PiQASO D3.1 PiQASO QR Crypto Primitives Design, Optimization and Acceleration

Grant Agreement	101190366
Project Title	Post-Quantum Cryptography As-a-Service for Common Transmission Systems and Infrastructures
Project acronym	PiQASO
Project Start Date	01 January 2025
Number of Deliverable	D3.1
Report Title	PiQASO QR Crypto Primitives Design, Optimization and Acceleration
Related Work Package	WP3
Related Task	T3.1 – T3.4
Lead Organisation	TAU
Submission Date	31.03.2026
Last Change Date	31.03.2026
Dissemination Level	Public

EUROPEAN PARTNERSHIP

Co-funded by the European Union

ECCC EUROPEAN CYBERSECURITY COMPETENCE CENTRE

Abstract

D3.1 is the first in a series of PiQASO deliverables dedicated to detailing the provision of the PiQASO PQC Ensemble. To this end, we first provide a high-level description of the architecture of the PiQASO Framework, including the provision of the PQC enablers through a PQC-as-a-Service modality. Next, we provide a detailed overview of the state-of-the-art in PQC algorithms investigated by PiQASO, focused on lattice-based constructions. We then describe the forward-secure cloud storage scheme of PiQASO based on Updatable Public Key Encryption. Afterwards, we describe the mathematical foundations behind the Functional Encryption capabilities of PiQASO, as well as ACE of SPADE (AoS), a novel lattice-based FE scheme based on the RLWE problem. Finally, the SW/HW co-design approach is detailed, focusing on the envisioned optimization and acceleration capabilities for operations identified as bottlenecks in lattice- and code-based schemes.

Keywords: Forward Security, Updatable Public Key Encryption, Functional Encryption, Learning With Errors, Algorithmic Optimization, HW-based Acceleration.

Authoring & approval

Authors of the document

Name (Organisation name)	Date
Dimitris Karras **(UBITECH)**	31/03/2026
Georgios Fotiadis **(UBITECH)**	30/03/2026
Stefanos Vasileiadis **(UBITECH)**	30/03/2026
Georgios Pekridis **(UBITECH)**	30/03/2026
Thanassis Giannetsos **(UBITECH)**	31/03/2026
Stylianos Kazazis **(QUBITECH)**	30/03/2026
Dimitrios Katsinis **(QUBITECH)**	30/03/2026
Georgios Pastras **(QUBITECH)**	30/03/2026
Eugenia-Niovi Sassalou **(QUBITECH)**	30/03/2026
Symeon Tsintzos **(QUBITECH)**	30/03/2026
Alexandros Tavernarakis **(QUBITECH)**	30/03/2026
Melina Hadjeres **(TAU)**	30/03/2026
Antonios Michalas **(TAU)**	30/03/2026
Camille Nuoskala **(TAU)**	30/03/2026
Hossein Abdinasibfar **(TAU)**	30/03/2026
Mark Manulis **(UniBwM)**	30/03/2026
Federico Valbusa **(UniBwM)**	30/03/2026
Daniel Slamanig **(UniBwM)**	30/03/2026
Charilaos Vakrinos **(BYTE)**	27/03/2026
Dimitris Dimitrakoudis **(BYTE)**	27/03/2026
Eleftherios Tsapralis **(BYTE)**	27/03/2026
Ilias Bouboussis **(BYTE)**	27/03/2026
Nikos Giannopoulos **(BYTE)**	27/03/2026
Kostas Palamaras **(BYTE)**	27/03/2026
Davide Ariu **(PLURIBUS)**	27/03/2026
Nicola Marcialis **(PLURIBUS)**	27/03/2026
Angelina Katsifaraki **(NCIS)**	31/03/2026
Olga Politi **(NCIS)**	31/03/2026
Maria Eleni Damkali **(NCIS)**	31/03/2026
Georgia Oikonomopoulou **(UNIS GR)**	31/03/2026
Angelina Broukou **(UNIS)**	31/03/2026
Dimitrios Karystinos **(UNIS)**	31/03/2026

EUROPEAN PARTNERSHIP
Co-funded by the European Union
ECCC

Reviewed by

Name (Organisation name)	Date
Nicola Marcialis **(PLURIBUS)**	31/03/2026
Savvoula Oikonomou **(UBITECH)**	31/03/2026

Approved for submission to -

Name (Organisation name)	Date
Savvoula Oikonomou **(UBITECH)**	31/03/2026

Document history

Version	Date	Contributor/Organisation	Additional information
0.1	26/01/2026	UBITECH, TAU	Table of Contents
0.2	06/02/2026	UBITECH, QUBITECH, BYTE, PLURIBUS, NCIS, TAU, UniBwM	Roadmap to PQC transition envisioned by PiQASO. Description of key enabling factors for PQC migration and alignment with European standards (Chapter 2)
0.3	24/02/2026	UBITECH, QUBITECH	Compilation of PQC State-of-the-Art for lattice-based crypto, KEMs, signatures, and integration into higher-end protocols (Chapter 4)
0.35	17/02/2026	UBITECH, QUBITECH	Definition of initial PiQASO threat model based on identified state-of-the-art (Chapter 4)
0.4	24/02/2026	UBITECH, QUBITECH, TAU, UniBwM	Description of core building blocks of PiQASO PQC Ensemble: SW library, Quantum-resistant TCB, PQC-as-a-Service (Chapter 3)
0.45	27/02/2026	UBITECH, QUBITECH, TAU	Description of core building blocks of PiQASO PQC Ensemble: Functional Encryption, Crypto Asset Inventorying (Chapter 3)
0.5	02/03/2026	UBITECH, UniBwM	Description of the Forward Security capabilities of PiQASO through Updatable Public Key Encryption (Chapter 5)
0.6	06/03/2026	UBITECH, TAU	Description of the preliminaries regarding Functional Encryption, the LWE problem, Homomorphic Encryption (Chapter 6)
0.65	10/03/2026	UBITECH , TAU	Description of the Functional Encryption capabilities of PiQASO and the ACE of SPADE scheme (Chapter 6)
0.7	13/03/2026	UBITECH	Description of the algorithmic optimizations and HW accelerations of PiQASO (Chapter 7)
0.8	18/03/2026	UBITECH	Finalization of the Introduction and Conclusions sections (Chapters 1, 8)
0.9	24/03/2026	UBITECH, PLURIBUS	Internal review
1.0	31/03/2026	UBITECH, TAU	Finalization of deliverable and submission

PiQASO

Post-Quantum Cryptography As-a-Service for Common Transmission Systems and Infrastructures

The project funded under Grant Agreement No. 101190366 is supported by the European Cybersecurity Competence Centre

Co-funded by the European Union

ECCC EUROPEAN CYBERSECURITY COMPETENCE CENTRE

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Cybersecurity Competence Centre. Neither the European Union nor the granting authority can be held responsible for them."

Executive Summary

Considering the advent of quantum computers, it is imperative to ensure that all existing information security systems are able to cope with attackers and malicious entities who possess quantum computing capabilities. However, one core challenge in this regard is that existing service graph chains (SGCs) may consist of heterogeneous devices with varying computational capabilities, including legacy devices with limited ability to support advanced crypto. In this regard, one core target of PiQASO is to *provide the tools required for the transition of such SGCs to the post-quantum era, through the provision of the required tools and security solutions for ensuring that all entities remain at a trustworthy level throughout their operational lifecycle*. It should be noted that such a transition is not a singular technical upgrade, but a **systematic transformation consisting of not only cryptographic algorithms, but also their integration into higher-end security protocols**, ensuring **security of stored data**, as well as **algorithmic accelerations and HW-based optimizations** to enhance applicability and performance in target infrastructures. To this end, this deliverable is dedicated to providing the basis for the development of the **PiQASO PQC Ensemble**, by establishing a common baseline to support the crypto design and implementation activities carried out throughout the project lifecycle. This will, in turn, enable the development of the testbed for evaluating the impact of all PQ crypto enablers in the context of the PiQASO use case demonstrators.

One important aspect of securing modern computing infrastructures is the use of cloud storage. Specifically, while such solutions offer scalability, availability, and cost efficiency, they introduce security challenges, such as long-term data confidentiality. In this regard, one core consideration explored entails the notion of **forward secrecy** (Chapter 5) for ensuring that the exposure of a secret key at a given point in time does not compromise the confidentiality of data protected in the past. To this end, PiQASO offers a forward-secure cloud storage scheme based on **Updatable Public Key Encryption (UPKE)**, which provides strong resilience against key exposure and insider threats, thus enhancing long-term data protection. As a result, since cloud-stored data often remains sensitive beyond the lifetime of any single cryptographic key, forward secrecy provides a solution for building trustworthy and robust cloud systems, culminating in the definition of the **Forward Secure Cloud Storage Scheme** of PiQASO.

Another functionality offered by the PiQASO PQC suite pertains to **Functional Encryption** (Chapter 6), which enables performing high-level operations over encrypted data, specifically numerical values. For instance, in a healthcare scenario, it is possible to perform a search operation over encrypted heart rate data in order to determine whether there is a value above a specific threshold, in order to identify whether the patient requires further medical attention. It should be noted that the **ACE of SPADE (AOS)** Functional Encryption scheme provided by PiQASO is based on the **Ring Learning With Errors (RLWE)** problem, for which a detailed analysis is provided. However, considering recent advancements in its mathematical analysis, future versions of the AOS scheme will investigate the use of ideal lattices and Module LWE constructions.

One core target of PiQASO is the ability to provide post-quantum cryptographic capabilities to the far edge, including embedded devices which may be subject to strict latency and memory constraints. To this end, PiQASO employs a hybrid **Software/Hardware co-design approach** (Chapter 7), based on

Co-funded by the European Union
ECCC

which algorithmic software optimizations aim to improve the performance of the cryptographic algorithms themselves, while hardware accelerations target the underlying mathematical operations that have been identified as bottlenecks. Particularly, with regards to the latter, PiQASO aims to accelerate two core arithmetic operations: (i) **NTT**, which refers to the polynomial multiplication which lies at the core of lattice-based constructions, and (ii) **Karatsuba**, which is the mathematical operation at the core of HQC constructions.

Compounding all the above, D3.1 is the first deliverable in a series of PiQASO deliverables aimed towards the design and implementation of the **PiQASO PQC Ensemble** (Chapter 3). Here, we define the roadmap of the related activities to be carried out as part of PiQASO (Chapter 2), and we set the foundation for the evaluation of the PiQASO offerings as part of the envisioned use cases, in order to evaluate their performance and practical applicability.

Co-funded by the European Union

Contents

Co-funded by the European Union
ECCC

Co-funded by the European Union
ECCC

Co-funded by the European Union

List of Figures

Co-funded by the European Union
ECCC

List of Tables

List of Abbreviations

Abbreviation	Translation
AB	Advisory Board
AoS	Ace of SPADE
ABE	Attribute-Based Encryption
AKE	Authenticated Key Exchange
AQAP	Allied Quality Assurance Publications
ARKG	Asynchronous Remote Key Generation
Adv	Adversary
B/FV	Brakerski/Fan-Vercauteren
CAI	Crypto Asset Inventory
CBOM	Cryptographic Bill Of Materials
CISA	Cybersecurity and Infrastructure Security Agency
CKKS	Cheon-Kim-Kim-Song
CM	Countermeasure
COP	Common Operational Picture
CRM	Continuous Risk Management
CU	Ciphertext Updatability
DL	Deliverable Leader
DO	Data Owners
DoA	Description of Application
DoS	Denial of Service
DT	Deliverable Team
EC/EDA	European Commission/European Defence Agency
ECC	Elliptic Curve Cryptography
ENISA	European Union Agency for Cybersecurity
ESPAS	European Strategy and Policy Analysis System
EUCC	European Union Cybersecurity Certification
FE	Functional Encryption
FHE	Fully Homomorphic Encryption
FIA	Fault Injection Attack
FIPS	Federal Information Processing Standards
FMN	Federated Mission Networking
FRI	Fast Reed-Solomon Interactive Oracle Proof of Proximity
GA	Grant Agreement
GDPR	General Data Protection Regulation
GIS	Geographic Information System
HE	Homomorphic Encryption
HQ	Headquarters

Co-funded by the European Union

HSM	Hardware Security Modules
IAM	Identity and Access Management
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
NID-CPA	Indistinguishability under Chosen-Plaintext Attack
IPR	Intellectual Property Rights
IPSEC	Internet Protocol Security
IR	Internal Reviewers
KAT	Known Answer Test
KC	Key Curator
KEK	Key Encryption Key
KEM	Key Encapsulation Mechanism
KPI	Key Performance Indicator
LACI	Lightweight Asymmetric Cryptography Implementations
LCA	Loosely Coupled Acceleration
LWE	Learning With Errors
MCFE	Multi-Client Functional Encryption
MitM	Man-in-the-Middle
MLWE	Module Learning With Errors
MVP	Minimum Viable Product
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NTT	Number Theoretic Transform
OEM	Original Equipment Manufacturer
PHE	Partially Homomorphic Encryption
PKI	Public Key Infrastructure
PPT	Probabilistic Polynomial Time
PQC	Post Quantum Cryptography
PQCaaS	Post Quantum Cryptography-as-a-Service
PK	Public Key
QLDPC	Quantum Low Density Parity Check
QR	Quantum Resistant
RLWE	Ring Learning With Errors
SCA	Side-Channel Attack
SHE	Somewhat Homomorphic Encryption
SK	Secret Key
SL	Subjective Logic
SOC	Security Operation Center
SSI	Self-Sovereign Identity
SSL	Secure Sockets Layer
TAF	Trust Assessment Framework
TCA	Tightly Coupled Acceleration
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TFHE	Torus FHE
TLS	Transport Layer Security
TOC	Tactical Operation Center

U	Users
UAV	Unmanned Aerial Vehicle
UC	Universal Composability
UPKE	Updatable Key Encryption
VoIP	Voice over IP
VPN	Virtual Private Network
ZKP	Zero-Knowledge Proof
ZTA	Zero Trust Architecture

Chapter 1

Introduction

1.1 Scope and Purpose

Cryptographic services constitute the backbone of trust in modern service graph chain infrastructures, enabling secure communications, authenticated command and control, controlled information sharing, and trustworthy platform operation across heterogeneous systems ranging from embedded endpoints to backend infrastructures. The advent of quantum computing challenges this trust model by undermining the hardness assumptions of widely deployed public-key cryptography and, consequently, threatening critical defence capabilities that depend on PKI-driven authentication, secure channel establishment, and credential lifecycle management. For modern service graph chains, this challenge is compounded by long platform lifecycles, constrained and intermittently connected tactical nodes, stringent assurance expectations, and adversarial capabilities that extend beyond remote cyber attacks to include physical access, side-channel exploitation, and fault injection.

In this context, the scope of D3.1 is to establish a defence-grounded baseline for PiQASO. The deliverable pursues four objectives. First, it frames the necessity of PQC migration for defence infrastructures and positions the transition as a system-level transformation affecting protocols, PKI ecosystems, and deployability constraints. Second, it provides a high-level description of the building blocks comprising the PiQASO framework. This will, in turn, set the scene for the development of the tesbed for the evaluation of the PiQASO security enablers in the context of all use cases, in order to evaluate their impact on regular operations. Third, it provides a detailed description on the mathematical principles behind the functionalities provided by PiQASO, namely **Forward Security** through **Updatable Public Key Encryption (UPKE)**, **Functional Encryption**, decompressing **Learning-with-Errors (LWE)** with Polynomial Matrices, and the **Algorithmic Optimization and HW-based Acceleration** capabilities of PiQASO.

Thus, compounding all the above, D3.1 is the first deliverable in a series of technical PiQASO deliverables focused on the provision of the **PiQASO PQC Ensemble**, offering the functionalities and primitives to be tested in the context of the use cases through the design and implementation of an appropriate testbed. To this end, in this deliverable we first provide a **high-level overview of the architecture of the PiQASO PQC Ensemble**, starting from a description of all considered offerings, going all the way to providing them through a **PQC-as-a-Service (PQCaaS)** modality. Note that *all PiQASO offerings are designed to enable the PQC transition of all layers in the compute continuum, ranging from the far-edge (through the provision of an SDK and a QR-TCB for converting devices to their quantum-secure equivalents) to the cloud-based backend (by supporting advanced functionalities such as Functional Encryption and UPKE for secure and privacy-preserving data sharing)*. On top of that, this baseline of primitives will be integrated into **high-end security protocols** for establishing authenticated and encrypted communication sessions.

Co-funded by the European Union
ECCC

1.2 Relation to other WPs and Deliverables

As previously outlined, one core target of D3.1 is to provide a high-level overview of the PiQASO PQC Ensemble, covering all offered modalities and all layers of the compute continuum. This provides the basis for the definition of the PiQASO reference architecture (D2.1), which will consist of a structured representation of the required security functionalities for the safety-critical services of the PiQASO use cases towards defining a layered architecture for enabling the transition of legacy systems to next-generation quantum-secure systems. D2.1 will also provide a more concrete version of the threat model initially defined in this deliverable, and will define the long-term security requirements and functional specifications for the PiQASO use cases. D3.1 also provides the basis for the design and implementation of the PiQASO SDK in D3.2, as the first core block of the PiQASO PQC ensemble containing the optimized SW-based implementations of all enhanced PQ cryptosystems, as well as optimized parametrizations for all use cases, and the first definition of the PQCaaS modality for providing the PiQASO offerings to organizations belonging to various application domains.

D3.1 also provides input to other PiQASO WPs, and sets the foundation for the integration activities to be carried out for the PiQASO PQC Ensemble in the context of all use case demonstrators, in WP2, thus setting the scene for the evaluation of all PiQASO modalities in all use cases through the creation of the required testbed, and the evaluation of their impact on the normal operational behaviour of the use case environments in WP4. The outcome of the design and implementation activities of PiQASO provides input to WP5, where the dissemination of PQC knowledge and training of all concerned stakeholders will be facilitated through the construction of the PiQASO Academy. Finally, in the context of WP6, dissemination activities will be carried out for facilitating the adoption of all PiQASO offerings in actual, real-world organizations, and detailed exploitation plans for PiQASO artefacts will be set forth.

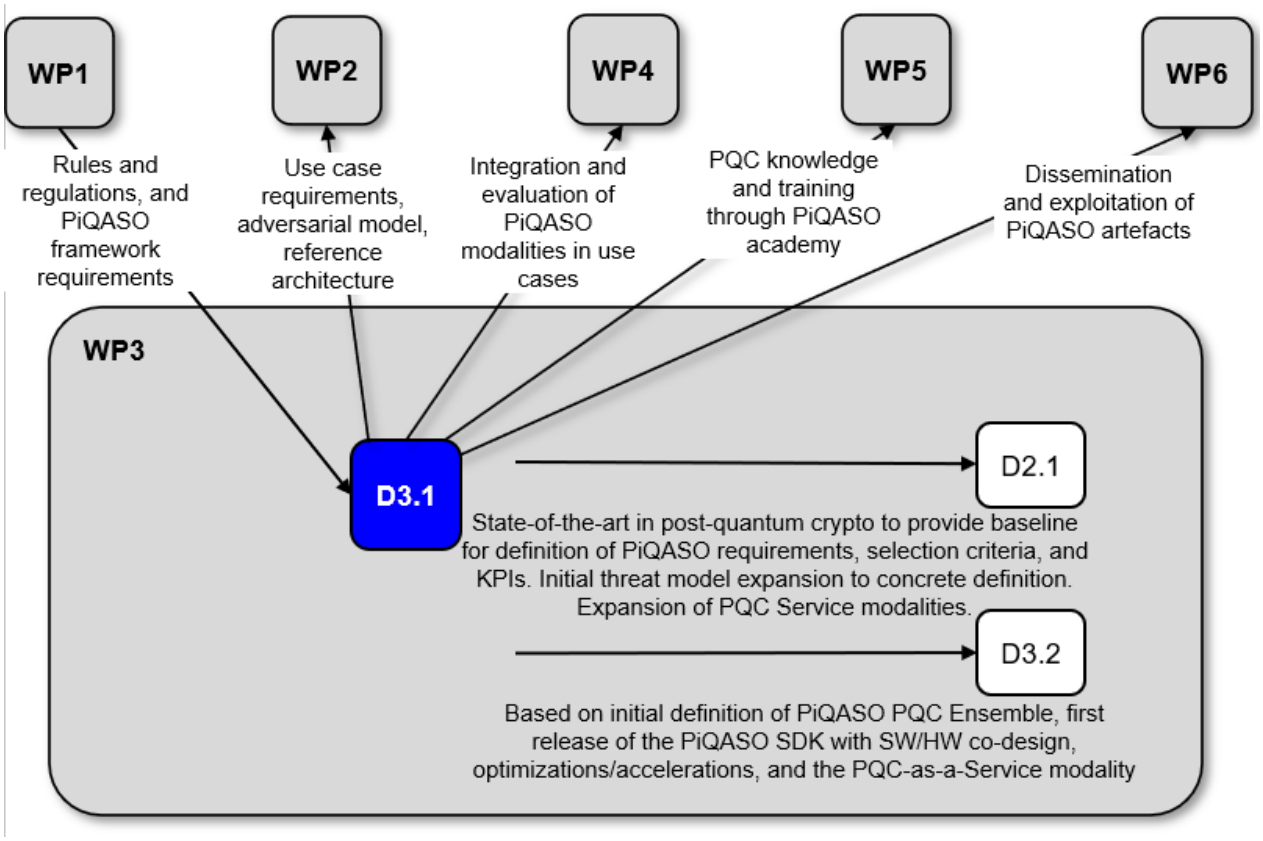

Figure 1.1: Relation of D3.1 to other PiQASO WPs and Deliverables

1.3 Deliverable Structure

The remainder of this deliverable is structured as follows:

Chapter 2 provides a detailed roadmap towards the development of the PiQASO PQC ensemble, while detailing the motivation behind the selection of PQC primitives.

Chapter 3 provides a high-level overview of all building blocks comprising the PiQASO PQC ensemble, as well as their positioning within the PiQASO framework.

Chapter 4 details the state-of-the-art in post-quantum cryptography, focusing on the aspects tacked by PiQASO.

Chapter 5 provides a description of the Forward Security functionalities of PiQASO, focusing on the Updatable Public Key Encryption (UPKE) functionality.

Chapter 6 focuses on the Functional Encryption capabilities of PiQASO, including Homomorphic Encryption and the ACE of SPADE (AoS) scheme.

Chapter 7 presents the algorithmic optimization and HW-based acceleration features of PiQASO.

Chapter 8 concludes the deliverable.

Chapter 2

PiQASO Ensemble PQC Roadmap

The security posture of modern service graph chains relies on the cryptographic foundations that enable secure communication, trusted execution, authenticated command exchange, and resilient information sharing. From edge devices and embedded systems to federated platforms and cross-domain gateways, cryptography constitutes the backbone of trust in network infrastructures belonging to various domain applications.

However, the advent of quantum computing introduces a structural disruption to this trust model. Cryptographic primitives that have underpinned security for decades—primarily those based on integer factorisation and discrete logarithm problems—are vulnerable to polynomial-time quantum algorithms. While large-scale quantum computers are not yet operationally available, the trajectory of technological progress, combined with the long lifecycle of existing infrastructures and the handling of sensitive information, necessitates immediate action.

Transitioning towards quantum-resistive networks is not a singular technical upgrade but a systemic transformation. It affects not only cryptographic algorithms but also architectural design choices, hardware-based Roots of Trust, protocol stacks, key management infrastructures, certification pipelines, and operational interoperability across allied ecosystems. Moreover, the diversity of PQC families and their heterogeneous performance and security characteristics introduce new complexity in selecting and integrating appropriate primitives into critical environments.

In this context, the objective of this chapter is to establish the foundations of PiQASO's structured and resilient PQC transition approach. Section 2.1 analyses the necessity for migration, highlighting the convergence of quantum threat acceleration and algorithmic trade-offs. Section 2.2 identifies the key enabling factors required to operationalise migration, including visibility of cryptographic assets, agility in cryptographic deployment, and elevation of trust anchors. Finally, Section 2.3 examines alignment with European standardisation efforts and identifies gaps in profiling and metrics construction.

Together, these sections articulate PiQASO's coherent pathway toward quantum-resistive networks—networks capable of preserving long-term confidentiality, authenticity, integrity, and operational continuity in the emerging post-quantum era.

2.1 Need for PQC Migration

The security of contemporary service graph chains is fundamentally anchored in asymmetric cryptographic primitives such as RSA and elliptic-curve cryptography (ECC). These schemes underpin authentication, digital signatures, key exchange protocols, Public Key Infrastructures (PKIs), and communication stacks. Their long-standing security assumptions are based on the computational hardness of integer

factorisation and discrete logarithm problems. The rapid progress of quantum computing, however, challenges these foundational assumptions. Advances in logical qubit stability, quantum error correction, and algorithmic optimisation indicate that the time horizon for cryptographically relevant quantum computers may be shorter than previously estimated. While large-scale quantum machines capable of breaking RSA-2048 or ECC are not yet available, adversaries do not need immediate decryption capabilities to pose a threat.

The “harvest-now, decrypt-later” paradigm fundamentally alters the risk model for modern infrastructures. Sensitive diplomatic, operational, intelligence, and industrial data intercepted today may remain confidential for decades. If stored encrypted under classical public-key schemes, such data becomes vulnerable to retrospective decryption once sufficiently powerful quantum computers emerge. As such, migration planning must begin well before quantum advantage is practically demonstrated. Therefore, PQC migration is not a speculative research exercise but a forward-looking resilience strategy aimed at preserving long-term confidentiality, authenticity, and integrity of communications and systems.

2.1.1 Post-Quantum Transition Timeline

The urgency of this transformation is underscored by the European Union’s roadmap, which mandates concrete transition strategies by the end of 2026, migration of critical sectors to PQC by 2030, and targets full PQC migration by 2035 [249]. Meeting these deadlines requires immediate action to hedge the scale of impact, particularly regarding the miniaturization of PQC footprints to enable integration across diverse devices.

While this timeline may appear proactive, the rapid pace of quantum computing development suggests it is merely necessary. Commercial vendors such as IonQ now project the delivery of systems capable of supporting 80,000 logical (2,000,000 physical) qubits by 2030 [175]. Furthermore, recent theoretical breakthroughs have drastically lowered the hardware threshold for cryptanalysis: new architectures utilizing Quantum Low Density Parity Check (QLDPC) codes suggest that RSA-2048 could be broken with as few as 100,000 physical qubits—an order of magnitude fewer than previously estimated [309]. These developments highlight that the ‘Q-day’ threat is credible and may arrive sooner than anticipated due to sudden leaps in both instrumentation and error-correction algorithms.

2.1.2 The Transition Roadmap

A transition roadmap for PQC must be engineered as a long-term resilience programme rather than a single replacement action. NIST [230], [46] and ANSSI [8] emphasise that cryptographic transitions are recurring events and that systems must be prepared to replace and adapt cryptographic algorithms, parameters, and protocol instantiations across software, hardware, firmware, and infrastructures without disrupting normal operations. Consequently, a credible roadmap is anchored on three mutually reinforcing pillars: (1) **crypto agility**, (2) **hybridisation** during the transition period, and (3) **interoperability with legacy infrastructures**. While the standardization of PQC primitives has reached a high level of maturity, the industry currently faces the mandate of developing hybrid cryptosystems that combine established classical schemes with quantum-resistant constructions. This hybridization is essential for avoiding security regression during the transition period and is required to transform high security protocols into a Post Quantum Public Key Infrastructure (PKI) capable of securing the entire compute continuum from far edge to cloud.

2.1.2.1 *Crypto Agility*

Crypto agility constitutes the cornerstone of the roadmap because it provides the capability to adapt under uncertainty. Previous cryptographic migrations (e.g., DES to AES, SHA-1 to SHA-2) were frequently reactive and operationally costly, often due to hard-coded dependencies, rigid protocol bindings, and inflexible hardware implementations. The PQC transition amplifies these difficulties: *multiple mathematical families (e.g., lattice-based, hash-based, code-based) exhibit heterogeneous operational trade-offs, and the long lifecycles of service graph chain mean that today's cryptographic design choices may remain operational for decades.* NIST therefore highlights the need to move from ad hoc remediation toward planned agility [46], in which systems are designed natively to accommodate future cryptographic substitutions, a concept that directly supports the operational superiority and longevity of critical assets.

In this context, cryptographic agility should be understood as a system-level architectural property rather than a software convenience. As defined in NIST Cybersecurity White Paper 39 [46], crypto agility encompasses the capabilities required to replace and adapt cryptographic algorithms, parameters, and protocols across software, hardware, and infrastructure without disrupting critical operations. This capability is necessary for two complementary reasons. First, it enables **remediation** if a deployed PQC primitive is later found vulnerable, deprecated, or no longer recommended. Second, it enables **optimisation** and tailoring by allowing systems to select primitives and parameter sets that best satisfy application-driven constraints (e.g., bandwidth and footprint versus computational cost, verification throughput versus signature size), while remaining aligned with assurance and interoperability requirements.

Achieving crypto agility requires the systematic decoupling of applications from cryptographic mechanisms through stable interfaces, controlled configuration, and policy-governed selection. In practice, applications should request security services (e.g., key establishment, signing, verification) without being tightly bound to a specific primitive, so that algorithm substitution can be enacted as a controlled operational change rather than an intrusive redesign [46]. This separation is particularly important in modern service graph chains because cryptography is deeply embedded across layers—secure boot chains, platform identity, secure communications, and PKI-driven authentication—and any tightly coupled dependency can become a migration bottleneck.

