at all.
The following tables are the supported block ciphers for TPM2.0:

Algorithm Name	**Type**	**Classification**	**Reference**	**Comments**
TPM_ALG_SHA3_256	H	A	NIST PUB FIPS 202	Hash algorithm producing a 256-bit digest
TPM_ALG_SHA3_384	H	A	NIST PUB FIPS 202	Hash algorithm producing a 384-bit digest
TPM_ALG_SHA3_512	H	A	NIST PUB FIPS 202	Hash algorithm producing a 512-bit digest

Algorithms that support the implementation of block cipher:

Algorithm Name	**Type**	**Classification**	**Reference**	**Comments**
TPM_ALG_XOR	H S		TCG TPM 2.0 library specification	the XOR encryption algorithm
TPM_ALG_SYMCIPHER	O S		TCG TPM 2.0 library specification	the object type for a symmetric block cipher
TPM_ALG_CTR	S E	A	ISO/IEC 10116	Counter mode – if implemented, all symmetric block ciphers (S type) implemented shall be capable of using this mode.
TPM_ALG_OFB	S E	A	ISO/IEC 10116	Output Feedback mode – if implemented, all symmetric block ciphers (S type) implemented shall be capable of using this mode
TPM_ALG_CBC	S E	A	ISO/IEC 10116	Cipher Block Chaining mode – if implemented, all symmetric block ciphers (S type) implemented shall be capable of using this mode.
TPM_ALG_CFB	S E		ISO/IEC 10116	Cipher Feedback mode – if implemented, all symmetric block ciphers (S type) implemented shall be capable of using this mode.
TPM_ALG_ECB	S E	A	ISO/IEC 10116	Electronic Codebook mode – if implemented, all symmetric block ciphers (S type) implemented shall be capable of using this mode.

Based on the above, the modes of operation are: CTR, OFB, CBC, CFB, ECB.

Moreover, as defined in the Algorithm Registry [155] the following block cipher algorithms are not, yet, supported in the TPM2.0 Library Specification [158].

Algorithm Name	Type	Classification	Reference	Comments
TPM_ALG_TDES	S	A	ISO/IEC 18033-3	block cipher with various key sizes (Triple Data Encryption Algorithm, commonly called Triple Data Encryption Standard)
TPM_ALG_CMAC	S X	A	ISO/IEC 9797-1:2011	block Cipher-based Message Authentication Code (CMAC)

Type:
{**X:** *signing algorithm*, **H:** *hash algorithm*, **O:** *object type*, **E:** *encryption mode*, **S:** *symmetric algorithm*}
C:
{**A:** *assigned, not TCG Standard*}
[TCG Algorithm Registry, Trusted Platform Module Library Part 2: Structures]

B.3 Elliptic Curves

The supported curves from the TPM are the following [155]:

Algorithm Name	Classification	Comments
TPM_ECC_NIST_P192		192 bits public key length
TPM_ECC_NIST_P224		224 bits public key length
TPM_ECC_NIST_P256	M	256 bits public key length
TPM_ECC_NIST_P384		384 bits public key length
TPM_ECC_NIST_P521		521 bits public key length
TPM_ECC_BN_P256	M	256 bits public key length curve to support ECDAA
TPM_ECC_BN_P63		638 bits public key length curve to support ECDAA
TPM_ECC_SM2_P256		256 bits public key length

Classification:
{**M:** *Mandatory, if ECC is defined then these curves must be supported*}
Also, the algorithms using elliptic curve cryptography in TPM are the following:

Algorithm Name	Type	Classification	Reference	Comments
TPM_ALG_ECDSA	A X		ISO/IEC 14888-3	signature algorithm using ECC
TPM_ALG_ECDH	A M		NIST SP800-56A	secret sharing using ECC
TPM_ALG_ECDAA	A X N		TCG TPM 2.0 Library specification	anonymous signing scheme based on ECC
TPM_ALG_SM2	A X	A	GM/T 0003.1-2012 GM/T 0003.2-2012 GM/T 0003.3-2012 GM/T 0003.5-2012	SM2 – depending on the context could be signature or key exchange algorithm
TPM_ALG_ECSCHNORR	A X		TCG TPM 2.0 Library specification	Schnorr signature based on ECC
TPM_ALG_ECMQV	A M	A	NIST SP800-56A	2-phase key exchange based on ECC
TPM_ALG_KDF1_SP800_56A	H M		NIST SP800-56A	concatenation key derivation function

Type:
{**A:** *asymmetric algorithm*, **X:** *signing algorithm*, **M:** *a method such as a mask generation function*, **N:** *an anonymous signing algorithm*, /textbfH: *hash algorithm*}

C:
{**A:** *assigned, not TCG Standard*}

Whenever ECC is supported in the TPM, then it is required that ECDSA (digital signatures) and ECDH (secret sharing) must be supported as well, with key sizes of at least 256 bits [157]. For the TCG defined ECDAA protocol, the curve described is a Barreto-Naehrig (BN) elliptic curve [157]. Moreover, in the client side if ECC is implemented, the curves that must be supported are *TPM_ECC_NIST_P256* and *TPM_ECC_BN_P256* (to support anonymous attestation on ECDAA) [159], [156]. In addition, the support of algorithms *TPM_ALG_SM2* and *TPM_ALG_ECMQV* is not mandatory [159], [155].
Elliptic curve cryptography can use binary or prime fields. TPM proposes prime fields over binary. The reason why prime fields are sometimes preferred over binary mainly for performance gains. That is, prime field curves are usually faster on general purpose CPU's as the integer multiplier circuit is usually faster than the binary one [130].