Crypto agility must also extend beyond software-only considerations, since embedded systems often depend on hardware-backed security and acceleration. While software abstraction improves adaptability, rigid single-purpose accelerators or non-updatable firmware can constrain future transitions. A roadmap that operationalises agility therefore favours designs that support re-parameterisation and reuse of common cryptographic kernels where feasible, so that cryptographic capabilities can evolve through controlled firmware and configuration updates rather than hardware replacement, consistent with NIST's system-wide interpretation of agility [46]. At the same time, agility depends on operational visibility: migration cannot be systematically executed without knowing where cryptography is used and how it is instantiated. For this reason, crypto agility is closely coupled with crypto asset inventorying (treated as a dedicated enabling factor elsewhere in this chapter), which provides the situational awareness needed to apply agility in a policy-driven and risk-informed manner.

For the most resource-constrained nodes where fully on-device agility is infeasible, agility can also be realised through architectural abstraction, including service-based models that offload certain cryptographic functions to secure and controllable layers (e.g., edge or back-end security services). Such approaches can decouple the security lifecycle from the device lifecycle, thereby enabling field-deployed assets to consume updated cryptographic capabilities without requiring disruptive hardware replacement, provided that the trust, latency, and availability constraints are preserved.

Finally, the rationale for agility is strengthened by the existence of multiple PQC primitives capable of fulfilling the same cryptographic role, each with distinct benefits and limitations. Figure 2.1 illustrates these trade-offs across PQC families and motivates the need for planned adaptability rather than single-

choice commitments. In particular, NIST-standardised (or draft-standard) primitives include ML-KEM (Kyber) [237] and KEM-HQC (HQC) [244] for key establishment, as well as ML-DSA (Dilithium) [235], SLH-DSA (SPHINCS+) [240], and FN-DSA (FALCON) [243] for digital signatures. The diversity of these options—combined with evolving profiling guidance and deployment constraints—reinforces the central roadmap principle: service graph chains must be engineered to adapt cryptographic choices over time without jeopardising continuity or assurance.

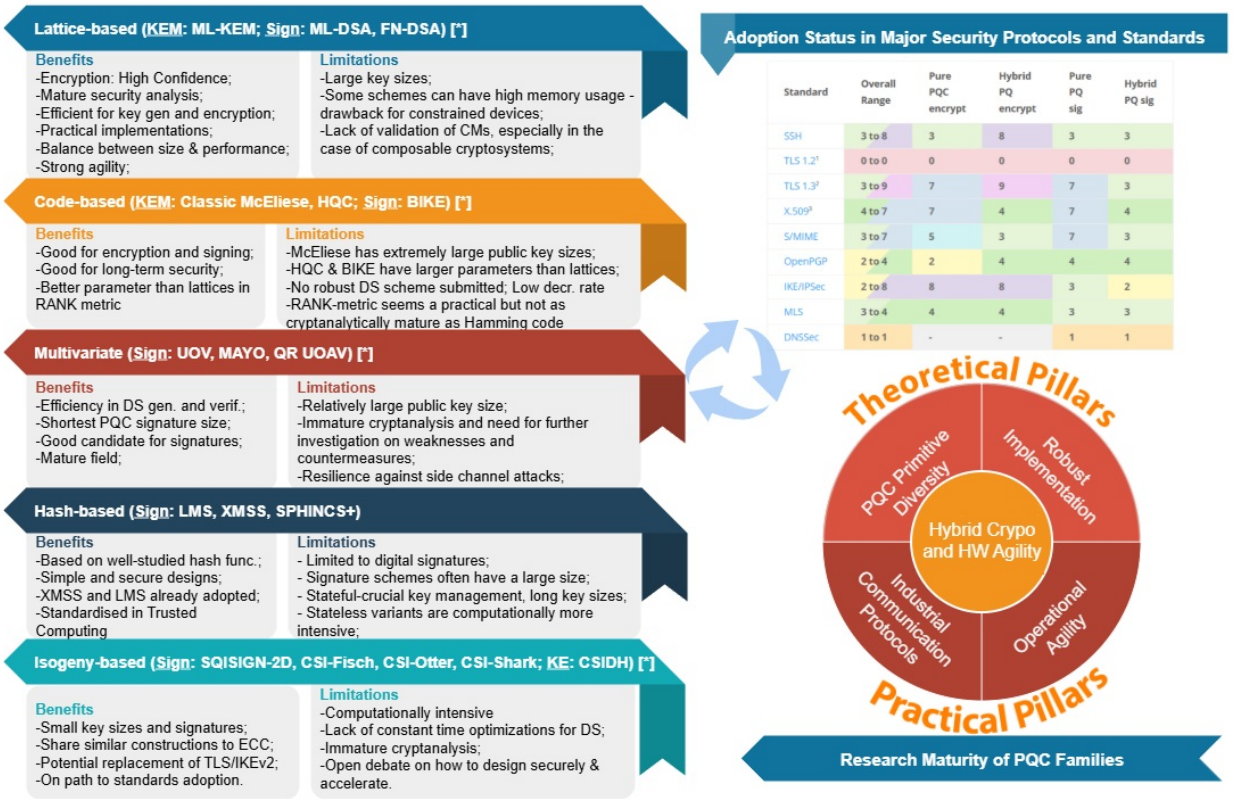

Figure 2.1: Benefits and limitations of various families of PQC primitives.

2.1.2.2 *Hybridisation*

Hybridisation is the second pillar of the transition roadmap because it provides a disciplined mechanism to preserve security guarantees and operational interoperability during the multi-year period in which PQC-enabled and legacy components must inevitably coexist. Unlike previous cryptographic updates, where a single dominant primitive could be phased out with limited ecosystem disruption, the PQC transition impacts the full PKI and protocol fabric across heterogeneous endpoints, administrative domains, and long-lived platforms. The IETF's engineering guidance stresses that, under incremental deployment, protocol and infrastructure adjustments are unavoidable because communicating peers will not be upgraded concurrently and because negotiation and failure behaviour must remain deterministic and downgrade-resistant [43].

At a high level, hybrid mechanisms combine a traditional public-key algorithm with a post-quantum algorithm so that the resulting security does not depend exclusively on a single cryptographic family during the migration period. The security rationale is that if one component is later found weaker than expected—whether due to cryptanalytic advances, parameter weaknesses, or implementation vulnerabilities—the other component can preserve essential properties such as the confidentiality of established session keys or the authenticity of credentials. ANSSI explicitly frames early adoption through hybrid approaches as a pragmatic strategy for long-term protection horizons and recommends hybridisation as a

hedge while the confidence level in post-quantum schemes and deployment profiles continues to mature [8].

To avoid inconsistent interpretations across systems, hybridisation must be specified using standard terminology and explicit construction semantics. RFC 9794 defines security-relevant terminology for post-quantum/traditional hybrid schemes and is intended to ensure consistent interpretation across protocols, standards, and organisations [114]. Within this terminology, the distinction between composite and non-composite hybrid protocol designs is particularly important, as it determines where the primary changes occur—within cryptographic elements themselves or within protocol fields and message flow.

In IETF terminology, a **composite** PQ/T hybrid protocol incorporates hybrid schemes in a way that keeps protocol fields and message flow essentially the same as in a single-algorithm version; the main changes occur in the formats of the cryptographic elements (e.g., keys, signatures, ciphertexts), which encapsulate multiple component algorithms behind a single composite interface [114]. By contrast, a **non-composite** PQ/T hybrid protocol composes multiple single-algorithm schemes so that the component cryptographic elements retain their single-algorithm formats; in this case, the main changes occur in the protocol fields and/or message flow, while changes to cryptographic elements are minimised, and most implementation modifications tend to fall within protocol libraries rather than cryptographic libraries [43]. Importantly, a protocol need not be purely composite or purely non-composite; it may combine both approaches (for example, composite key establishment with non-composite authentication), provided that negotiation and downgrade protections are engineered explicitly [114, 43].

For authentication and PKI migration specifically, recent IETF guidance compares alternative transition models—including composite certificates, dual-certificate approaches, and PQC-only certificates—and emphasises that the selection impacts operational deployment complexity, relying-party compatibility, and lifecycle management (issuance, rotation, revocation, and auditability) [270]. Therefore, the choice of hybrid strategy must be driven by operational constraints and assurance expectations: modern service graph chains may prioritise bounded message sizes and predictable negotiation under constrained links, whereas modern service graph chain infrastructures may prioritise auditability, deterministic validation semantics, and long-term maintainability of credential chains. Irrespective of the selected pattern, hybridisation must be treated as a governed transition mechanism rather than an ad hoc compatibility patch, because inconsistent hybrid handling across domains directly translates into risk and fragmented trust.

2.1.2.3 *Interoperability with Legacy Infrastructures*

Interoperability and integration with legacy infrastructures constitute the third pillar because it ultimately determines whether PQC can be deployed at scale without degrading operational capability. In modern service graph chains, cryptographic mechanisms are deeply embedded across system layers and platform categories, including secure boot and firmware validation, platform identity and attestation, secure communications, cross-domain gateways, and PKI-enabled authentication for critical applications. Many of these components are long-lived, certification-constrained, and not uniformly updatable; as a result, migration must preserve compatibility across mixed capability sets over extended periods. The IETF's guidance for engineers emphasises that PQC transition must explicitly account for incremental roll-out, heterogeneous endpoint capabilities, protocol negotiation behaviours, and the operational risks of partial compatibility, including downgrade resistance and deterministic selection of mutually supported mechanisms [43].

A central interoperability constraint is the PQC footprint: several PQC mechanisms entail larger public keys, signatures, certificates, and handshake payloads compared to legacy RSA/ECC deployments. These increases can produce first-order operational effects in bandwidth-constrained and latency-sensitive environments, especially in communication links, embedded systems, and far-edge deployments. More-

over, the computational and memory profiles of PQC primitives vary significantly across families and parameter sets, affecting energy consumption, real-time responsiveness, and throughput on constrained processors. Consequently, interoperability cannot be treated as a "final integration step"; it must be engineered from the outset of the roadmap through explicit profiling of cryptographic choices against application constraints and platform classes. NIST's transition guidance similarly stresses that the move to PQC is a multi-stage deployment challenge across products and infrastructures, requiring coordinated planning and integration across systems rather than isolated algorithm substitution [230].

Interoperability also has a strong assurance dimension. Legacy systems frequently rely on certified cryptographic modules and established validation artefacts; introducing PQC affects not only the cryptographic primitive but also the surrounding protocol logic, credential formats, and lifecycle processes (issuance, rotation, revocation, logging). Any ambiguity in negotiation and validation semantics can create security regressions even when "strong algorithms" are selected, particularly if fallback behaviours enable downgrade or inconsistent policy enforcement across domains. For this reason, interoperability must be aligned with governance and policy mechanisms that ensure consistent algorithm selection, parameter constraints, and transition rules, in line with the broader NIST framing of planned, system-wide agility [46].

Finally, interoperability with legacy infrastructures is inseparable from performance engineering. In practice, deployability at scale will depend on the ability to keep PQC overheads within acceptable operational envelopes across the compute continuum. This motivates systematic optimisation and, where appropriate, acceleration strategies to reduce communication expansion and computational cost, especially for certificate-heavy workflows and frequently executed authentication operations. Such engineering is not a "nice to have": uncontrolled footprint growth can directly impact communications, increase latency at critical decision points, or constrain deployment on embedded platforms. Therefore, the roadmap must treat integration engineering—protocol profiling, footprint management, and implementation optimisation—as a necessary condition for achieving quantum-resistive security without compromising continuity or fragmenting interoperability across legacy-dependent systems [43].

2.1.2.4 *PQC Families Availability, Trade-offs, and PiQASO Direction towards providing a PKI-equivalent capability*

The three transition pillars identified in this chapter—crypto agility, hybridisation, and interoperability with legacy infrastructures—are actionable only if credible post-quantum alternatives already exist for the two core required public-key roles: authenticated key establishment and digital signatures. This condition is now met. Figure 2.1 reflects that multiple PQC families can fulfil these roles, but with distinct operational properties and maturity levels, implying that algorithm choice is simultaneously a resilience enabler (via diversity) and an engineering constraint (via footprint and integration trade-offs). NIST's transition guidance further emphasises that adoption must be planned as a staged programme across products, services, and infrastructures rather than as a one-time replacement, precisely because cryptographic dependencies are pervasive and algorithm agility and coexistence are required during migration [230, 45]. In parallel, ANSSI explicitly stresses that hybridisation remains necessary in the short and medium term because post-quantum algorithms should not be assumed sufficiently mature to be deployed alone for all use cases, and because hybrid approaches provide a pragmatic hedge during the confidence-building period [34].

At the family level, Figure 2.1 summarises the principal benefits and limitations that drive transition engineering. Lattice-based schemes have become the dominant near-term option for broad deployment due to their favourable performance and extensive public scrutiny; this is reflected in NIST's first finalised PQC standards, where ML-KEM (FIPS 203) is specified for key establishment and ML-DSA (FIPS 204) is specified for signatures [238, 236]. Nevertheless, lattice mechanisms introduce larger keys/certificates and

non-trivial implementation constraints compared to classical ECC, and they require careful engineering and hardening to ensure robust operational deployment under realistic adversarial models. Hash-based signatures provide conservative security foundations grounded in well-studied hash functions and simple design principles; NIST's SLH-DSA (FIPS 205) specifies a stateless hash-based signature standard derived from SPHINCS+ [240]. At the same time, hash-based signatures typically impose larger signature sizes and/or computational overheads, and, for stateful variants (e.g., XMSS and LMS), operational fragility arises from state management requirements, motivating careful profiling and deployment discipline [240, 170, 223]. Code-based schemes provide valuable algorithmic diversity and strong security intuition, but practical deployment is often constrained by large public keys and associated communication overheads, which can be problematic for certificate-heavy workflows and bandwidth-limited environments; these trade-offs are reflected in NIST's broader PQC evaluation and status reporting [16]. Finally, additional families (e.g., multivariate and isogeny-based candidates) contribute diversity and, in some instances, attractive size properties, but they are generally less stabilised for near-term high-assurance deployment, reinforcing the need for staged adoption and agility-driven hedging [16, 34].

These family-level trade-offs translate directly into the protocol ecosystem readiness illustrated in Figure 2.1: progress across major security protocols and standards is uneven, and migration commonly relies on transitional and hybrid constructions. The IETF engineering guidance stresses that PQC transition differs from previous cryptographic updates because incremental deployment, negotiation behaviour, downgrade resistance, and ecosystem dependencies must be explicitly engineered [43]. Moreover, the IETF has standardised terminology for hybrid schemes to ensure consistent interpretation across protocols and organisations [96]. In practical terms, this means that PQC adoption must be executed at the protocol and PKI level, not merely at the primitive level: modern service graph chains require a post-quantum functional equivalent of classical PKI capabilities—supporting authenticated key establishment, certificate-driven authentication, and integrity protection—while preserving interoperability across mixed fleets spanning embedded nodes to back-end systems [230, 43].

In this context, **PiQASO prioritises lattice-based and hash-based cryptography**, as dictated by requirements for deployability, maturity, and achievable assurance under constrained operational conditions. Lattice-based primitives provide the most practical basis for scalable key establishment and signature services under current standardisation maturity [238, 236], while hash-based signatures complement this baseline with conservative security foundations that are particularly relevant for long-term integrity assurances and robustness-driven profiles [240]. To realise the transition pillars end-to-end, these primitives must be integrated into high-level security protocols and credential ecosystems. Representative IETF work items specify how ML-KEM and ML-DSA are incorporated into TLS 1.3 as part of the ongoing protocol-level transition [90, 167], while PKI-level migration guidance addresses the operational trade-offs among composite, dual, and PQC authentication models in certificate-based systems [272]. Together, these protocol- and PKI-level integrations provide the engineering pathway toward a quantum-resistive trust fabric that preserves operational continuity through hybrid coexistence, supports future remediation via agility, and maintains interoperability with legacy infrastructures.

Table 2.1: Post-Quantum crypto implementations and countermeasures

PQC Family	Implementations
ML-DSA	• **RISC-V-based**: matrix extension, 2D systolic array for NTT and other arithmetic optimizations [322] , PQ ALU with performance/energy optimization [231], Custom lattice crypto processor (Sapphire) for energy/performance optimization [44] • **ASIC-based**: ASIC-tailored KaLi polynomial arithmetic unit with memory management for optimal latency, Xilinx Zynq Ultrascale+ZVU102 FPGA [13], C-based FPGA and ASIC implementation for ML-DSA and ML-FSA [295] • **FPGA-based**: Ultrascale+ with parallelism, pre-computation, memory access sharing [157], Ultrascale+ with compact programmable coprocessor for key generation, encapsulation, decapsulation, signature creation and verification, Zynq-7000 SoC, custom HW for main algorithm, pre/post-processing on processor [308], Offloading to FPGAs to reduce energy consumption [52], Artix-7 with LUT and FF optimization with special purpose DSPs [197], two-stage pipeline for signing with low latency/ area [53] • **ARM-based**: Signing with trade-offs on stack/flash memory, speed [154], Intel Core i5-2260, Cortex-M4 unrolling, inlining-pre-signing [268], NTT optimization, uniform sampling [158], Cortex M7, leveraging 64-bit (double-precision) FP architecture [168], NTT/arithmetic signing optimizations, smaller prime modulus [3], ARMv8 (NEON) for optimized, interleaved NTT [193], Co-design for reducing data transmission overhead, with HW accelerator integrated with ARM Cortex-A9 in Xilinx Zynq [220], C implementation on Nios-II, speed/resource trade-off [324]. • **OpenTitan-based**: Extensions for optimizing lattice-based crypto with OpenTitan Big Number (OTBN) accelerator for improved performance [4], instruction set extensions in OTBN for polynomial arithmetic and sampling optimizations [299] • **Other HW optimizations**: Segmented pipeline NTT processing for storage/performance optimization [321], Performance, latency, area, energy optimization for embedded applications [51], VHDL parallelization with low cycles, reasonable HW resources [278], RISC-V co-design with enhanced performance using HW accelerations [190], Improved NTT HW architecture, implemented using HLS, memory write-back [245] • **With Countermeasures**: Low-cost fault-attack-resistant on Vehicle Network Processor S32G274A: CM: FIA [62], ARM Cortex-M3, CM: masking with power of two prime modulus for side-channel leakage [228], ML-DSA computations to be protected by Differential and Simple Power Analysis (DPA, SPA), masking, shuffling with performance/security trade-off [40]. SCA on signing vector. CM: Shuffling and masking [56]. SCA and FIA combination resilient to shuffling. CM: Masking [69], AI-based SCA on secret key unpacking. CM: Masking secret signing key [307], Instruction skipping FIA, deterministic and random ML-DSA. CMs: Signature re-computation, fault detection [123] Generalizes attack with fault in signing (randomized ML-DSA). CM: Sign-then-verify [195].

ML-KEM	• **Reference implementation**: Two-stage approach: (i) IND-CPA-secure PK encryption, (ii) FO transform for constructing KEM [37] • **ARM-based**: Cortex-M4 with signed Barrett performance optimization [3], 64-bit Cortex-A, NTT optimizations, noise sampling, AES-based accelerator for symmetric functions [283], Cortex-A72 and Apple M1 with caching incomplete NTT results, efficient modular multiplication with Armv8-A Neon vector instructions [49] • **With countermeasures**: ARM Cortex-M4, PQM4 with performance optimizations. CMs: 1st/higher order masking [164], ARM Cortex-M0+/M4F. CM: Masking against first- and higher-order attacks [64]. Artix-7, area-performance trade-offs. CM: Fault detection hashes, instruction cycle count, equality instructions, direct memory access isolation, protection against control flow modification, memory faults. Side-channel: address and instruction randomization [180], Virtex7 VC707 FPGA fully masked implementation for SCA protection with limited performance impact, CM: Masking and hiding through pipelining operations [185]. Masked co-design with ciphertext compression, NTT multiplier for optimization, CM: Masking [133]. High-order masking and polynomial comparison. New high-efficiency techniques proposed [91]. Improved FIA with reduced requirements for success, SACSA and lattice reduction [165], CPA targeting polynomial multiplication. CMs: Masking, hiding [196], Kyberslash, timing attack with non-constant-time modular division: CM: constant-time implementation [54], Defeating low-cost CMs, like sanity check and decapsulation failure check. No CM proposed [269]
FN-DSA	• **RISC-V-based**: Co-design with enhanced performance using HW accelerations [251] • **ARM-based**: Faster signature verification using ARMv8-A, benchmarked on Apple M1 and RPI4 Cortex-A72. Slower signing compared to ML-DSA [246], NEON parallel processing unit of ARMv8 MCU for accelerating FTT and NTT [192]. • **SW-HW co-design**: Efficient accelerator (EFX) with granular optimization of low-level operations [202], Xilinx SoC FPGA, co-design for accelerating FN-DSA Gaussian sampling subroutine [188] • **With Countermeasures**: Mitigations for leakage of floating point-based operations [77], [189]. Key recovery by faulting Gaussian sampling, reducing lattice dimension. CM: Zero-checking sampled secrets [222], Key recovery with a timing attack [132], EM-based SCA on pointwise multiplication of FFT. CM: Masking [187], [155], [319]
SLH-DSA	• **ARM-based**: Architecture optimizations (Cortex): (i) Acceleration of Keccak permutation, (ii) hybrid implementations leveraging scalar and Neon instruction sets of Aarch64 [50], TPM extensions for resource-constrained devices to sign messages [247] • **Intel-based**: Optimized library combining SHA-NI and AVX2 vector instructions [161] • **FPGA-based**: Area-efficiency for embedded systems [55], Performance-optimized, SCA identified (no efficient CMs proposed) [31], HW accelerator for signing [30] • **RISC-V-based**: Prototype implementation SLotH with HW optimizing padding formats and iterative hashing processes of SLH-DSA. CM: SCA-secure PRF computation and Winternitz chains [282], integrating two accelerator modules for Keccak-1600 and Haraka hash function, with processor modifications [211] • **Other HW optimizations**: Secure boot optimization, exploiting similarities of SLH-DSA with LMS/XMSS [306], First FPGA and ASIC implementation of HQC [287] • **With Countermeasures**: Protection against fault injection for forging signatures. Recommended redundancy checks, caching the intermediate W-OTS+s [140]. Key recovery with faulty signatures obtaining by signing same message twice [73], [141], Extension to randomized SLH-DSA. CMs: redundancy, recompute and compare [140], SACSA-based attack on Keccak. CM: Shuffling Keccak operations [186]

HQC	• **FPGA-based**: Xilinx Artix 7 low-latency polynomial multiplication and efficient Reed-Solomon decoder [203], cross-layer approach for optimizing HQC RNG [288]. • **Other HW optimizations**: Optimized designs for key generation, encapsulation, and decapsulation [102], fast prototyping of HQC with high-level synthesis (HLS) with parallelization [12]. • **With countermeasures**: Masked specification-compliant HQC vector sampling to address timing attacks: CM: Masking [296]. Single-trace SASCA based on belief propagation (BP), targeting the Reed-Solomon decoder in the HQC code. CMs: Masking, shuffling. [153], Cache timing attack. CM: Accessing whole index set in secret sparse vector [169], Message recovery based on EM-leaks of load and store operations. CM: Shuffling multiplication order [152], Key recovery using ciphertexts combined with EM-based SCA. CM: Boolean masking [150], Key recovery using ciphertexts with timing-based attack. CM: Constant-time random vector generation [151].

2.2 Key Enabling Factors for PQC Migration

2.2.1 Crypto Asset Inventorying

The accelerated evolution of quantum computing constitutes a profound and disruptive transformation in the foundational security assumptions upon which contemporary cryptographic systems are constructed. It challenges long-standing trust models. It alters the threat landscape in fundamental ways.

Public-key cryptographic mechanisms such as RSA and elliptic curve cryptography (ECC) currently underpin secure communications, digital authentication frameworks, certificate infrastructures, and cryptographic key management across global digital ecosystems. These mechanisms are deeply embedded in modern digital operations. They serve as the backbone of confidentiality and trust online. However, they are widely anticipated to become fundamentally vulnerable once large-scale, fault-tolerant quantum computers materialize. This projected capability elevates quantum computing from a distant theoretical abstraction to an imminent inflection point. It demands proactive governance, structured preparedness, and anticipatory mitigation. Reactive remediation will no longer be sufficient.

A particularly consequential dimension of this emerging threat paradigm is encapsulated in the harvest-now-decrypt-later model. Within this framework, adversaries may already be systematically aggregating encrypted communications and sensitive datasets. They may be preserving them for future decryption. This decryption would become possible once quantum-enabled cryptanalysis becomes operationally viable. The implications of this strategy are especially acute for information characterized by long-term confidentiality horizons. This includes personal and biometric data. It also includes proprietary industrial research, classified governmental correspondence, and operational data governing critical infrastructure systems. Consequently, cryptographic resilience must no longer be evaluated solely through the lens of present-day computational feasibility. It must be assessed from a forward-looking perspective. That perspective must account for the projected longevity of both protected data assets and the cryptographic primitives safeguarding them.

Within this evolving context, crypto asset inventorying emerges as an indispensable foundational pillar for any credible transition toward post-quantum cryptography (PQC). Without a comprehensive, continuously maintained, and empirically validated understanding of where cryptographic mechanisms are embedded, organizations operate with limited visibility. They may not know which algorithms and protocols are operational. They may lack clarity on how keys and certificates are provisioned and managed. They may also be uncertain about what categories of assets are ultimately protected. Such opacity constrains their capacity to assess quantum-related exposure. It also weakens their ability to design coherent mitigation roadmaps. Crypto asset inventorying establishes the essential baseline. It does so by systematically

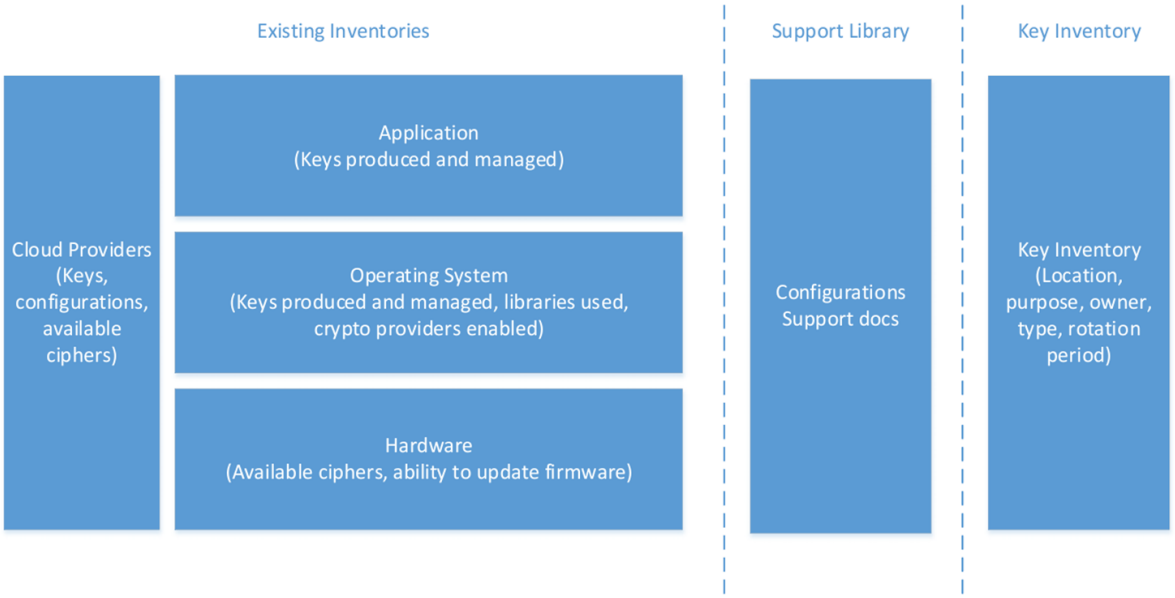

Figure 2.2: Ideal Crypto Asset Inventorying

identifying, classifying, and documenting cryptographic dependencies. These dependencies span applications, communication networks, operating systems, embedded platforms, and hardware security modules.

The imperative for structured inventorying is intensified by the architectural complexity of modern IT and OT environments. Contemporary organizations typically operate highly heterogeneous infrastructures. These environments integrate legacy systems with cloud-native services. They include distributed architectures, containerized workloads, and third-party service dependencies. Each layer may rely upon a diverse constellation of cryptographic standards, protocol implementations, and embedded libraries. Legacy systems are particularly challenging. They frequently depend on deprecated algorithms. They may rely on static configurations or hard-coded keys. In some cases, they use undocumented cryptographic modules. This makes them resistant to agile migration. The resulting structural diversity amplifies the operational intricacies of post-quantum transition efforts. It underscores the necessity of maintaining a dynamic, continuously updated, and centrally governed view of cryptographic utilization.

Regulatory and policy developments at the European level further reinforce the relevance of crypto asset inventorying. Policy instruments and guidance issued by the European Commission emphasize structured preparedness. The NIS2 Directive highlights the importance of risk management and resilience. Guidance from ENISA further stresses the need to evaluate and map existing cryptographic dependencies.

These frameworks collectively acknowledge that post-quantum migration is neither instantaneous nor purely technical. It is a long-term, risk-driven transformation process. It must be anchored in precise visibility. It must also rely on structured governance and foresight. The principal objective of crypto asset inventorying is to enable informed and defensible decision-making. It allows organizations to exhaustively identify cryptographic assets. These assets include algorithms, protocols, key pairs, certificates, trust anchors, cryptographic libraries, and hardware-based security components. Through this process, organizations attain granular visibility across every operational stratum of their infrastructure.

This enhanced visibility facilitates systematic vulnerability analysis. It is particularly relevant for algorithms demonstrably susceptible to quantum adversarial capabilities. It also enables quantitative and qualitative risk modelling. Cryptographic assets can be correlated with the sensitivity classification of the data they protect. The operational, financial, and reputational consequences of potential compromise can be more

accurately assessed. Beyond risk assessment, crypto asset inventorying plays a decisive role in structured migration planning. A rigorously maintained inventory permits organizations to prioritize remediation and transition efforts. Prioritization can be based on asset criticality, external exposure, system interdependencies, and practical feasibility of replacement or upgrade. Such prioritization is indispensable. Post-quantum cryptographic schemes exhibit significant variability in performance characteristics. They differ in computational overhead, bandwidth requirements, and technological maturity. Not all systems can transition concurrently. Nor can they uniformly adopt identical post-quantum mechanisms. Crypto asset inventorying therefore provides the analytical foundation for determining when, where, and which post-quantum cryptosystems should be deployed. It supports a calibrated balance between enhanced security assurances, operational continuity, and infrastructural resilience.

For inventorying processes to achieve substantive efficacy, they must extend beyond static documentation and theoretical policy declarations. Effective crypto asset inventorying requires empirical observation. It also requires automated discovery and continuous monitoring capabilities. These mechanisms must detect cryptographic elements embedded within network traffic, software dependencies, configuration files, authentication mechanisms, firmware, and application source code. This operational necessity has catalysed the development of specialized technological solutions. These solutions translate conceptual inventorying objectives into measurable and actionable capabilities. Tools such as Crypto Asset Inventory (CAI) demonstrate how organizations can achieve comprehensive, cross-layer cryptographic visibility. They help transform intent into tangible operational readiness in the face of the quantum computing paradigm shift.

2.2.2 Integration to high-level protocols

Crypto Asset Inventorying establishes the evidence base for migration by revealing where cryptography is used, which protocols and libraries are instantiated, and how trust is provisioned through keys and certificates across the infrastructure. However, visibility alone is insufficient to plan a defensible PQC transition. Migration feasibility is ultimately constrained by the maturity of high-level protocol integration, because modern service graph chains consume cryptography primarily through standardised security protocols and PKI-driven trust workflows rather than through standalone primitives. In operational terms, a PQC transition succeeds only if post-quantum (or hybrid) key establishment and authentication are supported end-to-end across protocol stacks, implementations, and relying parties. This system-level viewpoint is aligned with IETF guidance, which emphasises that PQC transition is an ecosystem migration problem: incremental deployment, deterministic negotiation, downgrade resistance, and backward compatibility dominate what is practically deployable in heterogeneous environments [43].

A consolidated view of the current maturity landscape is provided by the PQC adoption “heatmap” (Figure 2.3) summarised in the industry survey by Encryption Consulting and derived from the PQCC protocol adoption tracking. This representation compares major protocols (e.g., SSH, TLS 1.2/1.3, X.509/PKI, S/MIME, OpenPGP, IKE/IPsec, MLS, DNSSEC) across categories including pure PQ encryption, hybrid PQ encryption, pure PQ signatures, and hybrid PQ signatures, illustrating that maturity is uneven and frequently higher for hybrid key establishment than for pure PQ or for signature migration [124, 263]. For PiQASO, this unevenness is operationally significant: it indicates where near-term deployment can build on converging standards and where additional engineering effort is required to avoid fragmented adoption.

At the transport security layer, PQ integration is most advanced in TLS 1.3, where the IETF has defined a framework for hybrid key exchange [297]. This standardisation effort aligns with the current deployment pattern in which classical (EC)DHE is combined with a PQ KEM and the resulting secrets are combined to provide resilience during transition. In parallel, the IETF is standardising concrete TLS 1.3 integrations for ML-KEM-based key agreement and ML-DSA-based authentication, reflecting the practical convergence

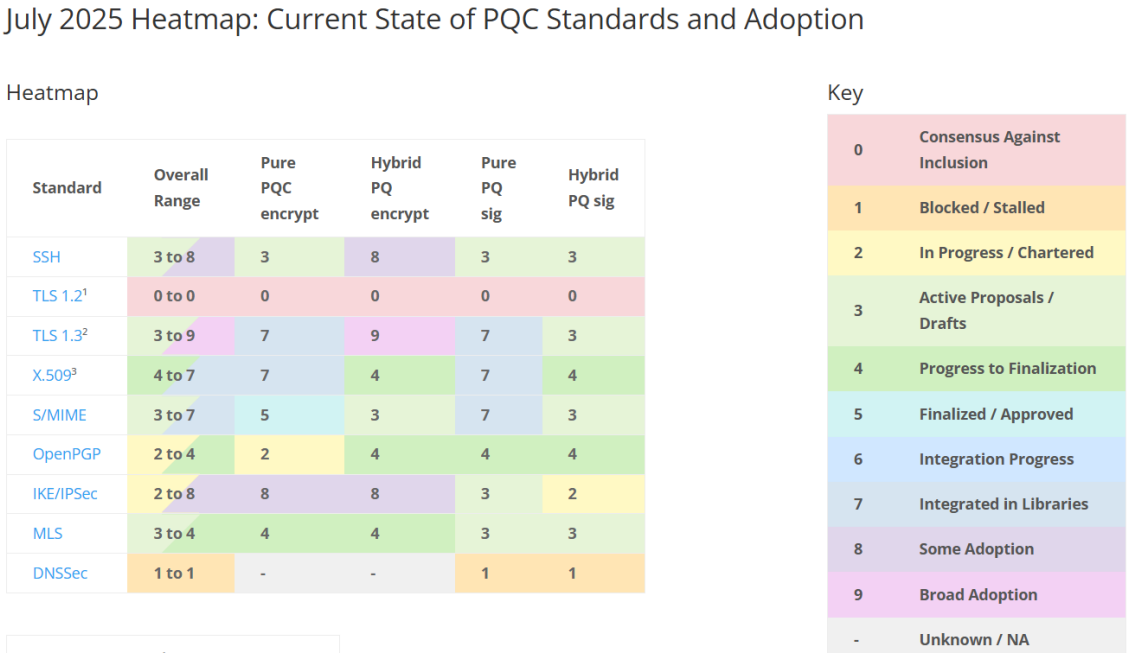

Figure 2.3: A recent and encouraging snapshot of the post-quantum PQC migration effort at high-end protocol level [124]

around NIST-standardised primitives for protocol deployment [90, 167]. This maturity is also reflected in large-scale, real-world experiments and rollouts: Cloudflare publicly reported broad deployment experiments for post-quantum/hybrid TLS key agreement and continues to document PQ support constraints (e.g., PQ key agreements supported in TLS 1.3-based stacks, with operational restrictions such as FIPS-mode considerations) [198, 86]. Chrome's platform-facing strategy similarly prioritises quantum-resistant key exchange and explicitly highlights the systemic challenge of PKI agility for large-scale transition, reinforcing that protocol adoption must be accompanied by trust ecosystem evolution [304].

At the network-layer security plane, the PQ transition of IKEv2/IPsec has proceeded through standards that explicitly address hybridisation and transitional operation. RFC 8784 specifies an IKEv2 extension that mixes preshared keys to increase resistance to quantum attacks, while RFC 9370 specifies a framework for performing multiple key exchanges in IKEv2 so that PQC can be integrated while maintaining backward compatibility [260, 314]. These documents provide a standards-based foundation for PQ-transitioned VPN and tunnel establishment, but also make explicit that protocol-level changes, negotiation semantics, and message-size constraints must be carefully engineered—an issue that is particularly relevant to deployments operating over constrained links and in environments where deterministic behaviour under partial compatibility is required.

Beyond "classical TLS-like" handshakes, the research community has also developed protocol variants that reduce the handshake overhead and shift authentication mechanisms to better accommodate PQ settings. KEMTLS proposes a TLS 1.3 handshake variant that uses KEMs instead of handshake signatures for server authentication, explicitly targeting efficiency and deployability constraints in post-quantum environments [286]. KEMTLS-PDK extends this line by leveraging pre-distributed public keys to further reduce bandwidth and computation overheads, at the expense of additional provisioning assumptions—an engineering trade-off that can be realistic in managed ecosystems with controlled key distribution channels [285].

At the implementation layer, protocol maturity is strongly influenced by the availability of deployable cryptographic stacks. The Open Quantum Safe ecosystem (notably the OpenSSL 3 provider approach) exposes post-quantum and hybrid capabilities for TLS 1.3 and associated X.509/CMS operations, enabling

experimentation and integration even when native vendor stacks lag [252, 253]. In parallel, the OpenSSL Foundation has publicly discussed post-quantum feature development paths (including hybrid ML-KEM support), reflecting an increasing convergence toward production-grade support for post-quantum primitives and hybrid configurations [257].

Overall, the protocol maturity landscape motivates a concrete conclusion for PiQASO: integration readiness is not uniform, and therefore protocol-aware profiling and prioritisation is a necessary enabling factor for any credible transition plan. Crypto Asset Inventorying provides the visibility needed to identify which protocols, cryptographic libraries, and trust dependencies are present across the infrastructure. The maturity assessment discussed here, then provides the basis to determine which dependencies can transition first, which are expected to rely on hybrid transitional modes, and which remain bottlenecks that must be mitigated through architectural controls and deployment planning. Importantly, protocol readiness alone does not guarantee deployability. Even when standards and implementations exist, operational deployment must account for hostile threat models, including physical access, side-channel exposure, and fault-injection capabilities against embedded endpoints and cryptographic modules. This creates the need to instantiate sensitive PQC and hybrid protocol functions within a protected execution context—i.e., as a trusted application anchored on quantum-safe cryptographic services and hardware-based key protection so that protocol-level security claims remain valid under realistic adversarial conditions.

2.2.3 Deployment and Instantiation as a Trusted Application

The maturity of PQC integration into high-level protocols is a necessary precondition for migration, but it is not sufficient for modern service graph chains. In contested and physically exposed environments, protocol-level security guarantees can be undermined if secret-dependent computations are executed in weakly protected software contexts, or if long-term private keys are exposed through physical attacks. This is particularly relevant for PQC implementations, which often operate on larger secret-dependent data structures and may involve longer computations, increasing the observable footprint for side-channel leakage and expanding the attack surface for fault injection.

PiQASO therefore treats deployability as a trusted instantiation problem: cryptographic functions that handle long-term secrets—key generation, private-key storage, key decapsulation, and signing—must be deployed as Trusted Applications within an isolated execution context and anchored to hardware-backed key protection. The objective is to ensure that the host operating environment, even if compromised, cannot directly access private keys or intermediate secret state, and that the interface between the rich execution environment and cryptographic services is minimised, strictly validated, and policy-governed.

At the architectural level, this instantiation model aligns with established secure-execution principles as formalised in Trusted Execution Environments (TEEs), where trusted applications run isolated from the rich OS and are accessed through a controlled client API and secure session model. Such isolation and trusted application lifecycle management is consistent with the GlobalPlatform TEE architecture and its trusted application model, which define security boundaries, secure storage, and the trusted application execution paradigm used across commodity devices. [143] TEEs serve as the enforcement point for cryptographic policy and for secure mediation of cryptographic operations, ensuring that sensitive code and data remain confined to the trusted world.

Beyond protocol maturity, trusted deployment is an architectural necessity driven by three imperatives. First, isolation of critical cryptographic kernels is required because not all components of a cryptographic workflow require the same protection level. A trusted-application deployment model draws explicit boundaries between trusted and untrusted execution contexts and confines the most sensitive operations—such as KEM decapsulation and digital signature generation—to an isolated environment. This prevents a compromised host operating system or application stack from directly accessing private keys, decapsulation

intermediates, or signing state, thereby preserving the confidentiality of long-term secrets even under host compromise.

Second, resilience against physical attacks must be assumed in realistic threat models. Software-only implementations—even when functionally correct—can be exposed to side-channel leakage (e.g., power and electromagnetic analysis) and fault injection, particularly on embedded platforms with extended field exposure. Trusted deployment therefore requires a hardware-backed secure element (or equivalent tamper-resistant module) that is explicitly engineered with countermeasures against leakage and fault adversaries. By offloading the most sensitive computations and key storage to this hardened module, the system reduces the attack surface available to physical adversaries and provides a practical basis for achieving measurable, certifiable resistance against side-channel and fault attacks. Foundational results on differential power analysis and practical fault attacks illustrate the feasibility of key recovery when implementations are not protected, motivating countermeasures such as masking, hiding, and redundancy/error-detection mechanisms as part of a defence-in-depth design [194, 58, 218]. In addition, recognised evaluation frameworks provide the baseline for structuring and validating assurance claims for cryptographic modules and their physical security posture [2, 233].

Third, trusted deployment provides the root capability for trust establishment and attestation in distributed security architectures. Beyond protecting execution, the trusted domain can act as the device's anchor for producing integrity evidence—attestation assertions over software configuration, cryptographic posture, and runtime integrity—that can be verified by remote relying parties. Such evidence is essential for establishing trust relationships in Zero-Trust settings, enabling policy enforcement that only verified nodes and verified cryptographic configurations may participate in protected communication and key management workflows.

Operationally, the Trusted Application boundary also enables a clean integration path with high-level protocols. Protocol stacks (e.g., secure channel establishment and certificate-driven authentication workflows) can invoke PQC services via the Trusted Application interface, while long-term keys and secret-dependent computations remain confined to the trusted domain. This design reduces the likelihood that protocol integration inadvertently reintroduces key exposure through debugging interfaces, memory disclosure vulnerabilities, or untrusted software dependencies. It also supports policy-governed hybridisation and algorithm agility at runtime: protocol-facing components can negotiate and select approved algorithm suites, while the actual private-key operations are executed under trusted enforcement, preserving assurance even in mixed and partially upgraded deployments.

The deployment and instantiation as a Trusted Application provide the missing link between "protocol support exists" and "protocol security holds under considered threat models". It anchors PQC operations in a protected execution and key-protection substrate that is compatible with recognised assurance frameworks, and it constrains the exposure of long-term secrets in the presence of realistic adversaries, including those with physical access and advanced measurement or fault capabilities. This trusted instantiation must also remain interoperable and profile-driven: interfaces and operational behaviours must align with emerging PQC protocol profiles and conformity expectations to avoid fragmentation across vendors and coalition deployments.

2.3 Alignment with European PQC Standardisation

The post-quantum transition is being shaped by a small set of highly influential standardisation and guidance ecosystems that collectively determine what is deployable, interoperable, and certifiable at scale. In practice, two primary reference points currently drive national and industrial transition programmes: NIST in the United States, which is producing formal algorithm standards and associated transition guidance, and ANSSI in Europe (France), which provides security-agency guidance on algorithm choices,

hybridisation strategy, and deployment constraints for high-assurance products. While these ecosystems are broadly aligned on the necessity of staged transition and hybridisation, they are not fully aligned today with respect to algorithmic preferences and profiling choices, and this divergence creates a practical convergence gap for service graph chains operating across transatlantic supply chains and coalition environments.

On the US side, NIST has already published its first post-quantum cryptography standards (e.g., ML-KEM, ML-DSA, SLH-DSA) and maintains an evolving programme for additional algorithms and transition guidance, positioning these standards as the baseline for widespread adoption across products and protocols [242, 239]. On the EU side, ANSSI's position papers emphasise early deployment through hybrid mechanisms and provide explicit agency guidance on algorithm selection principles, assurance evolution, and conservative options for high-security use cases [33, 34]. Importantly, ANSSI explicitly highlights that "choosing algorithms selected by NIST for standardization is not an absolute prerequisite" in early deployments, and it encourages conservative options where performance and bandwidth constraints permit [33, 34].

A concrete manifestation of the current profiling divergence is the treatment of FrodoKEM. In its follow-up guidance, ANSSI describes FrodoKEM as a more conservative alternative to structured lattice KEMs due to its reliance on plain LWE, while acknowledging that this conservatism incurs substantial performance and bandwidth penalties; ANSSI therefore encourages FrodoKEM as a valid option in high-security applications where this penalty is not prohibitive [34]. By contrast, NIST's first published standards designate ML-KEM as the primary standard for general key establishment, reflecting the stronger emphasis on deployability and efficiency for broad ecosystems and protocols [242, 239]. This example illustrates a wider issue: even where long-term convergence is expected, near-term deployment profiles may differ (e.g., "conservative but heavy" versus "efficient and standardised"), and guidance is needed to avoid fragmentation of interoperability profiles across service graph chains, vendors, and certification pipelines.

Beyond algorithm selection, convergence also depends on protocol and platform standards bodies that operationalise PQC into real systems. The IETF is central to this layer, because it defines how PQC is integrated into high-end security protocols and PKI artefacts that underpin authenticated key exchange and trust establishment across networks. IETF engineering guidance emphasises that PQC transition is an ecosystem migration problem in which incremental deployment, negotiation behaviour, downgrade resistance, and backward compatibility dictate real-world feasibility [43]. Recent IETF work items further extend this transition into application and PKI layers, including recommendations for PQC usage profiles in TLS-based applications and explicit treatment of pure PQ versus hybrid (composite) X.509 certificate strategies [271]. This IETF layer is therefore a key determinant of what constitutes "compatible" post-quantum operation in practice.

At the hardware root-of-trust layer, the Trusted Computing Group (TCG) provides the dominant industry specifications for trusted platform modules and related trust anchors. TCG has explicitly stated that it is updating specifications to prepare for the post-quantum era and that these updates depend on the algorithms and parameter sets published by agencies such as NIST, highlighting that platform trust standards evolve under external cryptographic dependencies [305]. This adds a further dimension to convergence: even if NIST and ANSSI guidance converges at the algorithm level, practical alignment still requires consistent profiles that can be embedded into platform trust anchors and consumed uniformly by protocol stacks, credential ecosystems, and relying parties.

The current landscape indicates an alignment gap across (i) algorithmic profiling preferences (e.g., conservative options such as FrodoKEM versus efficiency-driven baselines), (ii) protocol-level integration profiles governed by IETF processes, and (iii) platform trust anchor evolution governed by TCG specifications. For service graph chains where interoperability, certification, and long lifecycle constraints are binding, this gap cannot be addressed by algorithm selection alone. What is required is structured guidance that (a) defines interoperable PQC profiles across agencies, (b) aligns protocol and certificate

migration patterns with those profiles, and (c) ensures that platform trust anchors can support these profiles consistently. This is precisely why PiQASO treats profiling and metrics construction as a necessary step towards harmonised, deployable, and assurance-aligned migration.

2.3.1 Gaps in PQC Standardisation Profiling and Security Evaluation Metrics

A second—and currently more severe—convergence gap concerns how PQC implementations will be evaluated and certified under realistic attacker models. While algorithm standards are now being published and protocol integrations are progressing, harmonised certification methodologies for PQC modules (especially under physical attacks) are not yet mature or consistently codified across schemes. This matters directly for practical deployments, as PQC will often be instantiated in security-critical, physically exposed platforms (secure elements, TPM-class roots of trust, embedded endpoints), where certification is a procurement and assurance prerequisite rather than an optional add-on.

In Europe, the newly adopted EUCC scheme is explicitly grounded in Common Criteria evaluation principles and associated methodologies [126, 88, 1]. EUCC inherits the long-standing CC approach to vulnerability analysis (AVA-VAN) and attacker capability modelling (including attack potential scoring), which has been refined substantially in the smartcard/secure-element community [125]. In the United States, comparable assurance is typically achieved through FIPS 140-3 / CMVP, where algorithm validation (CAVP) is a prerequisite for module validation [233, 232, 241]. Importantly, these frameworks are mature as certification structures, but they were developed primarily around classical cryptographic modules and established evaluation practice for conventional primitives.

For PQC, the gap is not the absence of any evaluation method, but the lack of PQC-specific, standardised guidance that reliably maps (i) which physical attack classes are critical per PQC primitive and implementation style, (ii) which countermeasures must be present and how their effectiveness should be tested, and (iii) how to express these expectations as interoperable protection profiles / evaluation vectors. Existing standards for non-invasive attack testing—such as ISO/IEC 17825—provide structured test metrics for assessing mitigation against non-invasive attacks at the module boundary, but they do not, by themselves, deliver PQC-family-specific evaluation profiles or resolve the composability and coverage questions that arise for diverse PQC kernels and parameter sets [176, 177, 311]. Similarly, CC evaluation methodology (CEM/ISO 18045) specifies what evaluators must do at a process level, but PQC-specific test artefacts (e.g., reproducible attack traces, fault models, leakage models, and pass/fail criteria for concrete PQC kernels) are not yet harmonised in the way that classical smartcard evaluation practice has achieved for certain established primitives [88, 1, 125]. The result is a practical uncertainty: organisations may be able to integrate PQC into products and even validate algorithms under CAVP/CMVP workflows, yet still face ambiguity over which physical attacks must be demonstrated, which countermeasures are “sufficient”, and how evaluation effort and cost should scale with attacker capability in a consistent, repeatable way [241, 232].

PiQASO’s activities on physical-attack demonstration and protection for PQC implementations can directly contribute to closing this gap by providing the kind of evidence and technical artefacts that certification schemes require to converge on actionable profiles. In particular, PiQASO’s experimentally grounded results can be leveraged by the wider PQC community to: (i) define attack-path profiles for PQC kernels (e.g., decapsulation/signing) that align with attacker-capability concepts used in EUCC/CC and the smartcard evaluation ecosystem [125, 126]; (ii) produce repeatable evaluation vectors (measurement setups, leakage/fault models, and pass/fail criteria) that complement existing non-invasive testing standards and make them more directly applicable to PQC modules [177, 176]; and (iii) support the construction of PQC protection profiles / assurance templates that are interoperable across vendors and evaluators, reducing fragmentation between “algorithm standardisation” and “certifiable deployment.” This is a necessary precondition for practical adoption, because it enables procurement-aligned assurance claims and reduces

the risk that PQC migration stalls at the point where products must demonstrate measurable resilience against physical adversaries.

Chapter 3

PiQASO Framework Main Building Blocks

In order to fulfil the *core motivation of PiQASO for providing quantum-resistant security capabilities to service graph chains belonging to a wide variety of application domains*, the approach followed by PiQASO entails the creation of a **testbed that provides the capability for benchmarking the PiQASO Post-Quantum security protocols in the context of the use cases**, in order to measure their impact on the operations conducted as part of their normal operation. Throughout this Chapter, we will provide a high-level overview of the building blocks comprising the PiQASO framework, which will then constitute the basis for their development throughout the lifecycle of the project.

Specifically, PiQASO adopts a **3-layered** approach for building a holistic quantum resilient solution towards enabling an operationally realistic transition to PQC across heterogeneous infrastructures. The approach is designed to (i) establish visibility and evidential grounding on where and how cryptography is used, (ii) enable controlled integration and deployment of PQC and hybrid transitional configurations with minimal disruption to operational services, and (iii) support continuous verification and assurance under realistic adversarial models, including physically capable attackers. In contrast to purely algorithm-centric viewpoints, PiQASO treats PQ transition as a system-level solution that couples cryptographic primitives, protocol stacks, and trusted execution into a coherent framework suitable for long-lived and mixed-criticality environments. Crucially, PiQASO is engineered to safeguard the **entire computing continuum**, spanning **edge nodes** to **back-end/cloud infrastructures**. This continuum-wide scope is essential in modern service graph chains, where services and trust relationships are inherently distributed and where adversaries may target the weakest point of the chain—often a physically exposed or resource-constrained endpoint—to undermine end-to-end confidentiality, integrity, availability, and operational assurance.

Figure 3.1 provides a consolidated view of PiQASO's end-to-end transition logic across the computing continuum. It highlights the closed-loop nature of the framework: starting from crypto-asset inventorying and posture assessment, progressing through PQC selection and orchestration (including hybrid transitional configurations where required), and culminating in verification activities such as implementation rating, continuous monitoring, and recommendations. The figure also reflects PiQASO's design objective of preserving deployability across both low-end embedded platforms and high-end infrastructures (comprising the entire computing continuum), ensuring interoperability with legacy infrastructures. The latter is one of the main bottlenecks identified in EU's roadmap for a concrete transition strategy: *Miniaturization of PQC implementation footprint with full hardware re-use so as to unlock their integration in diverse applications across different services*. This is one of the main endmost goals of PiQASO to be achieved through not only a detailed profiling of each PQC primitive but also the provision of different modalities capturing **industry's favour towards hybrid- and agile-by-design PQC settings.**

Assessment and Characterisation of Cryptographic Assets and Posture. PiQASO establishes the starting point for transition by characterising the cryptographic posture of an infrastructure and producing

Co-funded by the European Union ECCC

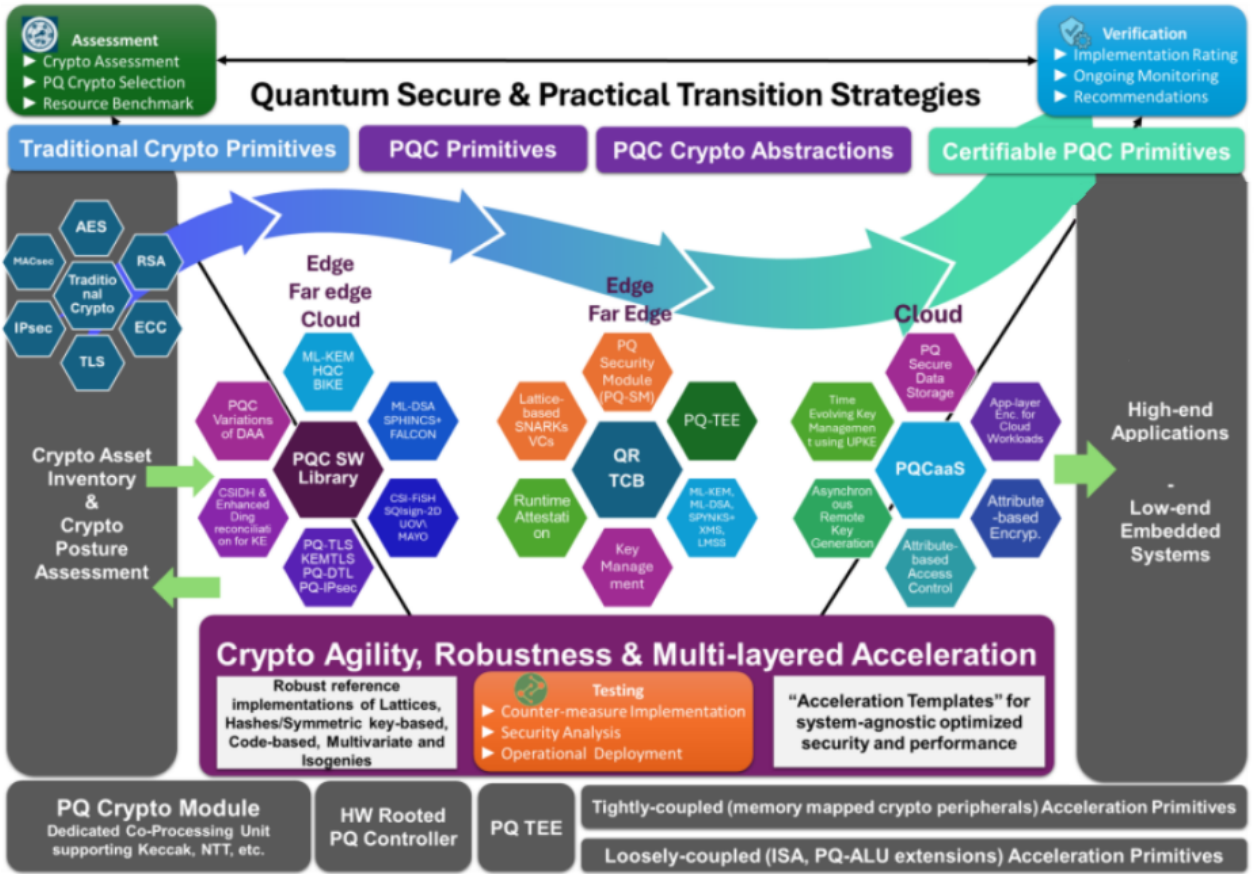

Figure 3.1: PiQASO PQC framework safeguarding the computing continuum (edge to cloud)

structured inventories of cryptographic assets (Section 3.5). This includes the identification of cryptographic dependencies at multiple layers—protocol instantiations, libraries, certificate chains, key sizes and lifetimes, and key management interfaces—so that quantum-vulnerable elements can be prioritised for remediation. The assessment phase is not limited to passive listing; **it is designed to support continuous observability and traceability of cryptographic usage across operational services, enabling the construction of crypto inventories that are actionable for migration planning across the edge-to-cloud chain.** The outcome of this process is a structured representation of cryptographic dependencies (e.g., a cryptography bill of materials–style view) that feeds risk and conformity reasoning, supporting the definition of short-term (hybrid) and long-term (pure PQ) transition strategies consistent with operational constraints.

PQC Integration and Orchestration. Based on the characterised posture, PiQASO supports the orchestrated deployment of PQC capabilities in a way that preserves interoperability with legacy systems and reduces operational disruption. The orchestration & operational deployment of post-quantum cryptosystems leverages PiQASO PQC Ensemble secure implementations (Section 3.1) featuring a fully-fledged (SW) optimization and (HW) acceleration design space while embodying advanced counter-measures (CMs) against the most prominent types of physical attacks (see Section 4.7 for the threat model to be considered as a basis demonstrating PiQASO's enhanced resilience). To better capture the current needs of both high-end (safety-critical) applications but also low-end embedded devices, PiQASO will provide a 3-tier service modality: (a) a HW-augmented and accelerated implementation providing a full instantiation of a QR Trusted Computing Base (QR-TCB as described in Section 3.2) enabling cryptographic functions including secure authentication, KEM, encryption and signing functions in a tamper-resistant manner, turning the host device into a "hardened" security token with sustainable security, featuring protection techniques against both side-channel and fault injection attacks with the specific target to be the first PQ-ready Secure Element to achieve certifiable physical security; (b) a cryptographic SDK (PQC SDK) that

can be integrated into any commodity system for secure data processing and crypto operations; and (c) an "as-a-service" model that provides the first instance of a cloud self-service model (PQC "as-a-Service" (PQC AAS)) (Section 3.3) capable of delivering the trustless and computational secure benefits of PQC to all and any connected devices that want long-term secure-by-design data storage (PQC anywhere and anytime service model). All these functional components will be commonly designed by adopting a flexible SW/HW co-design approach enabling full programmability, as it pertains to optimization and acceleration, resulting into robust (universal) accelerators that can support the optimal (re-)parameterization of different families of crypto algorithms (i.e., lattice-, hash-based) which in turn guarantees high security with high performance. PiQASO PQC Ensemble SW/HW co-design will leverage two types of acceleration primitives: (i) HW acceleration of those crypto workloads/operations leveraging the underlying (soft-core) general-purpose processing unit, and (ii) HW acceleration of those resource-intensive computations supported through dedicated co-processing units. Combining the flexibility of the soft-core and the performance of dedicated HW accelerators, makes PiQASO PQC co-design an ideal design approach for resource-constrained commodity devices. Our implementations will be tested on an FPGA platform representing a commodity platform rather common in the majority of defence applications.