Appendix C

DAA Security Model Details

In this section we explain the interfaces of F^{l}_{daa} and identify the checks, labelled in Roman numerals, that are performed by the ideal functionality.

SETUP

On the input(SETUP, sid) from the issuer I, F^{l}_{daa} does the following:

- Verify that $(I,\ \mathsf{sid}') = \mathsf{sid}$ and output (SETUP, sid) to $\mathcal{S}$.
- SET Algorithms. Upon receiving the algorithms (Kgen, sig, ver, link, identify) from the simulator $\mathcal{S}$, it checks that (ver, link, identify) are deterministic [Check-I].
- Output (SETUPDONE, sid) to I.

JOIN

1. JOIN REQUEST: On input (JOIN, $\mathsf{sid},\ \mathsf{jsid},\ \mathsf{tpm_i}$) from the host $\mathsf{host_j}$ to join the TPM $\mathsf{tpm_i}$, the ideal functionality F^{l}_{daa} proceeds as follows:
   - Create a join session $\langle \mathsf{jsid},\ \mathsf{tpm_i},\ \mathsf{host_j}$, request$\rangle$.
   - Output (JOINSTART, $\mathsf{sid},\ \mathsf{jsid},\ \mathsf{tpm_i},\ \mathsf{host_j}$) to $\mathcal{S}$.
2. JOIN REQUEST DELIVERY: Proceed upon recieving delivery notification from $\mathcal{S}$.
   - Update the session record to $\langle \mathsf{jsid},\ \mathsf{tpm_i},\ \mathsf{host_j}$, delivery$\rangle$.
   - If I or $\mathsf{tpm_i}$ is honest and $\langle \mathsf{tpm_i},\ \star,\ \star\rangle$ is already in Members, output $\perp$ [Check II].
   - Output (JOINPROCEED, $\mathsf{sid},\ \mathsf{jsid},\ \mathsf{tpm_i}$) to I.
3. JOIN PROCEED:
   - Upon receiving an approval from I, F^{l}_{daa} updates the session record to $\langle \mathsf{jsid},\ \mathsf{sid},\ \mathsf{tpm_i},\ \mathsf{host_j}$, complete$\rangle$.
   - Output (JOINCOMPLETE, $\mathsf{sid},\ \mathsf{jsid}$) to $\mathcal{S}$.
4. KEY GENERATION: On input (JOINCOMPLETE, $\mathsf{sid},\ \mathsf{jsid},\ gsk$) from $\mathcal{S}$.
   - If both $\mathsf{tpm_i}$ and $\mathsf{host_j}$ are honest, set $gsk = \perp$.
   - Else, verify that the provided gsk is eligible by performing the following checks:

  - If $\mathsf{host_j}$ is corrupt and $\mathsf{tpm_i}$ is honest, then CheckGskHonest(gsk)=1 [Check III].
  - If $\mathsf{tpm_i}$ is corrupt, then CheckGskCorrupt(gsk)=1 [Check IV].
- Insert $\langle \mathsf{tpm_i},\ \mathsf{host_j},\ gsk\rangle$ into Members, and output (JOINED, $\mathsf{sid},\ \mathsf{jsid}$) to $\mathsf{host_j}$.