Testing, Validation, and Verification for Conformity and Assurance. PiQASO closes the loop between deployment and assurance by combining continuous monitoring of cryptographic posture with validation and verification activities that quantify both performance and resilience. This includes (i) implementation-level benchmarking and profiling to evaluate operational overheads induced by PQ and hybrid configurations, and (ii) security validation against realistic adversarial models, with particular emphasis on physical and implementation-level attacks (e.g., side-channel and fault attacks against embedded endpoints and cryptographic modules). The objective is not merely to "test implementations," but to generate structured evidence that supports repeatability, comparability, and eventual convergence toward evaluation profiles and conformity expectations for PQC modules and PQ-enabled protocol deployments.

PiQASO's layered approach enables a disciplined transition programme: it begins with evidence-driven characterisation of cryptographic dependencies, proceeds to orchestrated PQC integration, anchored in trusted deployment patterns across the computing continuum, and is continuously supported by validation and verification activities that produce actionable assurance evidence. In doing so, PiQASO aims to reduce fragmentation between "standardised algorithms" and "deployable, certifiable systems," providing a coherent pathway toward quantum-resilient networks that preserve operational continuity under realistic threat conditions.

3.1 PiQASO Software Library

One of the main objectives of PiQASO is to **develop robust and highly-optimized implementations of the NIST-standardized PQC primitives ML-KEM, ML-DSA and SLH-DSA**. Robustness of the implementations is ensured by the multi-layered threat model, described in Section 4.7, which will enable a series of countermeasures at implementation-level for mitigating risks and vulnerabilities introduced by SCAs, SW-based FIAs and inconsistencies in SW implementations (SW-bugs). For the optimization of PQC algorithms, **PiQASO aims at conducting an extensive study on investigating algorithmic improvements in order to overcome performance bottleneck**s. More precisely, this rigorous study will focus on optimizing performance-intensive subroutines in PQC primitives, particularly the NTT-based polynomial arithmetic operations, which dominate the execution time of lattice-based algorithms. Since NTT is the main computational building block of lattice constructions, accelerating NTT will yield performance gains also in the more advanced lattice-based schemes which are part of the PiQASO initiative (ABE and UPKE).

The **PiQASO PQC Software Library** is dedicated to high-quality, embedded-oriented implementations of

post-quantum cryptographic (PQC) algorithms. Its objective is to collect, consolidate, and maintain optimized PQC implementations developed within the project, ensuring a high level of assurance, systematic testing, and integrated countermeasures against side-channel attacks.

The PiQASO PQC Software Library focuses on providing lightweight asymmetric cryptographic primitives suitable for constrained platforms, combining efficiency, robustness, and side-channel resistance. In particular, the library integrates masking and shuffling techniques as built-in countermeasures against side-channel leakages. To the best of our knowledge, no existing state-of-the-art library currently offers embedded-oriented PQC implementations that simultaneously combine high code quality, fine-grained function-level testing, and integrated first- and second-order masking protections. The library includes the following main components:

- **ML-KEM implementation compliant with FIPS 203**: An embedded-oriented implementation of ML-KEM fully compliant with the FIPS 203 specification is provided. The design targets constrained platforms and relies on static memory management to avoid dynamic allocation. All internal functions are individually tested to ensure correctness and robustness. The implementation supports hardware accelerators developed by CEA. Dedicated validation functions are included to verify correctness against official NIST test vectors and reference implementations such as mlkem-native[1].
- **ML-DSA implementation compliant with FIPS 204**: An embedded-oriented implementation of ML-DSA compliant with the FIPS 204 specification is also included. As for ML-KEM, the implementation uses static memory management and ensures that all internal functions are thoroughly tested. Support for CEA-developed hardware accelerators is provided where applicable. Validation functions ensure conformance through official NIST test vectors and available reference implementations such as mldsa-native[2].
- **First- and second-order masking gadgets for ML-KEM**: The library provides all necessary cryptographic gadgets to enable first- and second-order masking of ML-KEM. These components are specifically designed for embedded environments and are systematically tested and characterized to ensure the absence of exploitable side-channel leakages. Particular attention is devoted to resistance against power and electromagnetic analysis attacks, including higher-order attacks.
- **First- and second-order masking gadgets for ML-DSA**: Similarly, the library includes the required masking gadgets to support first- and second-order protected implementations of ML-DSA. These components are thoroughly tested and characterized to verify their effectiveness in preventing information leakage through side-channel attacks, ensuring robust protection in embedded deployment scenarios.
- **First- and second-order masked ML-KEM implementation**: A fully protected version of ML-KEM is included, leveraging the independently developed first- and second-order masking gadgets. This implementation integrates the masking components into the complete algorithm flow and may incorporate additional countermeasures, such as shuffling at carefully identified sensitive points of the algorithm to further reduce side-channel leakage. The masked implementation is thoroughly characterized through dedicated leakage assessment campaigns and includes support for hardware acceleration, ensuring compatibility with CEA-developed cryptographic accelerators while maintaining side-channel resistance.
- **First- and second-order masked ML-DSA implementation**: A fully protected ML-DSA implementation is also provided, integrating the dedicated first- and second-order masking gadgets within the full signature scheme. As with ML-KEM, additional countermeasures such as selective shuffling

[1] https://github.com/pq-code-package/mlkem-native
[2] https://github.com/pq-code-package/mldsa-native

may be applied at critical stages of the algorithm. The implementation is characterized through systematic side-channel evaluation to assess its resistance against power and electromagnetic attacks. Support for hardware acceleration is included, enabling integration with CEA-developed accelerators while preserving the intended security guarantees.

3.1.1 Algorithmic Optimizations of Lattice-Based Algorithms

The practical deployment of lattice-based Post-Quantum Cryptography (PQC) requires not only theoretical security guarantees but also highly optimized implementations capable of meeting stringent latency and resource constraints. As these primitives rely heavily on polynomial arithmetic over finite fields, the efficiency of the underlying multiplication algorithms is the primary driver of overall system performance. While standard techniques such as the Number Theoretic Transform (NTT) provide asymptotic efficiency, they often incur significant concrete overhead due to the dense frequency of modular reduction operations required in software. In this section, we propose an algorithmic optimization that mitigates this bottleneck by transposing the arithmetic burden from the modular domain to the complex domain. By leveraging fixed-point arithmetic, we achieve better performance while preserving the constant-time execution, which is necessary for security against timing side-channel attacks.

Context and Mathematical Foundations Lattice-based PQC primitives rely on the hardness of problems over structured lattices to achieve an optimal balance between security, computational performance, and transmission bandwidth (ciphertext and key sizes). These primitives typically operate over polynomial rings, where elements are represented as polynomials—or matrices of polynomials—residing in quotient rings of the form:

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1), \tag{3.1}$$

where N is a power of two (typically 256 or 512) defining the lattice dimension, and q is a prime modulus. The efficiency of polynomial operations within this ring structure is a critical determinant of the primitive's overall performance. The most computationally intensive operation is polynomial multiplication. Given two polynomials $a(x) = \sum_{i=0}^{N-1} a_i x^i$ and $b(x) = \sum_{i=0}^{N-1} b_i x^i$, their product $c(x) = a(x) \cdot b(x) \mod (X^N + 1)$ is defined as a negacyclic convolution. The coefficients c_k of the resulting polynomial are given by:

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} - \sum_{i=k+1}^{N-1} a_i b_{N+k-i} \quad \text{for } k = 0, \ldots, N-1. \tag{3.2}$$

A naive implementation of this convolution requires $O(N^2)$ scalar operations, which is prohibitively slow for the large values of N required for high-security applications.

Standard Approach vs. Computational Bottlenecks To mitigate the $O(N^2)$ complexity, parameters N and q are typically selected to support the NTT [284]. The NTT is an analogue of the Fast Fourier Transform (FFT) over finite fields, allowing multiplication to be performed in the frequency domain. By transforming the polynomials into their NTT representations, point-wise multiplication can be applied, reducing the asymptotic complexity to $O(N \log_2 N)$. However, while the NTT reduces the operation count, the standard implementation introduces significant overhead due to the nature of modular arithmetic. Modular reductions are required after every addition and multiplication to keep coefficients within the field $\mathbb{Z}_q$. These modular reduction operations hinder the performance.

Proposed Optimization: Complex Domain Mapping To address these bottlenecks, we propose an algorithmic optimization that shifts the arithmetic domain. Instead of performing operations directly in $\mathbb{Z}_q$, we map the original polynomials to appropriate polynomials with complex coefficients. This allows us to leverage standard FFT-based convolution techniques rather than the modular NTT. The core of this strategy involves implementing fixed-point arithmetic to retain constant-time execution—a mandatory requirement for preventing timing side-channel attacks. In this optimized flow, all intermediate operations (FFT, point-wise multiplication, inverse FFT) take place using standard integer instructions without intermediate modular reductions. The costly modular reduction modulo q is deferred until the very end of the polynomial multiplication process.

Precision and Correctness Strategy The viability of this approach hinges on managing numerical precision to prevent data loss. Since fixed-point arithmetic is an approximation of the continuous complex domain, strict error analysis is required. We ensure correctness through a dual-selection strategy:

1. Integer Width Selection: We carefully select the appropriate container size (e.g., int32 vs int64) to accommodate the dynamic range of the values during the accumulation steps of the convolution, ensuring that the maximum possible value does not exceed the container's capacity (overflow protection).
2. Fractional Bit Allocation: We determine the optimal number of bits dedicated to the fractional part of the fixed-point representation. This minimizes the quantization noise introduced during the FFT/IFFT stages.

By rigorously configuring these parameters, we maintain sufficient precision to ensure that the accumulated error remains strictly below the rounding threshold. Consequently, the final fixed-point result rounds correctly to the exact integer expected from the theoretical convolution, guaranteeing mathematical equivalence to the standard NTT approach while reducing instruction overhead.

3.2 Quantum-Resistant Trusted Computing Base (QR-TCB)

The Quantum-Resistant Trusted Computing Base (QR-TCB) considered in PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). Long-term private keys are generated, stored, and used exclusively inside a Trusted Execution Environment (TEE), ensuring isolation from the Rich Execution Environment and untrusted software. These keys serve as the root for deriving session keys, enabling secure communication, and providing authenticity through PQC signatures. The blueprint emphasizes minimizing the trusted surface while maintaining sufficient functionality to preserve security guarantees and avoid excessive secure/non-secure transitions that would increase latency and observable leakage. To accommodate heterogeneous device capabilities and deployment requirements, the QR-TCB follows a three-tier architectural model that balances security and efficiency. The highest-assurance tier executes full cryptographic primitives within the TCB, providing maximal isolation and reduced exposure of intermediate secrets.

A second tier confines only secret-dependent operations, such as key generation, secure key storage, signing, and decapsulation, to the TCB, while non-sensitive computations are delegated outside to reduce trusted code size and improve performance. The third tier introduces fine-grained partitioning of cryptographic primitives, selectively executing only the most sensitive algorithmic components inside the TCB, guided by leakage and performance analysis. This tiered approach enables adaptable deployment

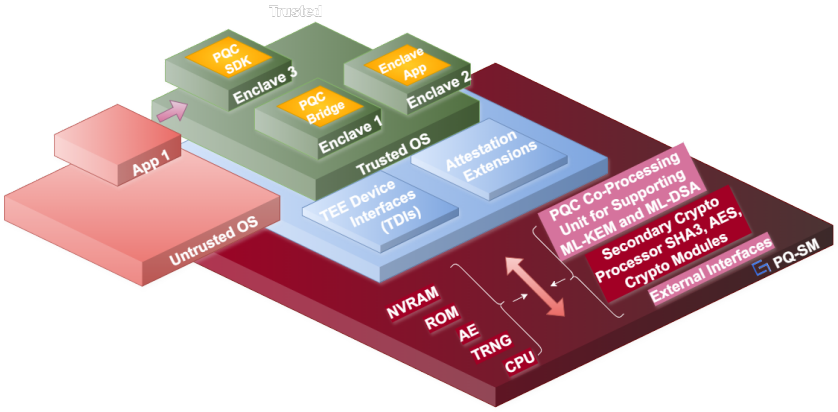

Figure 3.2: QR-TCB blueprint in the context of PiQASO

across platforms with varying performance, energy, and threat constraints. The QR-TCB will hold long-term keys that will be isolated from normal execution via a TEE. These keys will, in turn, be used to derive new session keys for communication, or provide authenticity via signatures. In short, the three levels that will be considered are

1. Full execution of the cryptographic primitives inside the TCB.
2. Delegation of only secret-key dependent high-level functions inside the TCB (key generation and key storage, and signing/decapsulation).
3. Low-level segmentation of the primitives that need to be run in the TCB or outside.

Beyond functional partitioning, the blueprint incorporates configurable hardware and execution options. Hardware accelerators for PQC operations may be bound to the TCB to provide constant-time, leakage-resilient execution, or alternatively shared with the non-secure domain where performance constraints dominate. In some deployments, dual accelerators may be employed to maintain strict isolation between trusted and non-trusted computation. Similarly, ephemeral/session key processing, used to ensure forward secrecy, may be executed either inside the TCB for maximal protection or outside when performance or resource constraints require it, provided that long-term secrets remain fully protected within the trusted domain. A central design objective of the QR-TCB is resistance to side-channel leakage, with masking serving as the primary protection mechanism. The blueprint considers the possible integration of masked implementations with constant-time execution, controlled memory access patterns, and careful coordination with the TEE to limit observable behavior across domain boundaries.

For signing the attestation evidence used in the context of the attestation enablers exposed by the PiQASO QR-TCB, we will investigate (hash-based) group signatures from symmetric primitives. To construct a group signature scheme, the first design choice is selecting group membership credentials. A credential is essentially a signature on a user's key generated by the group issuer, while a group signature serves as a non-interactive zero-knowledge proof of the credential. Since we utilise only symmetric primitives, the credential can either be a hash-based signature or a signature from another symmetric primitive. Many researchers have employed a single Merkle tree signature as a group membership credential. Some examples include group signatures [70, 120, 191, 290, 317, 318] and enhanced privacy ID (EPID) [59]. However, these schemes are limited to handling small group sizes. Recently, Chen et al. introduced a new group membership credential, which is a modified SPHINCS+ signature and can support a large group size of up to 260. They proposed to use this type of credentials to generate group signatures [79], Direct Anonymous Attestation (DAA) [78] and Enhanced Privacy ID (EPID). Unfortunately, the performance of this type of group membership proof is not optimal; with a group size of 260 and a security

level of 128 bits, it takes 30 seconds to prove a group membership and 14 seconds to verify, resulting in a signature size of approximately 2 MB. In PiQASO, we will investigate the development of a more efficient group signature scheme from symmetric primitives. We will explore suitable group membership credentials and their corresponding zero-knowledge proof. For example, we will investigate the possibility of employing a symmetric-setting signature, FAEST, which is a candidate in the second round of NIST's PQC for additional digital signature schemes. It is based on the VOLE-in-the-Head technique, which is similar to MPCitH.

3.3 PQC-as-a-Service (PQaaS)

PQC-as-a-Service (PQCaaS) is envisioned by PiQASO as a means to *provide devices that do not possess the computational resources to execute complex cryptographic operations with the capability to run the PQC schemes of PiQASO*. Specifically, one modality offered by the PQCaaS aims to enable devices to perform secure storage of their own data, i.e., perform encryption at rest. Devices with low computational power typically do not possess the capability to create their own keys, thus PiQASO offers a **PQ Data Encryption Server** responsible for performing the encryption of the data, so that devices do not need to perform the encryption themselves. This enables the implementation of lattice-based **Attribute-Based Encryption (ABE)**, leveraging PQ crypto primitives, so that the encryption server can encrypt a set of data at rest under a specific set of attributes. However, this assumes that the encryption server can be trusted to manage the attribute keys, which may not always be a realistic assumption.

It follows that, in case the encryption server cannot be assumed trusted, this modality may be insufficient, and in cases where security, privacy, and unlinkability are needed, the trust assumption of the encryption server may not be applicable. Thus, the PiQASO PQCaaS is envisioned to offer the additional modality of **Updatable Public Key Encryption (UPKE)** in order to mediate the secure and privacy-preserving sharing of data. Thus, if a data provider wishes to associate their data, either with specific attributes or encrypted under specific attribute policies, it is possible for only decryptors that possess a specific set of attributes to decrypt the data. Through the novel UPKE scheme of PiQASO, a Private Key can be associated with multiple Public Keys, which are unlinkable between them. *Thus, considering that a data encryptor is not trusted by default, the UPKE scheme enables the provision of such a Public Key for the encryption of the data, in a manner that does not reveal any information about other encryptors using different Public Keys to encrypt their data*. Then, in case an entity queries for a set of encrypted data, the PQCaaS acts as a mediator between the requester and the data source in order to determine if they are permitted to access and decrypt the data. Further information on these two modalities of the PiQASO PQaaS component will be provided in Section 3.3.2.

3.3.1 Towards Sustainable Security and Forward Secrecy

In order to achieve sustainable security, PiQASO aims to establish a PQC ensemble that is able to fulfil all overarching security requirements of the target infrastructure, while providing protection against the post-quantum threat landscape. In this regard, as aforementioned, a core consideration of the PiQASO PQCaaS component is the **secure management and protection of stored cryptographic keys**, which is a highly important and challenging issue in practical applications, as the strength of the cryptographic keys is of paramount importance in the overall security of any cryptographic scheme. For instance, if a secret key is revealed, all encrypted data associated with it is compromised and is henceforth considered public. In scenarios where **fine-grained access control** is required, the issue of key management becomes even more complex, as there is typically a key hierarchy that governs the access control process, thus highlighting the importance of secure key management.

In order to address these issues and mitigate the impact of key leaks, cryptographic properties such as forward secrecy and post-compromise security are typically considered. In the context of PiQASO, a powerful concept that can enable the realization of fine-grained key management is **puncturable encryption**, which is able to provide strong security to encrypted data sharing applications, and can be elevated to quantum-safe assumptions. Especially in the case of key management, it is useful to consider the concept of **puncturable key wrapping**, which specifically targets encrypted data at rest. This is realized through the PiQASO **Updatable Public Key Encryption (UPKE)** scheme, which is a notion of asymmetric encryption that has been demonstrated to be an important tool for key rotation in outsourced storage, and can also be constructed from quantum-safe assumptions. This can be likened to **key rotation** techniques, based on which one attribute-based key is associated with each data workload.

Updatable Public Key Encryption (UPKE) [182, 26] is an innovative PKE approach offering forward security through regular updates of public/private key pairs such that potential compromise of future private keys does not break confidentiality of ciphertexts encrypted under earlier public keys. In UPKE, senders can generate update tokens which are then asynchronously used by the recipient to update their own public key. To ensure that at any time only one version of the recipient's public key is used, UPKE requires interaction and coordination regarding the order in which the recipient processed the update tokens. The UPKE concept suits particularly well for achieving end-to-end forward security in asynchronous messaging, such as in the recent MLS standard [47], and offers tremendous potential in encrypted data sharing applications.

Earlier UPKE constructions were obtained using complicated and inefficient building blocks such hierarchical IBE. More recent UPKE schemes offer better efficiency and enjoy generic constructions, e.g. [108, 6], which can be instantiated using post-quantum secure techniques from lattice-based cryptography. They also offer so-called "insider security" capable of resisting chosen-ciphertext attacks. For example, [108] offers a UPKE instance from an LWE-based PKE scheme with the properties of circular security, leakage resilience, and homomorphism for key / message spaces.

The core innovation of our approach are the new post-quantum secure flavours of UPKE tailored for use in the cloud-based PQaaS architecture. By using UPKE within PQaaS we aim to enable two different innovative post-quantum secure encryption services. Our first PQaaS service will provide forward secure end-to-end encrypted data sharing amongst multiple users. A trusted PQaaS provider will be responsible for the evolution of time and management of public key updates for all users over the evolving time epochs. Our new UPKE flavours will support co-existence of multiple unlinkable public keys per user and epoch and the PQaaS provider will equip each sender with a different (unlinkable) public key per each time epoch when encrypting data for the same receiver. Our UPKE flavours will be constructed using Asynchronous Key Remote Generation (AKRG), a recent technique which allows to asynchronously derive multiple unlinkable public keys from some given public key. The use of post-quantum secure AKRG in the context of UPKE design is an independent novelty. Our second PQaaS service will adopt UPKE to provide forward secure attribute-based encryption (ABE) by initialising and managing an alternative chain of evolving public keys that will be associated with attributes/policies. Any sender wishing to share data encrypted under some policy/attributes will obtain the currently valid corresponding public key(s) from the service. Any eligible receiver will be able to obtain appropriate decryption keys from the service following verification of their eligibility. The corresponding ABE public keys will evolve over time using UPKE techniques to ensure forward security of the underlying encrypted data.

3.3.2 Description of the PiQASO UPKE Approach

From all the above, it follows that the goal of the PiQASO PQCaaS is twofold: (i) provision of all the high-assurance and accelerated PQC operations through a cloud-based service so as to ensure inter-

operability and avoid fragmented solutions with barriers due to discrete HW architectures and secure elements, and (ii) secure data compute and processing capabilities, thus, enabling the execution of complex data sharing agreements with strict security and privacy controls. The PiQASO PQCaaS will expose PQC crypto operations as-a-service, utilising at its core the QR-TCB (Section 3.2) and the PQ SW library (Section 3.1) modalities. As shown in Figure 3.3, PQCaaS comprises various components, summarized as follows:

- The **PQ Data Encryption Server (PQ-DS)** supports encryption and signing operations, by interfacing with the QR-TCB and instrumenting the advanced Key Management System offered by the Key Provider component. The PQ-DS is dependent on three core service components, namely **Key Provider**, **IO Provider** and **Identity Provider**.
- The **Key Provider** exposes new key management and key provisioning functionalities for allowing the quantum encryption/decryption of received Data Payloads. It is responsible for managing the lifecycle of crypto keys generated by the PQCaaS, allowing client-based encryption/decryption of the Data Payloads, while it exposes the core innovation of time-evolving key management as part of UPKE/ABE functionalities, maintaining key chains over the time epochs (incl. generation of key update packages) per each user/client and attribute/policy.
- The **IO Provider** and **Identity Provider** expose the traditional interfaces for allowing the (sharing) transactions with the stored data payloads, by data requestors, after having authenticated and verified the identity of the end-user (for either pushing data for quantum-secure storage or requesting access to already stored data). The Identity Manager component undertakes the user/device onboarding authentication and attribute-based authorisation actions. In PiQASO, all entities will be equipped with a PQ digital certificate and their lifecycle management will be supported through QR PKIs.

To materialise the QR data protection vision of PiQASO, two different types of evolving key chains will be created and maintained, namely an attribute/policy-specific and a user-specific chain, in order to support two modes of operation. These are the **Policy-specific** and **User-specific** modalities and, as depicted in Figure 3.3, are differentiated by whether they feature policy- or user-centric handling of key management. Their characteristics are detailed in the following sections.

3.3.2.1 Attribute-Based Encryption of Data Payload (policy-specific)

In this mode of operation, data sharing takes place between a specific data sender and any arbitrary number of entities (receivers), as long as the latter comply with a specific policy, defined by a trusted third party. Thus, any sender wishing to share data encrypted using the efficient PiQASO ABE variant under some policy/attributes, will obtain the currently valid corresponding public key(s) and any eligible receiver will be able to obtain appropriate decryption keys and data payloads if, and only if, they meet specific eligibility criteria reflected by its verifiable attributes.

Next, we provide further information on the operation of the ABE variant of the PiQASO PQCaaS. Once an (AES or ECC encrypted) Data Payload is provisioned to the PQ-DS, it proceeds in creating its quantum-secure equivalent by leveraging a different ML-KEM-based key pair provided by the Key Provider. For achieving an enhanced security level, a different key is used for creating multiple ciphertexts that can be associated to different identities (and/or attributes) that will be authorized to access each ciphertext. This process is governed by the end-user (as the data source) through the attribute- and identity-based policies circulated together with the encrypted Data Payloads to the PQ-DS. The PQ-DS then performs the **PQ**

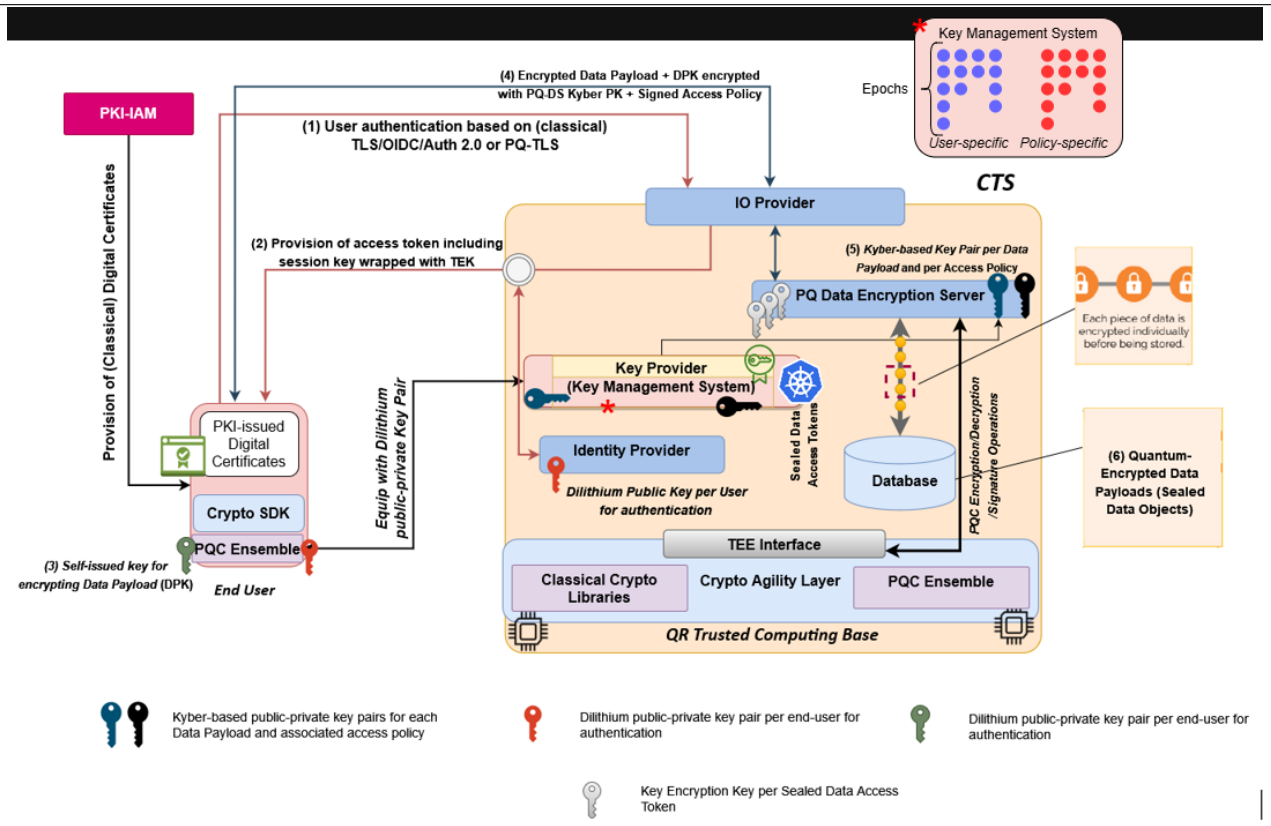

Figure 3.3: High-level overview of the PQCaaS component.

Data Encryption-at-rest operations and creates a Sealed Data Object (which is the quantum encryption of the Data Payload) and an associated Sealed Data Access Token. The leveraged ML-KEM-based key is wrapped using a **Key Encryption Key (KEK)**, which is saved as part of the Sealed Data Access Token which additionally also holds the identity (or attributes) of the authorized end-user that can access this encrypted data bundle. This means that only the target user(s) can prove ownership of the required identity or attributes can get access to the Sealed Data Access Token for then being able to decrypt the Data Payload with the appropriate (ML-KEM-based) key. Quantum-encrypted Data objects are decoupled and stored separately from the associated **Sealed Data Access Tokens**. This new key management and authentication process decouples the transmission of the encrypted Data Payloads and the keys used for the establishment of the respective secure and authenticated channel (used for the sharing) but also the key to be used for its quantum-based encryption. Besides enhanced programmability, this feature offers advanced immutability against outsider attackers that try to compromise the confidentiality of the encrypted data in-motion and at-rest.

3.3.2.2 Encryption/Decryption of Data Payloads with UPKE (user-specific)

In this mode, data sharing takes place between a specific user, functioning as the data receiver, and one or more data senders, utilizing time evolving user-specific key chains. This relies on the aforementioned AKRG scheme, which allows for the derivation of multiple Public Keys from a single Private Key. *This constitutes a core innovation of PiQASO, as the PiQASO Post-Quantum UPKE protocol enables secure end-to-end encrypted data sharing among multiple users, while providing resistance against chosen-ciphertext attacks (potentially with extra properties of circular security), ensuring also leakage resilience, and homomorphism capabilities for key/message spaces*. The UPKE scheme of PiQASO is an innovative PKE approach which offers forward secrecy through regular updates of public/private key pairs of data sharing parties.

Contrary to traditional PKE, where an entity holds a lifetime key pair that entails no post-compromise security, UPKE offers a secure way to refresh the key pair after certain time intervals (epochs) ensuring that even under a potential compromise of future private keys, confidentiality of ciphertexts encrypted under earlier public keys is not breached. The PQ UPKE will provide forward secure end-to-end encrypted data sharing amongst multiple users, leveraging post-quantum secure **Asynchronous Remote Key Generation (AKRG)**, which is a recent technique which allows to asynchronously derive multiple unlinkable public keys from some given (base) public key. The scheme offers "insider security" capable of resisting chosen-ciphertext attacks, potentially with extra properties of circular security, leakage resilience, and homomorphism for key/message spaces. PQCaaS is a central component of the data sharing infrastructure of PiQASO, as it supports UPKE key generation, management and distribution of key updates (through epochs) among data points, as well as data hosting and sharing.

3.4 Functional Encryption

As previously analysed, *the vision of PiQASO entails the ability to support the execution of data sharing agreements in a secure and privacy-preserving manner*. To this end, a core envisioned functionality that supports this vision is **the extension of the PQC portfolio of the PQCaaS modality with Functional Encryption capabilities**. The motivation behind the inclusion of Functional Encryption, which is analysed in detail in Chapter 6, is to enable carrying out computations on encrypted data, revealing only the result of these computations and avoid leakage of any sensitive information.

This is particularly pertinent in the PiQASO use cases which revolve around the sharing and management of sensitive information, such as the ones belonging to the **healthcare domain**. In particular, PiQASO envisions two such use cases, namely *Remote Patient Monitoring Intelligent System for Supporting Independent and Safe Living* and *Smart Ambulance*. The common denominator in these cases is the **need to share data in the backend service of the healthcare delivery organization (e.g., hospital), in order to enable monitoring the treatment of the patient**. For instance, this may apply to ECG data from monitoring the heart rate of a patient. In this case, Functional Encryption can be used in order to determine whether any recorded values match or exceed a threshold value, in order to determine whether an alarm should be raised signifying the need for further medical attention.

Considering all the above, PiQASO provides several cryptographic primitives to secure data in transit and at rest. Namely, Functional Encryption is used together with **Attribute-Based Encryption** in order to enable fine-grained control over data access. thus enabling secure computation in sensitive applications such as healthcare and surveillance systems. In these cases, a **cloud service provider (CSP)** must perform computations on sensitive data and return precise results to end users without accessing the underlying data. Only modern cryptographic primitives—**Homomorphic Encryption (HE)** and **Functional Encryption (FE)**—enable such computations without exposing sensitive data to the CSP.

High-level overview of HE and FE Consider a three-party setting: a **Data Owner (DO)** encrypts her data and outsources it to a **Cloud Service Provider (CSP)**. An **Analyst (A)**—such as a doctor—requests a specific computation from the **CSP**. The goal is to deliver the computation result without sharing the **DO**'s data with the **CSP** or **A**. The result itself must also remain hidden from the **CSP**.

- ***HE approach.*** The **A** generates a key pair and shares the public key with the **DO** and **CSP**. The **DO** then encrypts the data with this public key and outsources it to the **CSP**. Upon receiving a computation request, the **CSP** evaluates the function on the encrypted data and returns the encrypted result to the **A**, who alone can decrypt it. HE thus supports flexible computations on

encrypted data. However, it suffers from an "all-or-nothing" drawback: the **A** can decrypt not only the result but also the underlying data from the **DO** by having the encrypted data.

- ***FE approach.*** Most FE schemes rely on a trusted **Key Curator (KC)** to generate the master secret key, master public key, and functional keys, and to register users. The **DO** encrypts her data with the master public key and outsources it to the **CSP**. The **A** encrypts the desired function `f()` with the same master public key and requests functional keys from the **KC** tied to `f()` and the relevant user identifier. The **A** then obtains the encrypted data from the **CSP** and decrypts only the result of `f()`. In this setup, the **CSP** performs two tasks: *(i)* hosting the **KC** in a trusted zone, and *(ii)* storing and distributing encrypted data on request. FE avoids the "all-or-nothing" problem but requires a trusted **KC**.

To combine the strengths of these primitives into the PiQASO framework and enable computation over encrypted data, we introduce a post-quantum secure FE scheme. This scheme relies on **lattice-based security assumptions and Learning With Errors (LWE),** and it incorporates HE parameters and best practices. We analyse this scheme in detail in Chapter 6.

3.5 PiQASO's Crypto Asset Inventorying Toolkit

As part of PiQASO's quantum resilience solution for security systems and beyond, the first step to be employed is the composition of a full-blown "Crypto Bill-of-Material" enabling the identification and monitoring of the cryptographic posture of a system so as to better facilitate the planning and migration to PQC by selecting the most appropriate crypto algorithms to be instantiated, offering the following capabilities: (i) **Discovery** – Real-time monitoring and tracing of all events, on communication channels between systems comprising assets, so as to identify the type of (classical) cryptosystems and protocols used and create crypto asset inventories including information on running processes, crypto algorithms used, key sizes (number of keys pairs used throughout the system lifetime), certificates, libraries so as to better capture those components that will be mostly impacted by the advent of quantum attacks and need to be PQC compliant, i.e., dealing with the authentication, authorization, data protection, key management requirements that need to be quantum-resistant; (ii) **Security Analysis** – Testing the security level of each identified crypto asset through crawling and matching specific keywords capturing the possible use of crypto primitives that have been identified as vulnerable to quantum-computing attacks; e.g., the use of ECC-based crypto for instance. Furthermore, this will also focus on mapping similar weak communication APIs based on the use of high-end security protocols that can be exploited - e.g., send commands to a device's key management interface to verify its resilience against prominent PKCS11 API attacks; (iii) **PQC Crypto Agility** – Orchestrating the operational deployment of the best-suited PQC implementations that offer the highest degree of resilience and programmability, to the target system, with the smallest possible level of disruption, enabling strict compliance in the target environment.

Towards this directon, the proposed Cryptographic Asset Inventory (CAI) Toolkit is envisioned not merely as a static utility, but as a dynamic, evolving framework meticulously engineered to address the sophisticated visibility requirements identified in our previous analysis. To meet the urgent demands of post-quantum readiness, our architectural vision follows a phased implementation roadmap. In the initial delivery cycle, the toolkit will prioritize the most critical vectors of exposure: live network environments and the software supply chain. This foundational period focuses on the deployment of robust Network Scanning and Software Composition Analysis capabilities, specifically designed to generate comprehensive Cryptographic Bill of Materials (CBOM) and Software Bill of Materials (SBOM) datasets. Consequently, the CAI Toolkit is engineered as a dynamic framework to operationalize the audit and mapping phase of post-quantum migration, directly supporting the Cryptographic Agility requirements identified in Section

2.2.1. Unlike standard IT tools that are often proprietary and limited in their transparency, the CAI toolkit provides an accessible, community-driven alternative designed to identify undocumented cryptographic implementations across a tri-layered architecture.

Layer 0: High-End Security Protocols at a Network Level

At the network layer, the toolkit operates as a non-intrusive observer of Transport Layer Security (TLS) communications, passively capturing and analyzing API traffic packets to derive actionable intelligence. Through detailed profiling of the TLS handshake, across both real-time traffic flows and historical network captures. The system extracts critical cryptographic parameters, including key exchange mechanisms, key sizes, and negotiated cipher suites.

These elements are systematically classified according to their dependence on legacy classical algorithms, contemporary hybrid configurations, or emerging post-quantum primitives. This granular visibility enables precise identification of externally exposed assets that may be susceptible to “harvest-now, decrypt-later” threat models. Additionally, the platform provides the capability for manual inspection, allowing security teams to review cryptographic configurations per device and assess or initiate migration strategies directly at the endpoint level.

Layer 1: Application Logic at the Software Level

Simultaneously, the toolkit addresses the complexities of the software ecosystem through deep system-level analysis. By examining installed software packages and their intricate dependency chains, the CAI toolkit uncovers the ”shadow cryptography” that is often inherited through third-party libraries. In modern development environments, cryptographic functions are frequently buried deep within nested dependencies, remaining invisible to traditional manual audits. By generating high-fidelity SBOMs and CBOMs during this first phase, the toolkit provides an exhaustive inventory of the software supply chain, highlighting insecure or deprecated components that pose a systemic risk to the organization’s integrity.

Layer 2: Security function at the Device Level

Following the successful stabilization of these core modules, the toolkit’s roadmap envisions an expansion into the domains of authentication, identity, and application-specific logic. This subsequent phase will extend inventorying capabilities to the often-neglected realm of SSH keys. By extracting and analyzing algorithm types and bit lengths of keys residing within user environments, the toolkit will shine a light on long-lived administrative assets. These credentials represent a significant risk in a post-quantum landscape, as quantum-vulnerable authentication keys could grant persistent, unauthorized access to critical infrastructure long after other protections have been modernized.

Furthermore, the toolkit will evolve to include application-layer inspection, moving beyond binary analysis to the scrutiny of source code repositories. This allows the toolkit to map the specific cryptographic libraries and functions invoked by developers, identifying instances where cryptographic logic is “hard-coded” or tightly coupled with the application’s core business logic. Such insights are indispensable for architects who must estimate the human effort required for migration and design transition roadmaps that avoid breaking essential services.

The overarching strength of this envisioned CAI toolkit lies in its commitment to structured, machine-readable reporting. Rather than producing static documents, the toolkit generates actionable data schemas that document every identified asset alongside its specific risk profile and quantum-readiness score. This data-centric approach ensures that the toolkit supports continuous monitoring and remains in strict alignment with the evolving standards currently being drafted by international bodies such as NIST and ETSI.

By operationalizing crypto asset inventorying as a perpetual, automated process, the toolkit moves the organization away from reactive “point-in-time” assessments and toward a state of proactive cryptographic agility. Ultimately, this toolkit serves as the essential bridge between theoretical risk and operational execution, providing the evidence-based foundation required to navigate the transition to a post-quantum future with confidence and precision.

Chapter 4

State-of-the-Art in Post-Quantum Crypto

In continuation to the description of the PiQASO vision set forth in Chapter 2, the focus of this Chapter is on detailing the **state-of-the-art** in the design of the types of PQC algorithms and constructions investigated by PiQASO, with particular focus on lattice-based constructions. Note that, at the time of writing of this deliverable, we acknowledge the latest advancements in the domain of lattice-based constructions, based on which the level of security offered by the **Ring-Learning With Errors (RLWE)** variant may be compromised. However, this does not directly impact other modalities of lattice-based structures. Thus, as will be detailed in Chapters 5 and 6, the PiQASO PQC Ensemble employs ideal lattices and Module LWE (MLWE) structures, thus remaining up-to-date with the latest advancements in terms of offered security level. Finally, based on the state-of-the-art in all considered domains, Section 4.7 introduces an overview of the considered PiQASO threat model based on the identified threat landscape.

4.1 Introduction to Ideal Lattices and Learning with Errors

4.1.1 Motivations

Post-quantum motivation Contemporary cryptographic systems, often termed classical cryptography, rely on the presumed computational hardness of problems such as integer factorization and the discrete logarithm problem (DLP) in finite cyclic groups. Notable examples include the RSA cryptosystem [280], the Diffie–Hellman key exchange protocol [104], and cryptographic schemes based on elliptic curves over finite fields [160].

The security of these cryptographic constructions is increasingly threatened by advances in quantum computing. In 1994, Shor introduced a quantum algorithm [291] that efficiently solves both the integer factorization problem and the discrete logarithm problem. If large-scale quantum computers become available, Shor's algorithm would compromise the security of many currently deployed public-key cryptosystems.

Although quantum computers capable of executing such algorithms do not yet exist, ongoing advancements in quantum hardware make this threat increasingly realistic. Consequently, there is a growing need to develop cryptographic systems that remain secure against quantum adversaries.

This context has led to the emergence of *post-quantum cryptography (PQC)*, which does not rely on quantum computation but instead utilizes alternative mathematical structures believed to be resistant to both classical and quantum attacks.

Many research directions are currently explored, including lattice-based cryptography, code-based cryptography, multivariate cryptography, and hash-based cryptography. Among these approaches, lattice-

Co-funded by the European Union
ECCC

based cryptography is one of the most extensively studied fields and has emerged as a promising candidate for post-quantum security.

Lattices A lattice is an algebraic structure that defines an infinite set of point in an n-dimensional space. For cryptographic purposes, n is chosen very large to ensure security, but it helps to visualise a 2-dimensional lattice as a structured set of points in an infinite plan – hence its name. For a formal description, we refer the reader to the following definition 1.

Definition 1 (n-dimensional lattice)**.** *Let $\mathbf{B} = (b_1, \ldots, b_k) \in \mathbb{R}^{n \times k}$ be linearly independent vectors in $\mathbb{R}^n$. The lattice generated by $\mathbf{B}$ is the set $\mathcal{L}(\mathbf{B}) = \{\mathbf{B} \cdot \mathbf{x} : \mathbf{x} \in \mathbb{Z}^k\}$. $\mathbf{B}$ is a matrix that generates the lattices, and is hence called a basis of $\mathcal{L}(\mathbf{B})$. The integers n and k are called dimension and rank of the lattice, respectively.*

What makes lattices a good candidate for cryptographic primitives is that, for well chosen parameters, it is extremely hard, given a random point in the plan, to retrieve its closest point in the lattice. This problem is called Closest Vector Problem (CVP), and is stated as follows:

Definition 2 (Closest Vector Problem (CVP))**.** *Let $\mathcal{L}(\mathbf{B})$ be an n-dimensional lattice and $\mathbf{v} \in \mathbb{R}^n$ be a random vector in the plan. Find the vector generated by $\mathbf{B}$ that is the closest to $\mathbf{v}$.*

A similar hard problem that can be defined on lattices is the Shortest Vector Problem (SVP) that consists in finding the shortest non-zero vector of the lattice. CVP and SVP are the first and main two problems defined on lattices, and most recent hardness assumptions such as Learning With Error (LWE) reduce to CVP or SVP.

Definition 3 (Shortest Vector Problem (SVP))**.** *Let $\mathcal{L}(\mathbf{B})$ be a n-dimensional lattice. Find a non-zero vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that: $\|\mathbf{v}\| = \min_{\mathbf{w} \in \mathcal{L}(\mathbf{B}) \setminus \{\mathbf{0}\}} \|\mathbf{w}\|$.*

Why lattices Lattices are extensively studied due to their versatility and efficiency, enabling the construction of advanced cryptographic primitives such as Homomorphic Encryption (HE), Functional Encryption (FE), and Attribute-Based Encryption (ABE).

Early lattice-based cryptographic constructions relied on the hardness of lattice problems, such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP).

One of the first lattice-based cryptosystems, called the GGH Cryptosystem [144], was created in 1997 by Oded Goldreich, Shafi Goldwasser, and Shai Halevi.

Another important work was presented by Oded Regev in 2005 with the Learning With Errors problem [273].

In 2009, Lyubashevsky, Peikert, and Regev introduced an algebraic variant of LWE in their work on ideal lattices and learning with errors over rings [273], leading to the development of improved and structured lattices: Ring-LWE and Module-LWE. These variants aim to enhance efficiency while maintaining strong security guarantees. Many contemporary post-quantum cryptographic schemes are now based on these constructions.

This field of research has also contributed to the standardization of lattice-based cryptographic algorithms. Notably, the key encapsulation mechanism CRYSTALS-Kyber [61], which is based on the Module-LWE assumption, and the digital signature scheme CRYSTALS-Dilithium [116], which relies on the hardness of lattice problems over Module-LWE and was introduced in 2020, were selected for the NIST post-quantum cryptography standardization process.

Precisions on Ideal Lattices and Module Structures Initially, lattice-based cryptographic constructions relied on general lattices, evolving into modern schemes that use structured lattices to improve efficiency and reduce an important parameter: key size. In particular, these structured lattices are often derived from algebraic objects such as polynomial rings, leading to the notion of ideal lattices.

In general, an ideal lattice can be viewed as the lattice associated with an ideal of a polynomial ring. This additional algebraic structure enables more compact representations and more efficient arithmetic operations, especially when using fast polynomial multiplication techniques such as the Number Theoretic Transform (NTT).

The two main problems arising from this idea are Ring-LWE [214] and Module-LWE [199], which extend the LWE framework to structured algebraic settings. These two variants preserve the theoretical security of the LWE foundation while significantly improving computational efficiency and reducing parameter sizes.

4.1.2 Learning With Errors (LWE)

The Learning With Errors (LWE) was introduced by Regev [273]. It can be seen as the problem of solving a seemingly overdetermined system of linear equations modulo q in the presence of small noise.

Let $n, m \geq 1$ be integers, $q \geq 2$ be a modulus, and χ be an error distribution over $\mathbb{Z}$ (typically a discrete Gaussian distribution). For a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, the LWE distribution $A_{\mathbf{s},\chi}$ is obtained by sampling a uniformly random vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an error $e \leftarrow \chi$, and outputting the pair $\Big(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q\Big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

There are two main variants of the LWE problem:

- **Search LWE:** Given m independent samples $(\mathbf{a}_i, b_i)$ drawn from $A_{\mathbf{s},\chi}$, find the secret vector $\mathbf{s}$.
- **Decision LWE:** Distinguish between m independent samples drawn from $A_{\mathbf{s},\chi}$ and m samples drawn uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Regev showed a quantum reduction from worst-case lattice problems (like approximate SVP) to average-case LWE, demonstrating that solving LWE on average is as hard as solving certain lattice problems in the worst case. More details on LWE-based schemes and its variants are presented in Section 6.2.

The security and efficiency of LWE-based schemes were recently demonstrated by their selection in the NIST standardisation process [61, 116]. Furthermore, LWE supports a broad range of cryptographic applications, including: Inner-Product Encryption from Middle-Product LWE(MP-LWE) [316] addresses the issue of oversized public keys and ciphertexts by proposing an MP-LWE-based solution that relaxes restrictions on polynomials and yields a better balance between security and efficiency. Inner-Product Encryption from Ring-LWE [129] focuses on optimising key sizes and enhancing encryption efficiency. Functional Encryption (FE) for Inner-Product Predicates from LWE [10] extends LWE techniques to FE, enabling more flexible computations. Additionally, Attribute-Based Access Control for Inner-Product Functional Encryption from LWE [10] introduces a construction of an attribute-based FE scheme in a key-policy setting under the LWE assumption.

However, the field still faces several limitations, such as noise growth, scalability challenges, and large key sizes [259, 129, 10]. In practical applications, cryptographic libraries such as Microsoft SEAL [289] and OpenFHE [15] these techniques, but they continue to employ an all-or-nothing decryption model and do not support partial decryption. This work addresses this gap by introducing partial decryption within the LWE framework, an area that remains largely unexplored.

4.1.3 Ring-Learning With Errors (RLWE)

Introduced by Lyubashevsky *et al.* [214], RLWE is a structured variant of LWE that improves efficiency by operating over polynomial rings rather than over integer vectors. While traditional LWE parameters are integers used to encrypt a single bit, RLWE operates over a polynomial ring $\mathcal{R}_q := \mathbb{Z}_q[X]/(P(X))$ where $P(X)$ is typically a cyclotomic polynomial. A common instantiation of RLWE uses the cyclotomic polynomial $P(X) = X^N + 1$, which is secure when N is a power of 2, ensuring that $X^N + 1$ is irreducible. It is a significant performance improvement, allowing a broader use of lattice-based encryption in PKE and FE. This is particularly beneficial in scenarios where using standard LWE parameters is infeasible. This advantage is especially relevant for HE schemes [68, 128], whose evaluation algorithms are computationally intensive.

4.1.4 Module Learning With Error (MLWE)

Module-LWE, introduced by Langlois et al. [199], generalizes both LWE and RLWE by considering module lattices over polynomial rings. While RLWE operates over a single ring element, Module-LWE extends this setting to vectors of ring elements. Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, where N is typically a power of two. In the Module-LWE setting, samples are drawn from $\mathcal{R}_q^\kappa$, where κ denotes the rank of the module.

This construction can be seen as interpolating between RLWE and standard LWE: when $\kappa = 1$, Module-LWE reduces to RLWE, while larger values of κ decrease the algebraic structure and bring the problem closer to standard LWE. As a result, Module-LWE provides a possible trade-off between efficiency and security.

In practice, implementations typically fix the ring parameters such as N and q to enable optimized polynomial arithmetic, and the adjusted rank κ can be changed for different security levels. This approach retains much of the efficiency of RLWE while offering a more conservative security assumption.

Module-LWE is notably used in practical constructions such as Kyber, selected in the NIST post-quantum standardization process.

4.2 Key Encapsulation Mechanisms

4.2.1 Introduction

Key Encapsulation Mechanisms (KEMs) are cryptographic primitives designed to securely establish a shared secret between two parties over an insecure communication channel. Informally, a KEM allows one party to encapsulate a randomly generated symmetric key using the public key of another party, such that only the intended recipient—holding the corresponding secret key—can decapsulate and recover the shared key.

KEMs are a fundamental building block in modern cryptographic systems because they enable the secure use of efficient symmetric cryptography without requiring a pre-shared secret. In practice, they are typically used as part of hybrid encryption schemes, where public-key techniques are employed only for key establishment, while symmetric encryption is used for protecting bulk data.

Early public-key encryption schemes, such as RSA [280] and ElGamal [122], were originally designed to encrypt messages directly. However, public-key operations are computationally expensive and inefficient for large amounts of data. Moreover, many real-world protocols require only the establishment of a shared secret rather than general-purpose message encryption.

Therefore, the cryptographic community adopted hybrid encryption. In this approach, public-key cryptography is used only to protect a secret key for a symmetric-key encryption scheme. This secret key is a fixed-length string—much shorter than typical messages—while the actual message is encrypted using the symmetric scheme under this key, yielding a significantly more efficient construction overall.

4.2.1.1 KEMs in the Pre-Quantum Era and Transition to Post-Quantum KEMs

In the classical setting, widely deployed KEMs are based on number-theoretic assumptions such as integer factorization and the discrete logarithm problem. Representative examples include RSA-based KEMs and Diffie–Hellman-based constructions, particularly those using elliptic curves. These schemes have been extensively studied and deployed in major security protocols such as TLS [276], SSH [210], and secure messaging systems.

Despite their maturity and efficiency, these constructions rely on mathematical problems that are efficiently solvable by quantum computers using Shor's algorithm [292], posing a serious threat to their long-term security.

The prospect of large-scale quantum computers has motivated the development of post-quantum cryptography (PQC), which aims to design cryptographic schemes secure against both classical and quantum adversaries. Within this context, KEMs have become the primary focus, as secure key establishment is a critical requirement for most communication protocols.

Post-quantum KEMs are typically based on hardness assumptions that are supposed to be resistant against quantum adversaries, including lattice-based, code-based, and multivariate polynomial problems. Among these, lattice-based constructions have emerged as particularly promising due to their strong theoretical foundations, efficiency, and flexibility.

This research effort culminated in the NIST Post-Quantum Cryptography standardization process [237], which selected ML-KEM, derived from the CRYSTALS-Kyber scheme [61], as the standardized post-quantum KEM. ML-KEM represents the current state of the art, offering strong security guarantees, efficient implementations, and suitability for large-scale deployment.

4.2.2 Lattice-Based KEMs

Among lattice-based key encapsulation mechanisms, some of the most prominent proposals include CRYSTALS-Kyber, NTRU [166], SABER [98], and FrodoKEM [142]. These schemes rely on different lattice assumptions, ranging from structured variants such as module-LWE and NTRU lattices to the more conservative standard LWE assumption used by FrodoKEM. CRYSTALS-Kyber, which is based on the module-LWE problem, was the one selected for standardization due to its balance between security, efficiency, and implementation simplicity. At the same time, Kyber offers strong performance across software and hardware platforms and supports efficient, constant-time implementations, making it particularly well-suited for deployment in real-world protocols.

4.2.2.1 ML-KEM

Design ML-KEM is based on the hardness of the Module-LWE problem over polynomial rings modulo a prime modulus, using small, centered noise distributions. Its security relies on the hardness of Module-LWE (MLWE) for the selected parameters and the security of the FO transform in the random oracle model. It is standardized in three parameter sets—ML-KEM-512, ML-KEM-768, and ML-KEM-1024—corresponding approximately to NIST security levels 1, 3, and 5.

Efficiency and Performance From a theoretical perspective, the sizes of ML-KEM keys and ciphertexts are determined by a small set of fixed public parameters and a single variable parameter, namely the module rank k. All standardized ML-KEM instances operate over the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ with fixed parameters $n = 256$ and $q = 3329$. The public encapsulation key (ek) consists primarily of a seed describing a uniformly random public matrix together with a vector of k ring elements, while the decapsulation key (dk) contains the secret vector, a copy of the public key, and auxiliary hash values required by the CCA-secure transform. Ciphertexts similarly consist of a vector of k ring elements and one additional ring element. Consequently, the sizes of keys and ciphertexts scale linearly with the module rank k, while the size of the encapsulated shared secret is fixed at 32 bytes for all ML-KEM instances as specified in FIPS 203 [237].

All three instances use the same ring parameters $n = 256$ and $q = 3329$, and performance differences arise solely from the module rank k (with $k = 2, 3, 4$ for ML-KEM-512, ML-KEM-768, and ML-KEM-1024, respectively). Key generation is dominated by matrix-vector multiplication over R_q, requiring the computation of $t = A \cdot s + e$, and thus incurs a cost proportional to k^2 polynomial multiplications and associated Number-Theoretic Transform (NTT) operations. Encapsulation is similarly dominated by polynomial matrix-vector and inner-product computations, in particular the evaluation of $A^T \cdot r$ and $t^T \cdot r$, again yielding a cost proportional to k^2 polynomial multiplications and NTTs. Decapsulation additionally performs an implicit re-encapsulation step as part of the CCA-secure FO transform, but its overall cost remains proportional to k^2 polynomial multiplications and NTT operations [38].

Concrete key and ciphertext sizes for standardized ML-KEM instances The resulting concrete sizes of encapsulation keys, decapsulation keys, and ciphertexts for the three standardized ML-KEM instances are summarized in Table 4.1. These values are fixed by FIPS 203 and illustrate the linear growth in size with increasing security level. For all instances, the encapsulated shared secret has a fixed length of 32 bytes [237].

Table 4.1: Sizes of keys and ciphertexts for ML-KEM instances (FIPS 203)

ML-KEM instance	Public key (bytes)	Private key (bytes)	Ciphertext (bytes)
ML-KEM-512	800	1632	768
ML-KEM-768	1184	2400	1088
ML-KEM-1024	1568	3168	1568

Concrete performance measurements in software In addition to asymptotic analysis, extensive benchmarking results are available for optimized ML-KEM software implementations. On modern x86-64 processors, reference and optimized implementations derived from the CRYSTALS-Kyber codebase and its successors report key generation costs of approximately 30-50 thousand CPU cycles for ML-KEM-512, 50-80 thousand cycles for ML-KEM-768, and 80-120 thousand cycles for ML-KEM-1024 when AVX2 optimizations are enabled [95, 254]. Encapsulation is typically slightly faster than key generation, while decapsulation is the most expensive operation due to the additional re-encapsulation step required for CCA security, with reported costs ranging from roughly 40-70 thousand cycles for ML-KEM-512 up to 100-150 thousand cycles for ML-KEM-1024 [265, 258]. On ARM platforms, similar relative performance trends are observed in NEON-optimized implementations, albeit with higher absolute cycle counts [254]. Overall, these measurements indicate that ML-KEM supports high-throughput and low-latency use cases such as TLS handshakes, and that its performance overhead compared to classical key establishment mechanisms is dominated by increased bandwidth rather than computational cost.

4.2.2.2 State of the art in ML-KEM implementation

Software implementations in general-purpose cryptographic libraries Following its standardization as ML-KEM in FIPS 203, the scheme has rapidly transitioned from a research prototype into a production-ready primitive supported by several widely deployed general-purpose cryptographic libraries. Most notably, OpenSSL provides first-class ML-KEM support through its native KEM and EVP interfaces, enabling direct use of ML-KEM encapsulation and decapsulation operations in TLS, PKI tooling, and application-level cryptography without reliance on external providers [258, 237]. In parallel, BoringSSL has incorporated ML-KEM—with particular emphasis on ML-KEM-768—to support large-scale experimentation and early production deployments in the web ecosystem, especially in hybrid key establishment mechanisms [148]. Beyond TLS-centric libraries, Botan and wolfSSL/wolfCrypt provide production-grade ML-KEM implementations aimed at portability and deployment in embedded or constrained environments, often in combination with classical key establishment algorithms [65, 312]. In managed-language ecosystems, Bouncy Castle supports ML-KEM in both its Java and .NET distributions, facilitating adoption in enterprise middleware and application frameworks [66]. Similarly, Microsoft's integration of ML-KEM into its SymCrypt library signals readiness for deployment at operating-system and cloud-platform scale [227]. Transitional libraries such as liboqs and the oqs-provider for OpenSSL continue to play an important role in interoperability testing, benchmarking, and early adoption, and have strongly influenced API design and implementation practices across the ecosystem [255, 256].

Side-channel resistance and secure implementation techniques While the cryptographic security of ML-KEM relies on the assumed hardness of the MLWE problem, practical security critically depends on resistance to side-channel attacks. A substantial body of recent work has demonstrated timing, microarchitectural, power, and electromagnetic leakage vulnerabilities in insufficiently hardened implementations of Kyber and ML-KEM [54]. In response, state-of-the-art implementations emphasize constant-time execution, avoidance of secret-dependent memory access patterns, and hardened decapsulation procedures that prevent key recovery under chosen-ciphertext and leakage-based attack models. Production libraries such as OpenSSL, BoringSSL, and wolfCrypt explicitly aim to provide constant-time ML-KEM encapsulation and decapsulation, informed by lessons learned from prior side-channel analyses [258, 148, 312]. High-performance implementations such as `mlkem-native` explicitly incorporate side-channel countermeasures, including constant-time polynomial arithmetic and hardened comparison logic [265]. Side-channel resistance therefore remains an active and evolving aspect of ML-KEM implementation research and engineering.