SIGN

1. SIGN REQUEST: On input (SIGN, $\mathsf{sid},\ \mathsf{ssid},\ \mathsf{tpm_i},\ \mu,\ \mathsf{bsn}$) from the host $\mathsf{host_j}$ requesting a DAA signature by a TPM $\mathsf{tpm_i}$ on a message μ with respect to a basename bsn, the ideal functionality does the following:
   - Abort if I is honest and no entry $\langle \mathsf{tpm_i},\ \mathsf{host_j},\ \star\rangle$ exists in Members.
   - Else, create a sign session $\langle \mathsf{ssid},\ \mathsf{tpm_i},\ \mathsf{host_j},\ \mu,\ \mathsf{bsn},\ \mathsf{request}\rangle$.
   - Output (SIGNSTART, $\mathsf{sid},\ \mathsf{ssid},\ \mathsf{tpm_i},\ \mathsf{host_j},\ l(\mu, \mathsf{bsn})$) to $\mathcal{S}$.
2. SIGN REQUEST DELIVERY: On input (SIGNSTART, $\mathsf{sid},\ \mathsf{ssid}$) from $\mathcal{S}$, update the session to $\langle \mathsf{ssid},\ \mathsf{tpm_i},\ \mathsf{host_j},\ \mu,\ \mathsf{bsn},\ \mathsf{delivered}\rangle$. F^l_{daa} output (SIGNPROCEED, $\mathsf{sid},\ \mathsf{ssid},\ \mu, \mathsf{bsn}$) to $\mathsf{tpm_i}$.
3. SIGN PROCEED: On input (SIGN PROCEED, $\mathsf{sid},\ \mathsf{ssid}$) from $\mathsf{tpm_i}$
   - Update the records $\langle \mathsf{ssid},\ \mathsf{tpm_i},\ \mathsf{host_j},\ \mu,\ \mathsf{bsn}$, delivered$\rangle$.
   - Output (SIGNCOMPLETE, $\mathsf{sid},\ \mathsf{ssid}$) to $\mathcal{S}$.
4. SIGNATURE GENERATION: On the input (SIGNCOMPLETE, $\mathsf{sid},\ \mathsf{ssid},\ \sigma$) from $\mathcal{S}$, if both $\mathsf{tpm_i}$ and $\mathsf{host_j}$ are honest then:
   - Ignore the adversary's signature σ.
   - If $\mathsf{bsn} \neq \perp$, then retrieve gsk from the $\langle \mathsf{tpm_i},\ \mathsf{bsn},\ gsk\rangle \in$ DomainKeys.
   - If $\mathsf{bsn} = \perp$ or no gsk was found, generate a fresh key $gsk \leftarrow Kgen(1^\lambda)$.
   - Check CheckGskHonest(gsk)=1 [Check V].
   - Store $\langle \mathsf{tpm_i},\ \mathsf{bsn},\ gsk\rangle$ in DomainKeys.
   - Generate the signature $\sigma \leftarrow sig(gsk,\ \mu,\ \mathsf{bsn})$.
   - Check ver($\sigma,\ \mu,\ \mathsf{bsn}$)=1 [Check VI].
   - Check identify($\sigma,\ \mu,\ \mathsf{bsn},\ gsk$)=1 [Check VII].
   - Check that there is no TPM other than $\mathsf{tpm_i}$ with key gsk' registered in Members or DomainKeys such that identify($\sigma,\ \mu,\ \mathsf{bsn},\ gsk'$)=1 [Check VIII].
   - If $\mathsf{tpm_i}$ is honest, then store $\langle \sigma,\ \mu,\ \mathsf{tpm_i},\ \mathsf{bsn}\rangle$ in Signed and output (SIGNATURE, $\mathsf{sid},\ \mathsf{ssid},\ \sigma$) to $\mathsf{host_j}$.

VERIFY

- On input (VERIFY, $\mathsf{sid},\ \mu,\ \mathsf{bsn},\ \sigma,\ RL$), from a party V to check whether a given signature σ is a valid signature on a message μ with respect to a basename bsn and the revocation list RL, the ideal functionality does the following:
- Extract all pairs $(gsk_i,\ \mathsf{tpm_i})$ from the DomainKeys and Members, for which identify($\sigma,\ \mu,\ \mathsf{bsn},$ gsk)=1. Set $b = 0$ if any of the following holds:

  - More than one key gsk_i was found [Check IX].
  - I is honest and no pair (gsk_i, $\mathsf{tpm_i}$) was found [Check X].
  - An honest $\mathsf{tpm_i}$ was found, but no entry $\langle\star,\ \mu,\ \mathsf{tpm_i},\ \mathsf{bsn}\rangle$ was found in Signed [Check XI].
  - There is a key $gsk' \in RL$, such that identify($\sigma,\ \mu,\ \mathsf{bsn},\ gsk'$)=1 and no pair $(gsk,\ \mathsf{tpm_i})$ for an honest $\mathsf{tpm_i}$ was found [Check XII].

- If $b \neq 0$, set $b \leftarrow$ver$(\sigma,\ \mu,\ \mathsf{bsn})$ [Check XIII].
- Add $\langle\sigma,\ \mu,\ \mathsf{bsn},\ RL,\ b\rangle$ to VerResults, and output (VERIFIED, sid, b) to V.

LINK

On input (LINK, sid, σ_1, μ_1, σ_2, μ_2, bsn), with $\mathsf{bsn} \neq \perp$, from a party V to check if the two signatures stem from the same signer or not. The ideal functionality deals with the request as follows:

- If at least one of the signatures (σ_1, μ_1, bsn) or (σ_2, μ_2, bsn) is not valid (verified via the VERIFY interface with $RL \neq \emptyset$), output $\perp$ [Check XIV].
- For each gsk_i in Members and DomainKeys, compute $b_i \leftarrow$ identify(σ_1, μ_1, bsn, gsk_i) and b'_i= identify(σ_2, μ_2, bsn, gsk_i) then set:
  - $f \leftarrow 0$ if $b_i \neq b'_i$ for some i [Check XV].
  - $f \leftarrow 1$ if $b_i = b'_i = 1$ for some i [Check XVI].
- If f is not defined, set $f \leftarrow$link(σ_1, μ_1, σ_2, μ_2, bsn), then output (LINK, sid, f) to V.