Formally verified and high-assurance implementations In addition to conventional testing and code review, a growing line of work focuses on formally verified and high-assurance implementations of ML-KEM. These efforts aim to provide machine-checked guarantees of functional correctness, memory safety, and, in some cases, constant-time behavior for critical components of the encapsulation and decapsulation algorithms. Notable examples include implementations developed within the PQ Code Package and related efforts, which combine carefully audited C code with formally verified or mechanically checked assembly routines for performance-critical arithmetic operations [264, 265]. Although such implementations are not always intended for direct deployment, they serve as high-assurance reference implementations and as building blocks for production libraries. This trend reflects a broader shift in post-quantum cryptography toward higher assurance levels, motivated by the complexity of lattice-based arithmetic and the long-term security requirements associated with post-quantum standards.

Hardware optimizations and acceleration Hardware optimization and acceleration constitute another important dimension of the ML-KEM implementation landscape. On general-purpose processors, state-

of-the-art software implementations exploit vector instruction sets such as AVX2 and AVX-512 on x86 platforms and NEON on ARM architectures to accelerate NTT and polynomial arithmetic, which dominate the cost of ML-KEM encapsulation and decapsulation [265, 255]. Beyond software optimization, ML-KEM has also been implemented in dedicated hardware, including FPGA and ASIC prototypes, targeting applications such as secure communication endpoints and embedded systems [163, 217]. These hardware designs explore trade-offs between throughput, latency, area, energy consumption, and side-channel resistance. Collectively, these results demonstrate that ML-KEM can be efficiently realized across a wide range of platforms, from high-performance servers to resource-constrained embedded devices.

4.3 Digital Signatures

4.3.1 Introduction

Digital signature schemes are cryptographic primitives designed to provide authenticity, integrity, and non-repudiation for messages. Informally, a digital signature scheme allows a signer to produce a short piece of data—called a signature—using a secret signing key, such that anyone holding the corresponding public verification key can verify that the signature was generated by the legitimate signer and that the signed message has not been altered.

Digital signatures are a fundamental building block in modern cryptographic systems because they enable secure authentication and data integrity in open and adversarial environments. In practice, they are widely used in applications such as secure communication protocols (TLS) [276] or contract signing [261], where trust must be established without relying on shared secrets.

As for KEMs, early digital signature schemes, such as DSA [234], were designed under number-theoretic hardness assumptions, including the difficulty of integer factorization and the discrete logarithm problem.

This motivated the development of post-quantum digital signature schemes, among which lattice-based constructions play a prominent role. These constructions rely essentially on the same hardness assumptions as lattice-based KEMs.

4.3.2 Lattice-Based Digital Signatures

Among lattice-based digital signature schemes, some of the most prominent proposals include CRYSTALS-Dilithium [116], Falcon [131], and qTESLA [21]. These schemes rely on different lattice assumptions, ranging from structured variants such as module-LWE and NTRU lattices to constructions based on the standard Learning With Errors problem. CRYSTALS-Dilithium, which is based on the module-LWE and module-SIS problems, was selected for standardization due to its favorable trade-off between security, efficiency, and implementation robustness. In particular, Dilithium avoids complex arithmetic such as floating-point operations, enabling straightforward, constant-time implementations, and demonstrates strong performance across a wide range of software and hardware platforms, making it well-suited for deployment in real-world cryptographic systems.

Lattice-based digital signatures are typically built upon the Short Integer Solution (SIS) problem and its structured variants. Modern constructions aim to avoid costly Gaussian sampling and to tightly control information leakage during signing.

In this environment, two dominant design paradigms have emerged: Fiat–Shamir with aborts [103] and hash-and-sign constructions with advanced sampling techniques (see e.g., [181]).

4.3.2.1 ML-DSA

ML-DSA, derived from CRYSTALS-Dilithium [116], is the NIST-standardized lattice-based digital signature algorithm. It follows a Fiat–Shamir-with-aborts paradigm, emphasizing simplicity, ease of analysis, and the avoidance of Gaussian sampling. Security is proven in the random oracle model.

The standard specifies multiple parameter sets, commonly denoted ML-DSA-44, ML-DSA-65, and ML-DSA-87, corresponding to increasing security levels.

Cryptographic security ML-DSA bases its security on the presumed hardness of the Module Learning With Errors (Module-LWE) and Module Short Integer Solution (Module-SIS) problems over structured polynomial modules. As with other lattice-based constructions, these problems inherit security intuition from the Learning With Errors framework, for which reductions from worst-case lattice problems such as GapSVP and SIVP were established by Regev, providing evidence of hardness even against quantum adversaries [273]. Cryptanalytic evaluation of ML-DSA therefore primarily focuses on assessing the cost of lattice reduction attacks against the underlying Module-LWE and Module-SIS instances, including primal, dual, and hybrid attacks based on BKZ lattice reduction combined with enumeration or sieving. Security estimates rely on heuristic lattice cost models and are typically evaluated using tools such as the lattice estimator developed and maintained by Albrecht et al. [20, 18].

To date, no cryptanalytic result has demonstrated a practical break of any standardized ML-DSA instance (ML-DSA-44, ML-DSA-65, or ML-DSA-87) under widely accepted attack models, and the parameters selected by NIST are considered to provide comfortable security margins according to current lattice attack estimates [236]. Overall, ML-DSA is widely regarded as the state of the art among lattice-based digital signature schemes.

Efficiency and Performance From a theoretical perspective, the sizes of ML-DSA public keys, private keys, and signatures are determined by fixed ring parameters and a small number of scheme-specific parameters that control the dimension of the underlying module lattice and the bounds on secret and error distributions. All standardized ML-DSA instances operate over the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ with $n = 256$ and a fixed modulus q, similarly to ML-KEM. The public verification key consists primarily of a vector of ring elements derived from a public matrix and a secret vector, while the signing key additionally contains short secret vectors and auxiliary hash values required for the Fiat–Shamir-with-aborts signing procedure. Signatures consist of multiple vectors of ring elements along with challenge information. As a result, public key, private key, and signature sizes scale roughly linearly with the module rank and the selected security level.

All ML-DSA instances use the same polynomial ring dimension, and differences in performance arise primarily from parameter choices that affect the number of polynomial operations and rejection sampling iterations. Key generation and signing are dominated by polynomial matrix-vector multiplications, sampling from discrete distributions, and repeated norm checks inherent to the Fiat–Shamir-with-aborts paradigm, resulting in an overall complexity proportional to a small constant multiple of polynomial multiplications and NTT operations. Signature verification is comparatively less expensive, as it involves fewer polynomial multiplications and no rejection sampling, yielding lower computational cost than signing.

Concrete key and signature sizes for standardized ML-DSA instances The concrete sizes of public keys, private keys, and signatures for the three standardized ML-DSA instances are fixed by FIPS 204 and are summarized in Table 4.2. As expected, sizes increase with higher security levels, with ML-DSA-44 targeting NIST security level 2, ML-DSA-65 targeting level 3, and ML-DSA-87 targeting level 5 [236].

Table 4.2: Sizes of keys and signatures for ML-DSA instances (FIPS 204)

ML-DSA instance	Public key (bytes)	Private key (bytes)	Signature (bytes)
ML-DSA-44	1312	2560	2420
ML-DSA-65	1952	4032	3293
ML-DSA-87	2592	4896	4595

Concrete performance measurements in software Concrete performance measurements indicate that ML-DSA offers efficient signing and verification on modern platforms. Optimized software implementations derived from the CRYSTALS-Dilithium codebase report signing costs on the order of 100-200 thousand CPU cycles for ML-DSA-44, increasing to approximately 200-400 thousand cycles for ML-DSA-87 on x86-64 processors with AVX2 optimizations enabled [41, 95]. Verification is significantly faster, typically requiring less than half the number of cycles needed for signing. ARM-based implementations using NEON instructions exhibit similar relative performance trends, albeit with higher absolute cycle counts [254]. These results demonstrate that ML-DSA is suitable for high-volume signature verification workloads and practical deployment in authentication, software update, and certificate validation scenarios.

Software implementations in general-purpose cryptographic libraries Since its standardization as ML-DSA in FIPS 204, the scheme (derived from CRYSTALS-Dilithium) has been implemented in a wide range of general-purpose cryptographic libraries and toolkits. OpenSSL and its provider ecosystem offer ML-DSA support either natively or via PQ provider modules, enabling straightforward use of ML-DSA public-key operations and signature verification in TLS, code signing, and application-level authentication workflows [258, 236]. BoringSSL and other TLS-focused stacks have also incorporated Dilithium-based signatures for experimental and early production use, particularly where hybrid signature strategies are required. Libraries such as Botan and wolfSSL/wolfCrypt provide portable ML-DSA implementations oriented toward embedded and constrained environments, while managed-language ecosystems (Bouncy Castle for Java/.NET) and platform cryptography libraries (e.g., Microsoft SymCrypt) have introduced support for ML-DSA to ease enterprise and OS-level adoption [95, 255, 256, 312, 66, 227]. The Open Quantum Safe project and the PQ community repositories (e.g., the CRYSTALS-Dilithium reference code) continue to serve as primary sources for reference implementations, interoperability tests, and performance baselines [95, 255].

Side-channel resistance and secure implementation techniques Practical security of ML-DSA depends critically on careful implementation, particularly because Dilithium's signing algorithm uses sampling and rejection loops (the Fiat–Shamir-with-aborts paradigm) which can introduce side-channel risks if not handled correctly. Research has demonstrated that naïve implementations may leak through timing, power, or microarchitectural channels unless mitigations are applied. Consequently, production and reference implementations increasingly incorporate constant-time primitives for polynomial arithmetic, blinding of ephemeral randomness, and careful handling of rejection sampling to avoid secret-dependent timing or branching. Libraries such as OpenSSL, BoringSSL, and vetted reference implementations in the PQ community explicitly target hardened signing and verification routines informed by published side-channel analyses and implementation guidance [236, 95, 255]. High-performance, security-focused projects within the PQ Code Package and related efforts also integrate side-channel countermeasures (constant-time arithmetic, masked sampling, and hardened failure handling) into their Dilithium backends [264].

Formally verified and high-assurance implementations Given the complexity of lattice-based arithmetic and the long-term security goals of post-quantum signatures, there is active interest in formally verified or mechanically checked implementations of ML-DSA primitives. Efforts in the PQ community aim to produce high-assurance Dilithium components that provide guarantees of functional correctness and memory safety, and in some cases additional proofs or checks about timing-related behavior for critical kernels. Projects in the PQ Code Package and related repositories provide carefully audited reference code and, for selected arithmetic kernels, mechanically checked or formally verified artifacts that serve as high-assurance building blocks for production libraries and platform integrations [264, 95].

Hardware optimizations and acceleration As with other lattice-based schemes, ML-DSA benefits substantially from platform-specific optimizations. Optimized software backends exploit vector instruction sets such as AVX2 and AVX-512 on x86 processors and NEON on ARM architectures to accelerate NTT and polynomial arithmetic, which dominate the cost of key generation, signing, and verification; such optimized kernels are available in liboqs [255], the CRYSTALS-Dilithium reference implementation, and several production-grade libraries (e.g. [95]). Beyond software-level vectorization, ML-DSA has also been the subject of dedicated hardware implementations. Multiple FPGA prototypes of CRYSTALS-Dilithium have been reported in the literature, demonstrating efficient realizations of signing and verification while exploring trade-offs between throughput, latency, area, and energy consumption [277, 53, 217]. In addition, ASIC implementations targeting Dilithium have been proposed and evaluated, with particular emphasis on constant-time execution and side-channel-aware design suitable for embedded and high-assurance environments [204]. Taken together, these results show that ML-DSA can be efficiently implemented across a wide spectrum of platforms, ranging from constrained embedded devices to high-performance server-class systems.

4.4 Towards Robust PQC Implementation through QR Trusted Computing Architectures

The two core dimensions of the transition towards PQC are cryptanalysis maturity (incl. CMs design and implementation) and performance. In this regard, while SW-based implementations offer easy integration, they pose significant challenges in terms of efficiency. HW-based solutions (capturing also memory-mapped cryptography peripherals) can increase both performance and security (it is preferable not to have secret variables in commodity microcontroller processor registers as preventing leakage there is more difficult), at the cost of fragmentation due to dependence on specific HW. This highlights the benefit of SW/HW co-design, lowering the barrier of entry for secure and efficient PQ solutions with high interoperability. Co-designs can be categorized as tightly- and loosely-coupled [93], depending on whether they require HW modifications of the CPU. Several such approaches for lattice-based crypto are based on tightly-coupled HW modules [324, 301, 245, 220], or a combination of loosely- and tightly-coupled extensions [190], for the acceleration of NTT and SHAKE. Tightly-coupled HW acceleration has also been proposed for SLH-DSA [211, 306] and HQC [288, 102]. However, the consideration of CMs is lacking, as only masking has been considered for HQC [296], and masking, randomization, and custom PRF hardening for SLH-DSA [282]. Co-designs for isogeny- and multivariate-based crypto have not yet been explored.

SW/HW co-design is also the main design choice for establishing PQ Roots-of-Trust (RoTs). For instance, the OpenTitan RoT [212] has been leveraged for loosely-coupled instruction sets [300] and tightly-coupled Keccak acceleration extensions [4] in lattice-based crypto. However, due to high computational requirements, OpenTitan cannot yet be integrated into resource-constrained devices. The use of QR-TPMs

has been examined for ML-KEM, ML-DSA [130], and SLH-DSA [248], but the integration of CMs has not been considered. The use of TEEs has been investigated for mitigating side-channel vulnerabilities [205], but they lack implementation. All the above highlight the need to converge on an acceleration strategy allowing interoperability across multiple architectures, while supporting multiple PQC families. In addition, while the EU regulation on the adoption of the European Common Criteria-based cybersecurity certification (EUCC) is set to come in effect in 02/2025 [126], and the NIST CAVP [232] and CMVP [241] programs dictate the validation of vendor implementations of PQC schemes, it is still unclear which implementation attacks are the most critical and relevant for product security certification, and there are no well-established guidelines for running security tests.

4.5 Integration into High-End Security Protocols

Throughout this Chapter, we have provided a detailed overview on the state-of-the-art pertaining to various types Post-Quantum crypto primitives, focusing on both theoretical and implementation aspects. However, an important dimension that needs to be considered when examining the post-quantum transition of existing infrastructures is the **post-quantum elevation of higher-end security protocols, which use the aforementioned types of primitives as building blocks**. Specifically, many modern applications include components based on **standardized network protocols** and **security technologies** that need to be revised to support the use of PQC algorithms for the secure lifecycle management of all devices operating in a zero-trust manner. Thus, in order to enable the transition of such applications, there is a need not only for theoretical improvements on the underlying PQC primitives, but also *the crypto designs of the applications themselves, such as access control, data sharing, privacy-preserving authentication, and E2E communication*. Throughout this Section, we will provide a detailed overview of the state-of-the art pertaining to such higher-end security protocols.

Communication protocols for E2E security PQ-TLS has recently attracted considerable attention, and the IETF has provided a PQ implementation for TLS 1.3 [171]. KEMTLS [147] builds upon this by using ML-KEM for authenticated key exchange, and KEMTLS-PDK is a variant with pre-distributed keys [285]. However, all existing PQ solutions rely on replacing the classic DH used in TLS with a KEM-based solution, such as ML-KEM [293], code-based KEMs [310, 302, 111], and isogeny-based KEMs [310] (though using the now-broken SIKE). However, the use of KEMs may not be the optimal transition strategy for many applications relying on DH, thus highlighting the need for DH-alike and isogeny-based key exchange alternatives. There have also recently been efforts for enabling the IPsec protocol (for network application security) by quantum-hardening the underlying IKEv2 [136, 101] and integrating QKD/PQC solutions [7]. There has also been consideration for advanced PQC/QKD authenticated key protocols with increased key rotation capabilities, beneficial for telecom operators. However, while there is a convergence on the benefits of PQC/QKD in key management [172, 135], there is limited work on the validation and implementation of such schemes, and existing analysis is purely theoretical.

Progress on the migration of schemes like IKEv2 [260, 314] and IPSec [267] at the network layer, and OpenPGP [266] at the application layer, is currently slower, as their quantum transition is ongoing due to constraints posed by the size of the public keys. This leads to a core bottleneck regarding fragmentation, as the internal building blocks of these schemes, such as X.509 certificates, currently do not support the required sizes of signatures and keys [184]. Suppression of intermediate CA certificates for reduction of PQ certificate chains has been proposed for mitigating the fragmentation issue, but such solutions introduce security concerns [183].

One core issue towards the quantum transition of TLS 1.3 is the need for **PQ Authenticated Key Establishment (AKE)**, in order to simultaneously establish a shared session key between parties and verify

their identities in a quantum-resistant manner. Significant work has been performed by the IETF TLS Working Group [171], which has provided a draft on hybrid key exchange [173] combining classical ECDHE with PQ lattice-based KEMs, thus being one of the first to broadly adopt the current trend of hybrid KEMs. This highlights the benefits (but also the necessity) of a hybridization strategy, which has also been explored as part of the OpenSSL library v3.5 [298, 94, 303]. PQNoise [275] has been proposed as an AKE protocol that is more efficient than other PQ KEMs and dynamically supports multiple handshake patterns depending on the need of the communicating parties, but is hindered by the large public-key and ciphertext sizes of ML-KEM. It follows that these solutions pose a significant scalability hurdle, as they are subject to the performance bottlenecks of existing PQC (e.g., internal arithmetic operations of lattice KEMs). Also, the need for client-server interactions significantly increases the amount of data that needs to be exchanged as part of the handshake. Thus, there is a need for authenticated (non-interactive) key exchange that bypasses the need for server communication, thus minimizing the execution time and the amount of data exchanged during a handshake. A prominent solution is KEMTLS [147], which performs authentication through KEM key exchange, and demonstrates reduced handshake time and lower memory consumption than TLS 1.3. KEMTLS-PDK [285] further reduces handshake size and enhances scalability in server-based implementations, but requires pre-distributed public keys. A promising research direction entails the substitution of the traditional ECC-based Diffie-Hellman (DH) in TLS with lattice-based PQ key exchange protocols [63], but authentication has not been considered. The first proposed authenticated non-interactive PQ key exchange scheme is SWOOSH [134], which has performance limitations and requires large public keys (in order of 200kB), but it demonstrates the capability to move towards a more efficient authenticated pipeline. Specifically, it brings to the forefront the benefits of other crypto families, especially isogenies, which have smaller key sizes, features the same arithmetics as current ECC, and have potential for aggressive acceleration and optimization.

Identity Management and Execution of Data-Sharing Agreements Considering the increasing application of large-scale interconnected IoT, handling of private information at an individual-, business-, and government-level is a core concern. This is supported by Self-Sovereign Identity (SSI) infrastructures, leveraging the EU Digital Identity Wallet (EUDIW), which enable individuals to handle their own identities through Verifiable Credentials (VCs) with selective disclosure. Thus, there is a need for PQ privacy-preserving communication and identity management protocols, necessitating more advanced crypto abstractions to support and expand existing SSI infrastructures. The use of lattice-based schemes for achieving VCs suitable for selective disclosure, non-interactive renewal, and revocation has been explored [118], but the use of QR signatures in SSI has not yet been explored. An open issue pertains to the use of QR signatures for the delegation of credential use to third-party agents through the OAuth protocol [313].

To fulfil the strict privacy, integrity, and correctness requirements of SSI [127], the use of anonymous credentials was proposed [48]. The emergence of zkSNARKs enabled their construction for any signature scheme, and their application with (not PQ) ECDSA is currently considered. This approach is crypto-agile and can facilitate PQ transition, as it allows the convenient shift from ECDSA to a PQ signature scheme with a suitable PQ zkSNARK. ML-DSA with zkSNARKs has been investigated, but existing solutions are limited in scope [262] and lack rigorous security analysis [281]. Also, the use of anonymous credentials with attestation in a quantum-secure manner has not yet been investigated. Zero-knowledge proofs are also a core enabler for privacy-preserving authentication schemes, such as DAA and EPID, but existing solutions [78, 79] have low efficiency and lack security evaluation.

Research around QR management and execution of data sharing agreements mostly revolves around access control, such as ABE [74, 9, 315, 110] and ABS [121, 320, 213]. However, these generally do not provide forward secrecy, which is needed in E2E data sharing among multiple users. Updatable encryption (UPKE) offers regular updates of public-private key pairs, but research on PQ schemes is

immature [108], as they do not offer implementation or security analysis for optimal parameter selection.

4.5.1 Homomorphic Encryption

The versatile nature of Homomorphic Encryption (HE) – combined with the wide range of its applications – has rendered it one of the most significant fields of research in recent years. In 1978, one year after the introduction of RSA [280], Rivest et al. suggested HE serves as a solution against the incompatibility between user privacy and the need for data delegation and computational requirements [279]. HE performs computations on encrypted data and produces an encrypted result that can be decrypted to the same result as if the computations had been performed on unencrypted data.

Several existing public-key cryptosystems, including RSA [280], ElGamal [122], and Goldwasser-Micali [146], present homomorphic properties, though none support the computation of arbitrary functions on ciphertexts. The goal of obtaining a fully homomorphic encryption (FHE) scheme continued until Gentry's breakthrough in 2009 [137]. This scheme dealt with the difficulty of specific mathematical problems concerning ideal lattices and enabled *unrestricted* computations on encrypted data. Through the unrestricted computation feature, FHE has been brought as the leading solution to many applications, such as private information retrieval [32], private set intersection [76, 75, 89], and privacy-preserving machine learning [159, 201].

Over the years, researchers have developed three types of HE: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption (FHE), each with distinct properties and capabilities. FHE, though computationally intensive and resource-demanding, enables arbitrary computations on ciphertexts, including addition, multiplication, and other complex operations. In contrast, PHE supports only a single operation, either addition or multiplication, making it computationally less demanding but significantly less flexible than FHE. SHE offers an intermediate solution, allowing a limited number of additions and multiplications while being more practical than PHE and less resource-intensive than FHE.

Since the groundbreaking work by Gentry [137], numerous HE schemes have emerged to handle diverse data types. Notably, recommended by Homomorphic Encryption Standard [17], the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [68] supports modular arithmetic over finite fields, enabling computations on vectors of integers in $\mathbb{Z}_q$ with $q \geq 2$. Similarly, the Brakerski/Fan-Vercauteren (B/FV) scheme [67, 128] operates over finite fields and facilitates computations on integer vectors. The Cheon-Kim-Kim-Song (CKKS) scheme [80] allows approximate computations on vectors of real and complex numbers. Additionally, the Torus FHE (TFHE) scheme [81, 82, 83] employs boolean circuits and decision diagrams for low-precision integers [84]. Each HE scheme is meticulously designed to achieve specific optimization goals. BGV, B/FV, and CKKS prioritize minimizing the multiplicative depth in their decryption circuits, while TFHE focuses on reducing the gate count required for decryption operations. Section 6.3 presents a detailed explanation of the mentioned HE schemes, along with their correctness.

4.5.2 Functional Encryption

Functional Encryption (FE) and Multi-Client Functional Encryption (MCFE) FE allows a user to compute a specific function on encrypted data without revealing the underlying plaintext, and to obtain only the result of that function. For example, Inner Product Functional Encryption (IPFE) [259, 10, 129] and quadratic polynomials [42] being some of the most studied functions. The number of supported functions continues to grow; we can add the recent discovery of *partial decryption* [250] in 2024.

In 2014, a new functionality for FE was introduced: *Multi-Client Functional Encryption* by Goldwasser [145], which has since become a main feature of FE schemes. In an MCFE scheme, a ciphertext depends on a

specific label, thereby limiting decryption to users who possess a decryption key for that label. While this feature is perfect for a system requiring fine-grained access control, it narrows flexibility in data management. Indeed, over the past dozen years, countless constructions have been proposed [207, 206, 11], covering a large range of functionalities and system models. Another notable recent technique, *Ciphertext Updatability* [85], allows an authority to issue update tokens that enable updating the label under which a message was encrypted.

Ciphertext Updatability for tags The idea of rightfully tampering with a ciphertext has been introduced by Boneh *et al.* [60] in 2013. Named *Updatable Encryption* (UE), this technique modifies a ciphertext to change, or update, the key with which it is encrypted. Whilst simple, this feature has sparked interest in the field [225, 105, 294, 224]. Its adaptation to MCFE has emerged with the article of Cini *et al.* [85], who defined the notion of Ciphertext Updatable Functional Encryption (CUFE). While classical UE updates the entire key, CUFE updates the label under which the ciphertext was encrypted, without re-encrypting it from scratch. To sum up, the owner of the model can decide which ciphertext's lifetime can be extended for further use. Because this paradigm is very recent, little work has been conducted on proposing CUFE schemes. However, we believe this technique will gain greater attention in the future and will soon become an expected feature in most MCFE schemes.

Example of FE use: Partial Decryption FE schemes allow the computation of specific functions on ciphertexts. However, FE schemes that allow fine-grained control and selective computations over encrypted data, while revealing only a part of the plaintext, are difficult to come by.

To the best of our knowledge, the only FE scheme allowing this so-called partial decryption is SPADE [250]. Other techniques can enable the decryption of parts of an encrypted database by identifying specific patterns in a collection of ciphertexts [162], and another technique uses permutations [39].

SPADE [250] is an FE scheme introduced by Nuoskala *et al.*. In the article, the authors achieve fine-grained control over ciphertext data during decryption. It is considered state-of-the-art in partial decryption by FE. It relies on the cyclic group and Additive ElGamal and can be run on quantitative and qualitative data. It served as the main inspiration for AoS. Despite its novelty, in our opinion, SPADE is lacking additional components such as ciphertext updates as in [85] as well as post-quantum security.

4.6 Lattice Trapdoors and Gadgets and Applications

In lattice-based cryptography, trapdoors play a central role in enabling advanced cryptographic functionalities while preserving security based on hard average-case problems. Informally, a trapdoor is additional secret information associated with a public lattice object—typically a matrix defining a lattice—that allows one to efficiently solve certain computational tasks that are otherwise believed to be hard. A canonical example is Gaussian preimage sampling: given a public matrix A and a target vector u, finding a short vector x such that $Ax = u \bmod q$ is computationally intractable in general (namely, this would solve SIS), but becomes feasible when a suitable trapdoor for A is known. Trapdoors are used in many lattice-based constructions, including, for example, signatures [138, 131] and identity-based encryption [209, 178].

The Ajtai–Dwork [14] cryptosystem introduces the notion of a lattice trapdoor as a hidden geometric structure: the public key defines a lattice that appears random and for which decoding noisy lattice points is computationally hard. Here, the secret key consists of a short basis (the trapdoor) that enables efficient bounded-distance decoding. Gentry, Peikert, and Vaikuntanathan [138] generalize lattice trapdoors by designing structures that support efficient Gaussian sampling and trapdoor delegation, thereby transforming lattice trapdoors into a versatile primitive that is used to construct a signature scheme and an

identity-based encryption scheme. Alwen and Peikert [28] significantly improve this paradigm by providing an explicit and efficient trapdoor generation method for random-looking matrices, ensuring the existence of a trapdoor by construction rather than by probabilistic arguments. Finally, Micciancio and Peikert [226] introduce simpler and more efficient trapdoors with tighter security reductions and improved parameters, establishing an efficient framework that has become the standard foundation for modern schemes relying on trapdoors.

4.6.1 Gadgets

Early lattice trapdoors, such as those used in the GPV framework [138], rely on the knowledge of a short basis of a lattice. While theoretically powerful, these trapdoors suffer from large key sizes and limited efficiency, thus ruling out practical instantiations. To address these limitations, an alternative notion known as gadget trapdoors was introduced in [226]. Rather than relying on a hidden short basis of an arbitrary lattice, gadget trapdoors exploit the structure of a specially generated, publicly known gadget matrix. This matrix defines a simple lattice for which inversion and Gaussian preimage sampling can be performed efficiently. Random looking lattices, which should normally be used in cryptographic schemes, are then generated from the gadget lattice via linear transformations, effectively transferring the trapdoor functionality. Thanks to their efficiency, gadgets have become a fundamental tool for bridging the gap between theoretical lattice constructions and practical post-quantum cryptographic schemes, especially regarding advanced constructions whose realizations would otherwise be impractical.

A gadget matrix is a public, highly structured matrix $G \in \mathbb{Z}_q^{n \times n\ell}$ designed to support efficient decomposition of vectors modulo q. A canonical example is

$$G = I_n \otimes \mathbf{g} = \begin{bmatrix} \mathbf{g} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{g} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{g} \end{bmatrix}$$

where $\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1})$, $\ell = \lceil \log_2 q \rceil$ and $\mathbf{0}$ is the vector with all zero components of size ℓ.

The key property of G is the existence of an efficient algorithm that, given any $y \in \mathbb{Z}_q^n$, computes a short vector x such that $Gx = y \bmod q$. Instead of embedding a short basis directly into a public matrix, modern constructions express the public matrix as

$$A = [A' \mid A'T + G],$$

where A' is uniformly random, T is the trapdoor, and G is a gadget matrix. This approach ensures that the public matrix A is statistically close to uniform and the trapdoor T enables reduction of hard problems on A to easy problems on G.

An important advantage of gadget-based trapdoors is the ability to delegate trapdoors. Given a trapdoor for a matrix A, it is possible to efficiently derive a trapdoor for related matrices $A' = AR$, where R is a public transformation.

Trapdoor delegation is a key ingredient in hierarchical and identity-based encryption schemes.

4.7 PiQASO considered threat model

Based on the analysis of the relevant attack categories described in this chapter, the threat model envisioned in PiQASO will follow a two-layer approach and will aim at:

1. analyzing **attack vectors targeting low-level PQC primitives**, and
2. examining different **attack scenarios when these primitives are integrated into high-end protocols**.

For low-level cryptographic primitives, the threat model considered in PiQASO targets the implementations of the NIST-standardized ML-KEM, ML-DSA and SLH-DSA schemes. In particular, the threat model will consider **adversaries capable of launching the most prominent SCAs in HW-based implementations**, by exploiting access to side-channel leakage information, such as power consumption (DPA and SPA attacks). As a mitigation strategy for such types of SCAs, **PiQASO will aim at developing a fully masked implementation of first and second order for ML-KEM, ML-DSA and SLH-DSA**. In addition, **timing-based attacks** (on NTT for lattice schemes and on Keccak in SLH-DSA) **will also be considered in the threat model, targeting SW-based implementations**, and will be mitigated by including **constant-time implementations** of secret-dependent subroutines, as well as **masking and shuffling countermeasures**. Furthermore, the proposed implementations of the aforementioned **PQC primitives will be tested against known SW- and arithmetic-level bugs and vulnerabilities**. In particular, PiQASO will first investigate existing NIST Known Answer Tests (KATs) and Test Vectors, and will next aim to identify bugs and validate SW and HW implementations of ML-KEM, ML-DSH, and SLH-DSA. PiQASO will also leverage NIST-established parameter sets targeting specified security levels in order to ensure the correct instantiation of PQC primitives, and will verify their correctness against fundamental security properties, namely IND-CPA for PKE, IND-CCA for KEMs (based on the hardness of the standard MLWE) and SUF-CMA for digital signature schemes (based on the hardness of the standard MLWE and MSIS lattice problems).

At a higher layer, PiQASO envisions the integration of the implemented PQC primitives into legacy, high-end cryptographic protocols, such as TLS and IKEv2, which fundamentally rely on the development of a (hybrid) Authenticated Key Establishment (AKE) mechanism. In addition, PiQASO aims at developing new, lattice-based advanced cryptographic protocols, including Attribute-Based Encryption (ABE) and Updatable Public Key Encryption (UPKE), to support the PQ-as-a-Service (PQaaS) concept. **Hence, at a protocol layer, the considered threat model assumes adversaries targeting known weaknesses of communication protocols, but also vulnerabilities emerging from embedding low-level PQC primitives into complex protocol stacks**. In this case, threats include certificate validation flaws in PKI, CA private key leakage, downgrade and MITM attacks, as well as various attacks targeting the handshake layer in legacy protocols. Consequently, the **security requirements posed by protocol deployments will be formally verified** against an extended set of application-driven security and privacy properties, using well-established formal verification frameworks, selected based on the underlying threat model and security objectives. This may involve verification in the symbolic (Dolev-Yao) model, using automated tools such as Tamarin or ProVerif, the design of computational proofs following the provable-security framework, or simulation-based approaches, by deploying the Universal Composability (UC) framework. Furthermore, the migration to PQC, specifically **the integration of PQC primitives into high-end protocols, introduces additional risks, stemming from significantly larger public keys, ciphertexts and signatures** compared to classical cryptosystems. These increased values will result in exceeding the packet size limits in legacy communication protocols (e.g., TLS and IKEv2) and will lead to packet fragmentation at different layers of the network. Fragmentation can have a negative impact, not only in the overall performance, but also in the reliability and security of the underlying communication protocol. This is because it often leads to increasing latency and potentially the expected retransmission rates, exposing the protocol to Denial-of-Service (DoS) and traffic analysis attacks. **PiQASO realises these additional challenges in the PQC migration process and envisions a carefully-engineered transition to PQC**, with specific attention to protocol constraints, transmission requirements, and implementation-level mitigation strategies to ensure a robust and secure deployment of legacy, high-end cryptographic protocols. In addition, the construction of advanced PQC protocols (ABE and UPKE) often mandates the use of

new, non-established parameter sets to instantiate the lattice functions. These **new instantiations will be carefully selected to ensure that reductions to well-established hardness assumptions, such as MLWE and MSIS, provide concrete security** against both classical and quantum-enabled adversaries, while at the same time, the usual indistinguishability (IND-CCA/IND-CPA for KEMs and PKE) and unforgeabilty notions (for digital signatures) are not violated.

Chapter 5

Forward Security in PiQASO

Cloud storage services have become a fundamental component of modern computing infrastructures, enabling users to outsource data storage and management to remote servers. While this paradigm offers scalability, availability, and cost efficiency, it also introduces significant security challenges, particularly in adversarial environments where long-term data confidentiality cannot be assumed. In such settings, the compromise of cryptographic keys represents a critical threat, as it may allow an attacker to access not only current data but also previously stored information.

Forward security addresses this risk by ensuring that the exposure of a secret key at a given point in time does not compromise the confidentiality of data protected in the past. By periodically evolving cryptographic keys and securely erasing obsolete key material, a forward-secure cloud storage scheme limits the impact of key compromise to a narrow time window. This property is especially important in cloud environments, where servers are continuously exposed to attacks and where breaches may remain undetected for extended periods.

Incorporating forward security into cloud storage schemes provides strong resilience against key exposure and insider threats, thereby enhancing long-term data protection. As cloud-stored data often remains sensitive well beyond the lifetime of any single cryptographic key, forward security emerges as a crucial requirement for building trustworthy and robust cloud storage systems.

In this chapter, we discuss how to use a suitable cryptographic approach to guarantee forward security in the encryption provided by PiQASO: updatable public key encryption. After an introduction and a discussion revolving around the state of the art on this primitive, we show how to realize a trivial instance of it, which is enough to guarantee forward security in the case of a trusted server managing keys and updates. We make this formally by giving a security model for the cloud that captures forward security and proving that the cloud enriched with the update procedure satisfies forward security.

5.1 Updatable Public Key Encryption

With UPKE, we consider a broad class of public-key encryption (PKE) schemes supporting evolution of private/public key pairs, possibly along with associated ciphertexts. Its asymmetric nature implies that schemes within this class are suitable for applications requiring interaction between third parties (typically performing encryption) and the owner of a private/public key pair (typically, performing decryption). These schemes were introduced in the context of Forward Security for public-key-encrypted data [71] and more recently gained popularity in the context of secure messaging [182, 24].

The first construction of a public key encryption scheme with the goal of forward security can be found in Canetti et al. [71] under the name of forward-secure public-key encryption (FS-PKE). Since then, it has

Co-funded by the European Union
ECCC

become a central primitive for bringing forward security into the public-key setting, where confidentiality of past communications must be preserved even if long-term keys are compromised. Recently, this concept has found applications in efficient forward-secure key-exchange protocols [156, 99, 100]. The true strength of FS-PKE lies in its ability to provide a non-interactive approach: starting from an initial key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$, the scheme defines two synchronised key chains:

$$\mathsf{pk}_0 \to \mathsf{pk}_1 \to \cdots \quad \text{and} \quad \mathsf{sk}_0 \to \mathsf{sk}_1 \to \cdots.$$

Public keys can be updated in a publicly computable way, while only the receiver can evolve the secret key. Note that, in the literature, two equivalent viewpoints are often considered: public keys evolve in a deterministic chain or they remain constant. This stems from the fact that the encryption algorithm uses both a constant public key and the index associated with the current epoch. Crucially, the evolution of secret keys is one-way: compromising sk_i allows decryption only of ciphertexts encrypted under the corresponding public key pk_i and any pk_k for $k > i$, but it remains infeasible to recover any prior sk_j for $j < i$, thereby protecting past communications.

This property can be achieved with a simplified version of Hierarchical Identity-Based Encryption (HIBE) [139, 57], and indeed the first FS-UPKE construction was built from HIBE [71]. As more efficient and diverse HIBE schemes were developed—based on bilinear pairings [57], lattice assumptions [72], or even pairing-free groups [112]—corresponding FS-UPKE constructions followed. Some theoretical work has also shown instantiations from weaker assumptions, such as DDH [112], or low-noise LPN [113]. Despite this broad landscape of possible instantiations, all known FS-UPKE schemes inherit the drawbacks of their HIBE roots: they are more complicated to implement and significantly less efficient than standard PKE.

Another important limitation is that FS-UPKE only provides forward security, not post-compromise security (PCS) [87], where PCS means that a system can heal after a key exposure. Consequently, once a secret key is exposed, all future secret keys in the chain can be derived by the adversary, meaning that the system cannot recover security going forward.

Taken together, these limitations have so far prevented FS-UPKE from being deployed in real-world messaging or key-exchange protocols [97, 149]. Nonetheless, the primitive remains an important conceptual bridge: it shows how forward security can be formalised and achieved in non-interactive public-key settings, and it continues to inspire new approaches.

Notice that, after the introduction of FS-PKE, other primitives supporting PCS based on it were introduced [109, 106, 107]. However, these primitives are based on HIBE, therefore suffering from the same efficiency problems as FS-PKE.

To address the aforementioned performance limitations, Jost et al. [182] introduced a new primitive under the name of updatable public key encryption (UPKE). These schemes gained popularity due to their efficiency, comparable to standard PKE/KEM schemes (c.f. Table 5.5 for the comparison). These are mainly used in continuous group key agreement (CGKA) [27, 25], which serves as a foundation of secure messaging protocols [24], allowing a large group of users to maintain a shared secret key that is frequently rotated by group members to achieve FS and PCS [25, 36].

In UPKE, any user can decide to update the public key of any other user and provide a token that is used by the owner of the secret key to update it. This requires coordination between parties: since the public key is updated, all the parties need to be up to date regarding the current public key. Indeed, to guarantee FS, the previous secret key must be deleted after the update. This implies, if the scheme is secure, that the user is not able to decrypt messages encrypted with an old public key.

Updating the public key usually results in a randomisation of the secret key. This is the main difference with FS-UPKE, where public and private keys evolve in a deterministic chain, as explained before.

Although UPKE requires coordination, it usually provides PCS: if an adversary compromises a secret key

at epoch e, then after an update performed by an honest party (e.g., the key owner), security is restored, as the new secret key, which is the result of a randomization of the old one, becomes unknown to the attacker. In this sense, PCS can be viewed as the analogue of FS for future instead of past epochs. Some works in the literature (e.g., [22]) jointly refer to these properties as post-compromise forward security (PCFS).

5.1.1 Syntax and Semantics of UPKE

Although the UPKE notion provides a general understanding of the core functionalities and security models involved, different works in the literature diverge in significant syntactic and conceptual details (as was earlier observed in [26]). This is often because the proposed syntax in each work may be tailored to specific application scenarios.

Note that, as in the context of conventional public-key encryption, it is useful to distinguish between UKEM—the analogue of a KEM in the updatable setting—and UPKE schemes. Earlier works focused primarily on UPKE constructions [182, 119, 108]. More recent research [26, 19], however, has shifted attention to UKEMs, a formulation that better aligns with modern cryptographic protocols and composes more naturally with the established KEM/DEM paradigm. In addition, UKEMs provide improved compatibility with emerging standards such as the NIST post-quantum KEM specifications, i.e., ML-KEM [237], making them a natural candidate for practical deployment. Throughout the section, unless stated otherwise, we use UPKE as the default notion. We note that the transformation between the two notions is straightforward: an UKEM can be obtained directly from an UPKE, and, as in the standard setting (i.e., between PKE and KEM), hybrid-encryption techniques, i.e., KEM/DEM, also allow constructing a UPKE from a UKEM.

5.1.1.1 Defining UPKE and Its Correctness

For convenience, we adopt the syntax of [6], which is the same used also in other works, like [108, 19].

Definition 4 (UPKE)**.** *An Updatable PKE scheme consists of a PKE scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *and two associated update algorithms (*$\mathsf{UpdatePk}$,$\mathsf{UpdateSk}$*):*

- $\mathsf{UpdatePk}(\mathsf{pk}, up)$*: the update public key algorithm takes as input a public key* pk *and a public key update value* up *and outputs a new public key* pk'*, along with a secret key update token* Δ*;*
- $\mathsf{UpdateSk}(\mathsf{sk}, \Delta)$*: the update secret key algorithm takes as input a secret key* sk *and a secret key update token* Δ *and outputs a new secret key* sk'*.*

Correctness property. We require the following correctness property: We call a key-pair $(\mathsf{sk}', \mathsf{pk}')$ fair if it has been generated by the KeyGen algorithm or if it has been obtained by updating a fair key-pair, i.e., $(\mathsf{sk}', \mathsf{pk}')$ is fair if $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KeyGen}(1^\lambda)$ or $(\mathsf{pk}', \Delta) \leftarrow \mathsf{UpdatePk}(\mathsf{pk}, \delta)$, $\mathsf{sk}' \leftarrow \mathsf{UpdateSk}(\mathsf{sk}, \Delta)$, for $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Then, a UPKE is correct if, for every fair key-pair $(\mathsf{sk}', \mathsf{pk}')$, any possible update value up and any possible message m:

$$\mathbb{P}[\mathsf{Dec}(\mathsf{sk}', \mathsf{Enc}(\mathsf{pk}', m)) = m] = 1,$$

where this definition can be trivially relaxed to the case of δ-correctness, i.e., decryption is allowed to fail with probability δ.

This syntax is called “long symmetric syntax”. Other types are the “short syntax” and “long asymmetric syntax”. In the short syntax, there are no update algorithms: the public key is updated with a random

Co-funded by the European Union ECCC

update value up inside the encryption algorithm. The private key is updated inside the decryption algorithm, where this takes as input also the current public key: the update token Δ is retrieved from the new and old public keys. In the long asymmetric syntax, other than the update algorithms, there is an update generation algorithm that produces a string to update a public key and a string to update the related secret key.

Note that the syntax of UKEM and its correctness are symmetric to the one of UPKE, and therefore we don't show it explicitely.

Before discussing security, we introduce a property that will be used in the comparison of schemes later. Some UPKE constructions allow for agnostic updates, meaning that the update token can be generated independently of the specific public key to be updated (e.g., [182, 119]). This offers a more modular approach, potentially improving scalability and usability in distributed environments. Other schemes (e.g., [108, 35, 5]), instead, require update tokens to be tightly bound to the public key being updated, which may simplify security proofs but reduce flexibility.

5.1.2 Summary of Security Notions

Security for UPKE schemes extends classical PKE notions to account for key evolution over time. While different syntaxes can be adapted to each other, the security models differ significantly in how they treat updates, adversarial control over randomness, key exposure, and structural properties such as branching key histories.

Core Goals: FS and PCS. Most models aim to capture FS: once a trusted update has occurred, exposure of the current secret key must not compromise ciphertexts generated in past epochs. Stronger notions additionally capture PCS, requiring that confidentiality can be restored after a corruption, provided that honest updates follow.

One-Wayness Notions. The weakest notions are variants of one-wayness (OW-CR-CPA and OW-CR-PCA), which extend classical OW-CPA/PCA security to the updatable setting. The adversary may trigger key updates (possibly with chosen randomness (CR)), receive a challenge ciphertext encrypting a random message, and eventually learns the updated secret key after a trusted update. Security requires that the adversary cannot recover the plaintext. OW-CR-PCA implies OW-CR-CPA as in the standard PKE setting. Although these notions are insufficient in isolation, generic transformations can lift them to stronger indistinguishability guarantees.

IND-CPA Variants. Among CPA-based security notions, there is a spectrum of strength.

At the basic level, some models capture forward secrecy (FS) under the assumption that updates are honestly generated and trusted. The adversary can observe ciphertexts, but it can't influence how updates are performed or how randomness is chosen.

Stronger variants increase adversarial power. In some cases, the attacker can control the randomness used during encryption, making indistinguishability harder to achieve. In others, the adversary is given active influence over the update process itself — for example, by triggering updates, interacting with update mechanisms, or even choosing maliciously generated update outputs. These models aim to capture FS in a more realistic, adversarially controlled, and dynamic environment.

Beyond that, there are notions that also allow secret-key corruption. These are designed to model post-compromise security (PCS) and multi-instance settings, where keys may be exposed, and the system

is expected to recover security after subsequent updates. However, these corruption-based models are structurally different from the update-centric ones, and the differences in how updates and exposures are handled can lead to separations between the corresponding security guarantees.

IND-CCA Variants and External Updates. When moving to CCA security, the models additionally give the adversary access to a decryption oracle, with the usual restriction that it cannot decrypt the challenge ciphertext in its own epoch. Some variants also take into account external updates, where update tokens are publicly visible. In these settings, the adversary can observe updated information (as would happen if an external service distributes updates), but only the secret key owner can derive the corresponding new secret key. This reflects real-world applications where updates are not purely internal or hidden.

Forking and Tree-Based Security. All of the above models typically assume that keys evolve linearly over time — one update after another in a single chain. Stronger formulations generalize this to a tree-like evolution, capturing forking security. This becomes relevant in asynchronous or group messaging scenarios, where different branches of key updates can emerge from a common key-pair. In such settings, security must hold even if the adversary causes the key history to split and later corrupts selected branches. Schemes that are secure in the simple linear setting may fail in the tree-based one, showing that security under forking is strictly stronger.

In table 5.1 we compare all the security notions given for UPKE schemes in the literature; we refer to [219] for a more exhaustive explanation and a formal presentation of these security notions.

5.1.3 Comparison of Existing Schemes

In this section, we provide detailed comparison tables for all known UPKE/UKEM constructions. We include concrete instances—grouped according to the underlying mathematical frameworks (i.e., groups and residuosity-based, lattices, and isogenies)—as well as general constructions.

In Table 5.2 we compare the complexity of all constructions. Specifically, we report the sizes of secret key, public key, ciphertext, update token, and plaintext, and give the computational costs of encryption/encapsulation and public-key update. We focus on the cost of encryption/encapsulation because, in all considered schemes, decryption is no more expensive than encryption; similarly, the public-key update is at least as costly as the secret-key update, so we report only the former. Here, the notation $t \cdot S$ denotes the size of t elements in the set S. For group-based schemes, each group element has size 2κ, where κ is the security parameter. For lattice-based schemes, m, n and n' denote matrix dimensions over the base ring $\mathbb{Z}_p$ (or $\mathbb{Z}_{p'}$). In the short-syntax setting (see Section 5.1.1), the encryption cost already includes the public-key update procedure, and update sizes are omitted since no explicit update string is produced. Finally, the last three groups correspond to general constructions with CCA security, starting from an appropriate base scheme, that is, a PKE, an IND-CR-CPA, or an OW-CR-CPA UPKE (depending on the group). For these groups, only the dominant extra costs (overheads), such as the proof of knowledge, on top of the corresponding base schemes, are given. Regarding [108]'s CPA-to-CCA transformation, we report the sizes from the original paper even though the proof has been shown to be flawed, as the corrected version [35] maintains essentially the same asymptotic sizes and complexities.

In Table 5.3 we compare the security guarantees provided by existing UPKE/UKEM schemes along with some other properties. First, we indicate whether a scheme is a UPKE or a UKEM. For security, we list the model in which the security of the scheme is proved and the underlying assumptions. The other properties are as follows: **A** stands for agnostic updates, ∞ **Upd** means that the scheme supports an unbounded number of updates, **PUV** denotes public update validation and the last column, **Syntax**, follows what we discussed in Section 5.1.1, where L.A. denotes long asymmetric syntax, L.S. long symmetric one,

Property	Reference	CR	CU	Corrupt	MI	External Updates	Forks?	Notes
IND-CPA	[23]	✘	✘	✘	✘	✘	✘	Adversary chooses messages; cannot control update randomness; challenge after trusted update; FS.
IND-CPA*	[23]	✓	✘	✘	✘	✘	✘	As IND-CPA, but adversary controls encryption/update randomness.
IND-CR-CPA	[108, 5, 6, 35, 19]	✓	✘	✘	✘	✓	✘	As CR-CPA but the adversary chooses update values; updates allowed after challenge; requires trusted update before challenge answer.
IND-CU-CPA	[108, 5, 6, 35, 19]	✓	✓	✘	✘	✓	✘	Adversary can directly choose update strings (stronger than CR).
C-IND-CPA	[119, 182, 117]	✓	✘	✓	✘	✘	✘	Adversary can choose update values and corrupt keys; challenge must follow a trusted update and precede adversary-controlled updates + compromise.
MI-C-IND-CPA	[182]	✓	✘	✓	✓	✘	✘	Multiple key pairs from start; same update randomness can be applied to several keys; updates/exposures affect all keys in the same epoch.
IND-CR-CCA	[108, 5, 6, 35, 19]	✓	✘	✘	✘	✓	✘	CR-CPA variant with decryption oracle; cannot decrypt challenge ciphertext in challenge epoch.
IND-CU-CCA	[108, 5, 6, 35, 19]	✓	✓	✘	✘	✓	✘	CU-CPA variant with decryption oracle; cannot decrypt challenge ciphertext in challenge epoch.
T-IND-CCA	[26, 29]	✘	✘	✓	✘	✓	✓	Keys evolve in a tree (forking security); reveal oracle allowed; challenge restrictions prevent trivial wins; tailored for group messaging.

Table 5.1: Extended comparison of UPKE security properties.

and S is short syntax. For general constructions, the **Assumptions** column lists the basic primitives or schemes from which the final construction is derived. Finally, in the table, the notation ✘/✓denotes that the corresponding property is inherited from the chosen instantiation of the underlying base scheme, and therefore may or may not be satisfied.

In Table 5.4, we additionally emphasize the differences between the parameters used in [19] and in other lattice-based schemes. This comparison highlights that, although the asymptotic complexities suggest that [19] is the most efficient scheme, its concrete parameters are substantially larger in practice. This increase in parameter sizes, along with the need for additional assumptions, is the cost of supporting an unbounded number of updates. For performance reasons, [29] and [6] adopt the ring-LWE variant of the scheme. In contrast, [19] cannot do so due to the nature of its construction. As a result, the parameters used in these works and provided in Table 5.4 differ, as explained in the following. The parameter p is related to the size of the message space: in [6, 29] the message space consists of d elements, each in $\mathbb{Z}_p$, whereas in [19] it contains p elements. The parameter q represents the size of the base ring, that is $\mathcal{R}_q$ in [6, 29] and $\mathbb{Z}_q$ in [19]. The value n denotes the vector length: in [19] it refers to the lattice/code vector length, while in [6, 29] it is the message dimension when encrypting a vector. The parameter k gives the dimension of the code in [19], and d is the degree of the polynomial defining the base ring in [6, 29]. The quantity B bounds the error in [6, 29], and h denotes the hull dimension of the code in [19]. Finally, ℓ specifies the number of supported updates. The final column clarifies that although the ciphertext size in [6, 29] appears larger, this is due to their much larger message space; consequently, the ciphertext-to-

plaintext bit ratio is significantly smaller. Moreover, the ciphertext in [29] includes a member tag, making it secure in a model that supports forks, i.e., T-IND-CCA security, where [19] is IND-CR-CPA secure.

Finally, in Table 5.5 we provide a comparison of concrete sizes focusing on the most efficient UKEMs, and compare them with standard KEMs. Notice that, from a size perspective, the most efficient UKEMs in the pre-quantum setting are close to KEMs in current use. However, in the post-quantum setting where sizes are generally larger, a 5 times expansion for ciphertexts is rather significant.

	Sizes					Computational Costs	
Construction	**Sk**	**Pk**	**Ct**	**Upd**	**Pt**	**Enc**	**Upd PK**
Constructions based on Pre-Quantum Assumptions							
JMM18 [182]	2κ	2κ	2κ	–	2κ *	$\mathcal{O}(\kappa)$	$\mathcal{O}(\kappa)$
DKW22 [108] (Sec. 5.3)	$20\kappa^2$	$20\kappa^2+\kappa$	$20\kappa^2+\kappa$	$200\kappa^3+10\kappa^2$	2κ	$\mathcal{O}(\kappa^2)$	$\mathcal{O}(\kappa^3)$
HLP22 [5] (Sec. 3.2)	$2\kappa^2-2$	$2\kappa(\zeta+1)$	$2\kappa(\zeta+1)$	$2\kappa(\zeta+1)$	*	$\mathcal{O}(\zeta\kappa)$*	$\mathcal{O}(\zeta\kappa)$*
AFM24 [26]	2κ	4κ	2κ	–	2κ	$\mathcal{O}(\kappa)+\pi$	–
Constructions based on Lattices							
DKW22 [108] (Sec. 6.3)	$\mathcal{O}(m)\cdot\mathbb{Z}_2$	$\mathcal{O}(mn)\cdot\mathbb{Z}_p$	$\mathcal{O}(m)\cdot\mathbb{Z}_p$	$\mathcal{O}(m^2)\cdot\mathbb{Z}_p$	$\{0,1\}$	$\mathcal{O}(mn)$	$\mathcal{O}(m^2n)$
HPS23 [6] (Sec. 4)	$\mathcal{O}(n)\cdot\mathbb{Z}_p$	$\mathcal{O}(n^2)\cdot\mathbb{Z}_p$	$\mathcal{O}(n^2)\cdot\mathbb{Z}_p$	$\mathcal{O}(n^2)\cdot\mathbb{Z}_p$	$\mathcal{O}(n)\cdot\mathbb{Z}_p$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
AFMR25 [29]	$\mathcal{O}(n)\cdot\mathbb{Z}_p$	$\mathcal{O}(n)\cdot\mathbb{Z}_p$	$\mathcal{O}(n^2)\cdot\mathbb{Z}_p$	–	$\mathcal{O}(n)\cdot\mathbb{Z}_p$	$\mathcal{O}(n^3)+\pi$	–
ABL25 [19]	$\mathcal{O}(n')\cdot\mathbb{Z}_2$	$\mathcal{O}(n'k')\cdot\mathbb{Z}_{p'}$	$\mathcal{O}(n')\cdot\mathbb{Z}_{p'}$	$\mathcal{O}(n')\cdot\mathbb{Z}_{p'}$	$\mathcal{O}(n')\cdot\mathbb{Z}_{p'}$	$\mathcal{O}((n')^2)$	$\mathcal{O}((n')^3)$
Constructions based on Isogenies							
EJKM20* [119] (Sec. 6)	$1\cdot[-\mu,\mu]^{\hat{n}}$	$1\cdot\mathbb{F}_{\hat{p}}$	$1\cdot\mathbb{F}_{\hat{p}}$	–	$1\cdot\mathbb{F}^*_{\hat{p}}$	2 GA	1 GA
DFV24 [117]	$10\cdot\mathbb{F}_{\hat{p}'}$	$2\cdot\mathbb{F}_{\hat{p}'}$	$2\cdot(\mathbb{F}_{\hat{p}'})+E(\overline{\mathbb{F}}_{\hat{p}'})$	–	$\{0,1\}^{\hat{n}'*}$	*	*
JD25* [179] (Sec. 5.1)	$1\cdot[-\mu,\mu]^{\hat{n}}$	$1\cdot\mathbb{F}_{\hat{p}}$	$1\cdot\mathbb{F}_{\hat{p}}$	$1\cdot\mathbb{F}_{\hat{p}}$	$1\cdot\mathbb{F}^*_{\hat{p}}$	2 GA	3 GA
JD25 [179] (Sec. 5.2)	$1\cdot[-\mu,\mu]^{\hat{n}}$	$1\cdot\mathbb{F}_{\hat{p}}$	$1\cdot(\mathbb{F}_{\hat{p}}+E(\mathbb{F}_{\hat{p}}))$	$1\cdot(\mathbb{F}_{\hat{p}}+E(\mathbb{F}_{\hat{p}}))$	$\mathbb{Z}_{2\hat{q}-2}$	2 GA	3 GA
General Constructions with CCA Security from a PKE Scheme							
AFMR25 [29] (Sec. 5)	PKE.sk	PKE.pk	PKE.ct	–	PKE.pt	PKE.Enc + PKE.KG	–
General Constructions with CCA Security from an IND-CR-CPA UPKE **Scheme**							
DKW22 [108] (Sec. 7.4)			$+\lvert\pi\rvert$			$+\pi$	
DKW22 [108] (Sec. 7.5)			$+\lvert\pi\rvert$	$+\lvert\pi'\rvert$		$+\pi$	$+\pi'$
HLP22 [5] (Sec. 4.2)			$+\lvert\pi\rvert$	$\times 2$		$+\pi$	$\times 2$
HLP22 [5] (Sec. 4.4)			$+\lvert\pi\rvert$	$+\lvert\pi\rvert+\lvert\pi'\rvert$		$+\pi$	$+\pi+\pi'$
HPS23 [6] (Sec. 5)		$\times 2$					
HPS23 [6] (Sec. 6)		$\times 2$		$+\lvert\pi\rvert$			
General Constructions with CCA Security from an OW-CR-CPA UPKE **Scheme**							
AW23 [35]				$\times 2$			+Enc

Table 5.2: Sizes and computational costs of UPKE schemes. In [5], the parameter ζ takes values in $\{1,2\}$. The quantity $\hat{n}'$ denotes the number of distinct prime factors of $\hat{N}$, and $\hat{p}'=\mathcal{O}(\hat{N})$. The operations in [117] are not directly comparable to those of the other schemes, as the construction is based on a fundamentally different paradigm. Finally, the parameters μ and $\hat{n}$ are chosen so that $(2\mu+1)^{\hat{n}}\geq\#\mathsf{Cl}(O)$, where $\mathsf{Cl}(O)$ denotes the ideal class group of an order O. π and π' are the sizes of required proofs of knowledge. The reader can find even more detailed explanations and further remarks about the entries in [219].

		Security		Other Properties			
Construction	**Type**	**Model**	**Assumptions**	**A**	**∞ Upd**	**PUV**	**Syntax**
Constructions based on Pre-Quantum Assumptions							
JMM18 [182]	UPKE	C-IND-CPA	CDH, ROM	✓	✓	✘	L.A.
DKW22 [108] (Sec. 5.3)	UPKE	IND-CR-CPA	DDH	✘	✓	✘	L.S.
HLP22 [5] (Sec. 3.2)	UPKE	IND-CR-CPA	ζ-DCR	✘	✓	✘	L.S.
AFM24 [26]	UKEM	T-IND-CCA	AGM, co-CDH, ROM, SSEPOL*	✘	✓	✓	S
Constructions based on Lattices							
DKW22 [108] (Sec. 6.3)	UPKE	IND-CR-CPA	LWE	✘	✘	✘	L.A.
HPS23 [6] (Sec. 4)	UKEM	IND-CR-CPA	HNF-Aext LWE	✘	✘	✘	L.S.
AFMR25 [29]	UKEM	T-IND-CCA	LWE, ROM, SEPOK*	✘	✘	✓	S
ABL25 [19]	UPKE	IND-CR-CPA	LWE, PCE, ROM	✘	✓	✘	L.S.
Constructions based on Isogenies							
EJKM20 [119]* (Sec. 6)	UPKE	C-IND-CPA	CSSIDDH	✓	✓	✘	L.A.
DFV24 [117]*	UPKE	OW-PCA-U	SI-MPI	✓	✓	✘	L.A.
JD25 [179]* (Sec. 5.1)	UPKE	IND-CR-CPA	CSSIDDH	✘	✓	✘	L.S.
JD25 [179]* (Sec. 5.2)	UPKE	IND-CR-CCA	CSSIDDH + R_E + CSSIKoE	✘	✓	✘	L.S.
General Constructions with CCA Security from a PKE Scheme							
AFMR25 [29] (Sec. 5)	UKEM	T-IND-CCA	(ℓ, δ) corr., γ sp., Shifty-IND-CPA PKE, SE-POK	✘	✘/✓	✓	S
General Constructions with CCA Security from an IND-CR-CPA UPKE Scheme							
DKW22 [108] (Sec. 7.4)	UPKE	IND-CR-CCA	ot-f-tSSE-NIZK	✘	✘/✓	✘	L.S.
DKW22 [108] (Sec. 7.5)	UPKE	IND-CU-CCA	ot-f-tSSE-NIZK	✘	✘/✓	✓	L.S.
HLP22 [5] (Sec. 4.2)	UPKE	IND-CR-CCA	ζ-DCR, ROM, SS-NIZK	✘	✘/✓	✘	L.S.
HLP22 [5] (Sec. 4.4)	UPKE	IND-CU-CCA	ζ-DCR, ROM, Strong RSA, SS-NIZK	✘	✘/✓	✘	L.S.
HPS23 [6] (Sec. 5)	UKEM	IND-CR-CCA	ROM	✘	✘/✓	✘	L.S.
HPS23 [6] (Sec. 6)	UKEM	IND-CU-CCA	ROM, NIZK in ROM*	✘	✘/✓	✓	L.S.
General Constructions with CCA Security from an OW-CR-CPA UPKE Scheme							
AW23 [35]	UPKE	IND-CU-CCA	ROM	✘	✘/✓	✘	L.S.

Table 5.3: Security and other properties of UPKE schemes. Here, SEPOK stands for straightline-extractable proof of knowledge, ot-f-tSSE-NIZK is one-time strong true-simulation f-extractable NIZK for the function f (see Section 7.3 of [108] for details), SSEPOL is a strong simulation extractable proof of knowledge for DLog, and SS-NIZK is simulation sound NIZK. For further details, we invite the reader to check [219].

Scheme	κ	p	$\log q$	n	k	d	B	h	ℓ	ct size	pt size (in bits)	ctxt bits per ptxt bit
HPS23 [6]	115	5	21	3	–	256	1536	–	2^5	61.1 KB	≈ 594	≈ 823
AFMR25 [29]	115	5	21	3	–	256	1536	–	2^5	44.9 KB	≈ 594	≈ 605
HPS23 [6]	125	5	26	4	–	256	2048	–	2^{10}	88.7 KB	≈ 594	≈ 1194
AFMR25 [29]	125	5	26	4	–	256	2048	–	2^{10}	60.1 KB	≈ 594	≈ 810
HPS23 [6]	137	5	36	3	–	512	3072	–	2^{20}	130.7 KB	≈ 1190	≈ 878
AFMR25 [29]	137	5	36	3	–	512	3072	–	2^{20}	86.9 KB	≈ 1190	≈ 584
ABL25 [19]	128	2	13	7313	450	–	–	27	∞	11.6 KB	1	≈ 92800
ABL25 [19]	128	16	16	11000	550	–	–	26	∞	21.5 KB	4	≈ 43000

Table 5.4: Comparison of concrete lengths and parameters for lattice-based UPKE/UKEM schemes, highlighting the trade-offs between the security level, ciphertext size, and the number of supported updates, including constructions allowing an unbounded number of updates.

Protocol	Secret Key	Public Key	Ciphertext	Type	PQ	Security Level (bits)
AFM24 [26]	32B	144B	48B	UKEM	✘	128
AFMR25 [29]	640B	3.4KB	4.2KB	UKEM	✓	125
ECDHE (P256 curve) [208]	32B	32B	32B	KEM	✘	128
ML-KEM-512 [237]	1632B	800B	768B	KEM	✓	128

Table 5.5: Comparison of concrete lengths between standard KEMs and state-of-the-art UKEMs. Here, for elliptic curve group elements, we use point compression (i.e., storing one coordinate plus one bit). A careful reader may notice that sizes for AFM24 do not match those in Table 5.2. As they heavily depend on the implementation factors, we defer for a more detailed explanation to [219].

5.2 The Framework for basic PiQASO Services

A core objective of PiQASO is to design and implement a **PQaaS framework that will enable legacy Common Transmission Systems (CTS) to achieve end-to-end, quantum-resistant data protection and sharing services across cloud-based environments**. A key component of this PQaaS solution is the **PQ Data Encryption Server (PQ-DS)**, which supports all the PQ cryptographic mechanisms, for providing secure management, confidentiality services (different types of encryption/decryption), authentication, and fine-grained access control leveraging the **PiQASO PQC Ensemble cryptographic SDK**. The aim is to develop a PQ DS, using NIST-standardized PQ algorithms and ensuring seamless integration in existing CTS instances, thus **offering not only the expected security guarantees, but also the required scalability and crypto agility requirements for quantum-safe, cloud-based data sharing and storage services**, with limited disruption and modifications for users and service providers.

Besides the usual security properties of confidentiality, authentication and integrity, the PQ DS will innovate by offering strong **forward data secrecy** in data management (data sharing and storage), tailored for use in the target PQ cloud-based CTS solution. Thus, the proposed data management service will heavily rely on a novel, **quantum-resistant UPKE scheme from lattice assumptions**, which will be integrated in the PiQASO PQ Ensemble set of cryptographic functionalities. This new UPKE scheme will be designed to offer the required advanced forward secrecy guarantees, through regular updates of public/private key pairs (key rotation) of data sharing parties at specific time intervals (epochs). A key element in the proposed data sharing functionality that complements UPKE is the **Asynchronous Remote Key Generation (ARKG)** mechanism, a new technique that allows to **generate multiple unlinkable public keys from a given (base) public key, hence associated with a single secret key, in an asynchronous manner**.

Building upon the novel quantum-resistant UPKE and ARKG constructions, the envisioned end-to-end data management application will support **two distinct modes of operation, which rely on different trust assumptions**. Another key difference in the two modes of operation is that they are **driven by different time-evolving key chains, updated at regular epochs**. More concretely, the two modes are categorized as follows:

User-specific mode: This mode offers **encryption/decryption mechanisms of user-specific data payloads leveraging UPKE**, where the data sharing involves a specific data sender and multiple data receivers. In this scenario, a PQaaS entity, namely **the Key Provider (KP)** acting as part of the CTS, is responsible only for authenticating end-users. The encryption and decryption of communicating data is carried out by the end-users, leveraging the PiQASO PQC Ensemble cryptographic SDK. Hence **this mode of operation is fully decentralized, as all cryptographic operations for confidentiality are executed at the endpoints**, and no sensitive material is exposed to, or processed by the PQaaS infrastructure.

Attribute/policy-based mode: This mode offers **advanced policy-based encryption/decryption mechanisms of data payloads leveraging attribute-based UPKE**, where the data sharing involves a specific data sender and multiple data receivers, assuming that data receivers comply with a specific policy, defined by a trusted third party, the PQ DS. In this scenario, any sender wishing to share data encrypted using attribute-based UPKE under a defined policy or a set of attributes obtains the currently valid ML-KEM public key(s) from the PQaaS provider (the KP), while the eligible receiver will obtain the corresponding decryption keys if they meet specific eligibility criteria reflected by their verifiable attributes. In this mode of operation, the confidentiality operations (encryption/decryption) are executed by the PQ-DS, acting as a trusted entity. Hence, **this mode of operation is centralized, as it assumes that the entire cloud-based CTS is trusted to perform cryptographic operations and handle sensitive data and key material** on behalf of the end-users.

From a high-level view, the PiQASO system architecture considers a set of users $\mathcal{U} = \{U_1, \ldots, U_n\}$, where each user is equipped with a software-based PQC Ensemble cryptographic SDK, thus each user is able to access and deploy all core cryptographic functionalities offered by the PiQASO cloud-based data management solution. The users interact with the CTS, which comprises three main entities:

Identity Provider (IDP): The IDP is responsible for **verifying the identity of end-users by validating their public key certificate**. The IDP is also responsible for generating and securely (with symmetric encryption) **transmitting access tokens to the end-users**, consisting of arbitrary data, an expiration time, as well as attributes and permissions. These access tokens are used in other phases of the data management scheme for user authentication/authorization.

Key Provider (KP): The KP entity **offers key management and key provisioning functionalities and has a different role depending on the mode of operation** (user-specific or attribute/policy-specific). In the former case, the KP is responsible for the **authentication of end-users**. In the latter case, it is assigned with **the generation of all the required key material**, for different sets of cryptographic algorithms (PQC, especially ML-KEM, and symmetric cryptographic algorithms), as well as for **securely transferring the keys the the intended entity**, including end-users, the IDP for encrypting access tokens, and the PQ DS for executing the cryptographic operations.

PQ Data Server (DS): The PQ DS is the heart of the PiQASO cloud-based CTS as it executes all the required cryptographic operations, leveraging the PQC Ensemble cryptographic SDK. **It is the entity that receives, protects, stores, and controls access to the actual data**. Concretely, the PQ-DS **provides explicit authentication mechanisms** for end-users before accepting their data payloads, by verifying their access token generated by the IDP. Authentication is accomplished either via a challenge response protocol and PQ KEM/DEM encryption (as in ISO/IEC 9798), or through PQ TLS with the fully optimized and accelerated versions of PQC Ensemble ML-KEM and ML-DSA cryptosystems, or with the signature-less authentication TLS variant, namely KEMTLS. Furthermore, the PQ DS is in charge of ensuring the **confidentiality of data in-motion**, by deploying the PQ encryption/decryption operations offered by the PQC Ensemble, as well as for **guaranteeing the long-term confidentiality of data at-rest** through advanced encryption mechanisms offered by the newly designed (attribute-based) UPKE.

The full PiQASO cloud-based data management architecture is described in Figure 3.3. **In order to realize the envisioned PiQASO quantum-resistant data management service, we follow a stepwise approach**. As a first step, we introduce a simplified system model that captures the essential entities and fundamental operations and makes a formal security analysis feasible. More concretely, in this first step we concentrate on two main functionalities of the cloud-based data management system; file **upload** and file **retrieval**. Hence, we first focus on the cloud storage service, where all **files/data stored in the cloud are encrypted with a policy-based UPKE** scheme, based on ML-KEM. This approach captures the **forward secrecy requirement with respect to a pre-defined policy**, whereby ML-KEM encryption key pairs are updated at specific epochs. In addition, in this first model we consider the default UPKE functionality, where multiple ML-KEM public keys are derived by multiple secret keys, with the KP running the ML-KEM key generation routine multiple times. We note that this notion of UPKE is the predecessor of advanced, novel UPKE construction envisioned in PiQASO where multiple ML-KEM public keys are derived from a single secret key. Finally, the security model presented in this initial setting, abstracts away from additional cryptographic operations required by the cloud-based data storage scheme, for instance those related to end-user authentication. The security model is the entire PiQASO cloud-based data management system wil be enhanced in later phases of the project.

Figure 5.1 provides a visual representation of the simplified, cloud-based data storage scheme and it is a snapshot of the entire PQaaS description presented in Figure 3.3, as it focuses only on a subset

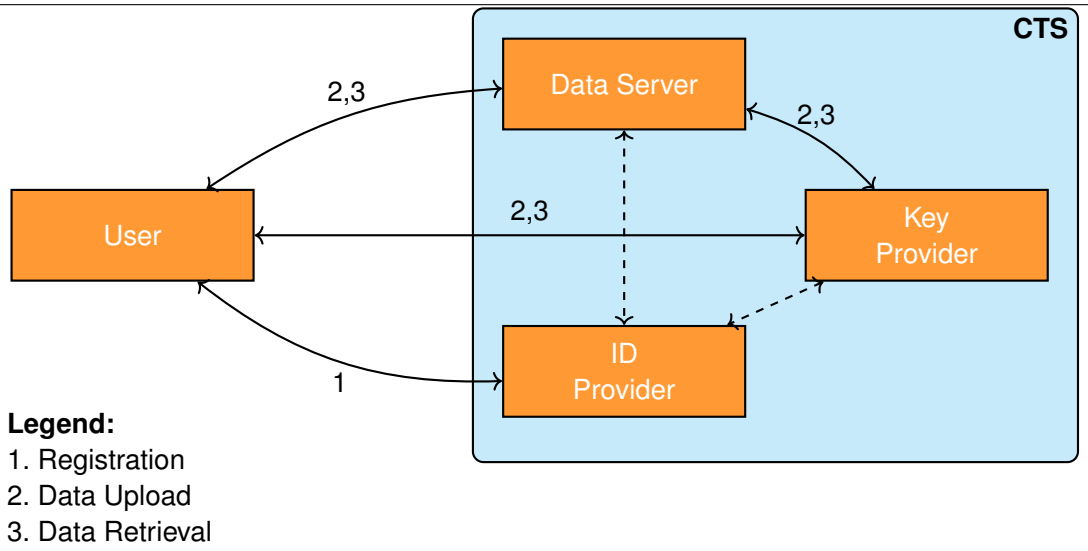

Figure 5.1: System model with registration, data upload, and data retrieval phases

of the envisioned functionalities of the PiQASO PQaaS solution. In particular, it illustrates the interactions between a user and the CTS, as well as the internal communication among the three entities that comprise the CTS, namely the IDP, the KP and PQ DS, during user registration, data upload and data retrieval. As stated earlier, the IDP, DS, and KP are assumed to be trusted, as in this initial step focuses on modelling the attribute/policy based mode of operation. This foundational model paves the way for the development of the full end-to-end privacy-preserving data management service, in subsequent phases of the PiQASO project and leveraging the novel UPKE functionality and the ARKG building block. In the following sections, we formally describe the syntax of the upload and retrieval procedures, define the security of the system, give a construction of these procedures, and prove their security according to the definitions previously stated. We assume that the authentication is already performed and valid, so no user of the system can impersonate any other user.

5.3 The Simplified Cloud Storage Scheme for Encrypted Data

We present a first step towards the security modelling of our simplified cloud-based storage scheme described in Figure 5.1, focusing in particular on **formalizing the notion of forward secrecy** via a standard UPKE scheme, as well as **incorporating policy-based access control** mechanisms for accessing encrypted data payloads. We start with the definition of **Hybrid Encryption**, which is a key building block in our scheme and proceed with the description of the core storage functionalities, including Setup, Upl, Ret and Update. We conclude this section with the definition of the forward secrecy property.

Notation. For a vector v of dimension n and values in a set S we use the notation $\mathrm{v}[1,\ldots,n] = s$, for a $s \in S$, to indicate that we initialize all the elements of the vector v to the value s. Also, by $\mathrm{v} \xleftarrow{\cup} s$ we mean that s is appended to v, resulting in a vector with dimension $n+1$ and last component s. For a database containing n key couples $\mathrm{DB_K}$, we write $\mathrm{DB_K}$.sk and mean $\{\mathrm{DB_K}[i].\mathrm{sk}\}_{i=1}^{n}$, and the same for $\mathrm{DB_K}$.pk.

5.3.1 Hybrid Encryption Scheme

The simplified cloud-based storage scheme considered in our scenario, assumes a hybrid encryption protocol that combines PQ encryption (ML-KEM) and symmetric encryption (AES-256 in combination

with a mode of operation). In particular, the hybrid encryption function executes two encryption subroutines; a symmetric encryption to encrypt the actual message m using a secret key k, and a public key encryption for encrypting the symmetric key k with an ML-KEM public key pk (key wrapping). Thus, the hybrid encryption process outputs two ciphertexts; one corresponding to the symmetric encryption of the message and the second is the public key encryption of the symmetric key. In this setting, the hybrid decryption requires first an ML-KEM decryption with the corresponding secret key sk to obtain the secret key k, and then the symmetric decryption with the symmetric key k to obtain the actual message m. We give the definition of hybrid key encryption, as it will be used later in the chapter:

Definition 5 (Hybrid Encryption Scheme)**.** *A hybrid encryption scheme is a tuple*

$$\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec})$$

defined as follows.

- $\mathsf{HE.KeyGen}(1^\lambda)$ *is a probabilistic algorithm that, on input the security parameter* λ*, outputs a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$*.*
- $\mathsf{HE.Enc}(\mathsf{pk}, m)$ *is a probabilistic encryption algorithm that, on input a public key* pk *and a message* $m \in \{0,1\}^*$*, outputs a ciphertext* $C = (c_1, c_2)$*;*
- $\mathsf{HE.Dec}(\mathsf{sk}, c)$ *is a deterministic decryption algorithm that, on input a secret key* sk *and a ciphertext* $C = (c_1, c_2)$*, returns a message* m *or a special error symbol* $\perp$*.*

The scheme is δ*-correct if for all messages* m *and all key pairs* $(\mathsf{sk}, \mathsf{pk})$ *output by* $\mathsf{KeyGen}(1^\lambda)$*,*

$$\Pr\left[\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{HE.Enc}(\mathsf{pk}, m)) = m\right] \geq 1 - \delta.$$

As stated earlier, the function $\mathsf{HE.Enc}()$ executes two types of encryption. Hence, the ciphertext C is composed of two parts (c_1, c_2), defined as:

$$c_1 = \mathsf{Enc}^{\mathrm{KEM}}(\mathsf{pk}, k) \qquad \text{and} \qquad c_2 = \mathsf{Enc}^{\mathrm{AES}}(k, m)$$

We also recall the definition of IND-CCA2 security for a hybrid encryption scheme:

Definition 6. *We say that an hybrid encryption scheme* HE *is IND-CCA-2 secure if, for any polynomial time adversary* $\mathcal{A}$*,*

$$\mathbb{P}[\mathsf{Exp}^{\textit{ind-cca2}}_{\mathcal{A},\mathsf{HE}}(1^\lambda) = 1] - \frac{1}{2} \leq negl_{cca}(\lambda),$$

for a negligible function $negl_{cca}$ *in the security parameter.*

$\mathsf{Exp}^{\mathsf{ind\text{-}cca2}}_{\mathcal{A},\mathsf{HE}}(1^\lambda)$:

1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$
2 : $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$
3 : **if** $|m_0| \neq |m_1|$ **then return** $\perp$
4 : $b \xleftarrow{\$} \{0,1\}$
5 : $c^* \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, m_b)$
6 : $b' \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(c^*)$
7 : **return** $b \stackrel{?}{=} b'$

The experiment $\mathsf{Exp}^{\text{ind-cca2}}_{\mathcal{A},\mathsf{HE}}(1^\lambda)$ models indistinguishability under adaptive chosen-ciphertext attack for a hybrid encryption scheme HE. The challenger first generates a key pair $(\mathsf{sk}, \mathsf{pk})$ using HE.KeyGen. The adversary $\mathcal{A}$ is given access to a decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ and outputs two equal-length challenge messages (m_0, m_1) along with some state information st. A random challenge bit $b \in \{0, 1\}$ is sampled, and the challenge ciphertext $c^* \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, m_b)$ is computed and returned to the adversary. The adversary continues to have adaptive access to the decryption oracle, except on the challenge ciphertext c^*, and outputs a guess b'. The experiment outputs 1 if $b' = b$, and 0 otherwise.

5.3.2 A Forward-Secure Cloud Storage Scheme

We proceed with the description of our modelling approach for the simplified cloud storage scheme. In this **initial modelling attempt, our primary objective is to capture the notion of forward secrecy** by assuming an UPKE in its simplest form, where ML-KEM key pairs are updated regularly. This ensures that the leakage of a secret key at a given point does not affect messages that were stored in the cloud and encrypted with keys from previous epochs. For simplicity, and for our goals, we assume that the cloud updates all the keys at fixed time intervals. Furthermore, **our model captures the concept of policy-based access control**, whereby users are granted access to specific encrypted data, as long as they comply with a specific policy. This implies that along with the ciphertext, each user circulates an attribute-based policy to be enforced by the cloud.

5.3.2.1 Syntax of Cloud Storage Scheme

In our analysis, we define a cloud storage scheme as a set of four probabilistic polynomial time algorithms $(\mathsf{Setup}, \mathsf{Upl}, \mathsf{Update}, \mathsf{Ret})$ together with a hybrid encryption scheme $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec})$ (as defined in the previous subsection). Below we describe the four functions, as well as the corresponding entities that participate in each function and their actions.

- **System Setup** ($\mathsf{Setup}(1^\lambda, n)$). The procedure takes as input a security parameter λ and an integer n (i.e., the number of users), and returns four initialized databases (lists), namely a database for files $\mathsf{DB_f}$, a database for keys $\mathsf{DB_K}$, a database for policies $\mathsf{DB_P}$ and a database for the epochs $\mathsf{DB_t}$. In particular, $\mathsf{DB_f}$ and $\mathsf{DB_P}$ are initialized as empty databases in the beginning. The $\mathsf{DB_K}$ contains n pairs of public key encryption keys for the scheme HE, along with the epoch related to each key pair. Finally, $\mathsf{DB_t}$ keeps track of all the epochs. The databases $\mathsf{DB_P}, \mathsf{DB_f}, \mathsf{DB_t}$ and the public keys of $\mathsf{DB_K}$ (i.e., $\mathsf{DB_K.pk}$) are publicly accessible after generation;
- **Data Upload** ($\mathsf{Upl}(f, P)$). It takes as input a file $f \in F$, representing the user's data payload, and a policy $P \in 2^{[n]}$, possibly changes the state of $\mathsf{DB_f}$ and $\mathsf{DB_P}$ and outputs a file identification string $\mathsf{ID_f}$, or a special error symbol $\perp$. The Upl function is executed by the PQ DS, which is in charge for deploying the HE.Enc function to encrypt data on behalf of a user.
- **Key update** ($\mathsf{Update}()$). Updates public key pairs in the database $\mathsf{DB_K}$ as the epoch evolves. In this analysis, we consider the standard UPKE approach, where an update of n key pairs implies the re-execution of the KeyGen function of the HE scheme and replacing the old key pairs with the new ones. For simplicity, we introduce the notation $\mathsf{DB_K.pk}^e$ to indicate all the public keys of time epoch e, that is $\{\mathsf{DB_K}[i][e].\mathsf{pk}\}_{i=1}^n$ (i.e., all the public keys of n users, related to epoch e). Furthermore, $\mathsf{DB_K.pk}$ indicates all the public keys (of all the epochs). This function is executed by the KP, upon request from the PQ DS.

- **Data Retrieval** ($\mathsf{Ret}(\mathsf{ID_U}, \mathsf{ID_f})$). This procedure takes as input a user identification string $\mathsf{ID_U}$ and a file identification string $\mathsf{ID_f}$ and outputs either a file f or a special error symbol $\perp$. The function Ret is executed by the PQ DS as it involves the cryptographic operation HE.Dec.

If a file f has identity string $\mathsf{ID_f}$ and it is encrypted in the time epoch e, then $\mathsf{DB_t}[\mathsf{ID_f}] = e$. As we keep track of which epoch a key couple is related to, the database of keys is a table. In particular, the keys related to a user with identification string $\mathsf{ID_U}$ and an epoch e, which we indicate as $(\mathsf{sk}^e_{\mathsf{ID_U}}, \mathsf{pk}^e_{\mathsf{ID_U}})$, are stored in the entry $[\mathsf{ID_U}][e]$ of $\mathsf{DB_K}$, that is $\mathsf{DB_K}[\mathsf{ID_U}][e] = (\mathsf{sk}^e_{\mathsf{ID_U}}, \mathsf{pk}^e_{\mathsf{ID_U}})$. The current epoch is indicated by a global value, which is incremented every time the epoch evolves and it is included in the policy associated with the user with identity $\mathsf{ID_U}$. Since the keys are stored in a global database, and the index related to the time epoch is also a global value, the algorithm Update does not have any input or output. Based on this syntax, we obtain the following definition of correctness.

Definition 7 (Correctness)**.** *From these algorithms, we expect the following notion of correctness:*

$$\mathbb{P}[\mathsf{Ret}(\mathsf{ID_U}, \mathsf{Upl}(f, P)) = f | \mathsf{ID_U} \in P] \geq 1 - \delta,$$

for $\mathsf{DB_f}, \mathsf{DB_P}, \mathsf{DB_K}, \mathsf{DB_t} \leftarrow \mathsf{Setup}(1^\lambda, n)$*, any file* f *and any policy* P*, for the upload happening at any epoch* t*, and for a negligible function* $\delta(\lambda)$*.*

5.3.2.2 Instantiation based on Hybrid Encryption

We now describe in more detail how to instantiate the algorithms of our simplified cloud storage scheme. Let $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec})$ be an IND-CCA2-secure hybrid encryption scheme. In our model, we assume that a user with an identification string $\mathsf{ID_U}$ obtained from IDP in the registration phase, has already authenticated with the CTS, particularly with the IDP, and has already established a secure communication channel with the CTS. As mentioned earlier, we exclude the authentication phase from our model for now.

The user $\mathsf{ID_U}$ who wishes to store their encrypted file in the cloud, generates a random secret key k, using a Key Derivation Function (KDF) provided by the PiQASO PQC Ensemble crypto SDK. This secret key is used to encrypt their file (data payload) with symmetric encryption (AES-256). Then, the user encrypts the symmetric key k with the ML-KEM public key of the CTS. Hence, the user executes the hybrid encryption function HE.Enc to produce a ciphertext $C = (c_1, c_2)$, such that $c_1 = \mathsf{Enc}^{\mathrm{KEM}}(\mathsf{pk_{CTS}}, k)$ and $c_2 = \mathsf{Enc}^{\mathrm{AES}}(k, f)$, and sends the ciphertext C containing both the encrypted key and the encrypted file to the PQ DS. Note that in scenarios where a user is not able to perform ML-KEM encryption and produce c_1 on its own, e.g., due to legacy constraints, the secret key k can be encrypted using any other KEM mechanism supported by the user and the CTS, or sent to PQ DS over an existing secure channel (e.g. TLS).

The PQ DS communicates with the KP, in order for the latter to generate an ML-KEM key pair $(\mathsf{sk}^e_{\mathsf{ID_U}}, \mathsf{pk}^e_{\mathsf{ID_U}})$ for the user with $\mathsf{ID_U}$, related to the current epoch e, using the PiQASO PQC Ensemble crypto SDK. The PQ DS obtains the secret key k, e.g., by decrypting c_1 with its secret key $\mathsf{sk_{CTS}}$, or by receiving it over the existing secure channel. It then re-encrypts k with the new ML-KEM public key $\mathsf{pk}^e_{\mathsf{ID_U}}$ on behalf of the user, producing the ciphertext $c_1' = \mathsf{Enc}^{\mathrm{KEM}}(\mathsf{pk}^e_{\mathsf{ID_U}}, k)$. Then, PQ DS stores the new key pair in the database entry $\mathsf{DB_K}[\mathsf{ID_U}][e]$, as well as the ciphertexts (c_1', c_2) in the database entry $\mathsf{DB_f}[\mathsf{ID_U}]$. Hence, in order for a user to access their encrypted file, they need to be able to access their symmetric encryption key k first.

We note that when a key pair $(\mathsf{sk}^e_{\mathsf{ID_U}}, \mathsf{pk}^e_{\mathsf{ID_U}})$ associated with a user with identity $\mathsf{ID_U}$, created at epoch e is updated, the user should still be able to have access to their encrypted data. This requires that the epoch

e is part of the policy associated with the user $\mathsf{ID_U}$, in which case a user can run the KEM key generation function to retrieve ML-KEM key pairs at previous epochs.

We now proceed with the concrete description of the algorithms $(\mathsf{Setup}, \mathsf{Upl}, \mathsf{Update}, \mathsf{Ret})$.

The setup algorithm initializes a global counter count_f, which is used to assign unique identifiers to uploaded files. While we fix the initial value to 0 and increase it every time a file is uploaded for convenience, any mechanism that guarantees uniqueness for the identification of files would suffice. Next, three empty databases are initialized: $\mathsf{DB_f}$ for storing encrypted files and $\mathsf{DB_P}$ for storing access policies and $\mathsf{DB_t}$ for storing the epochs. Finally, a key database $\mathsf{DB_K}$ is initialized with one key pair per user, at epoch 0. Each entry $\mathsf{DB_K}[i]$ is generated by running the key generation algorithm of the encryption scheme. The algorithm outputs the initialized databases. The Setup procedure is summarized as follows:

$\mathsf{Setup}(1^\lambda, n)$:

1 : **global** $\mathsf{count}_f \leftarrow 0,\ t \leftarrow 0,\ \mathsf{DB_f},\ \mathsf{DB_P},\ \mathsf{DB_t},\ \mathsf{DB_K}[1,\ldots,n][0] \leftarrow (\epsilon, \epsilon)$
2 : **for** $i = 1$ **to** n **do**
3 : $\quad \mathsf{DB_K}[i][0] \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$
4 : **return** $(\mathsf{DB_f}, \mathsf{DB_P}, \mathsf{DB_t}, \mathsf{DB_K})$

The upload algorithm first checks whether the access policy P is non-empty; if empty, it returns an error symbol $\perp$. This is because each file must remain retrievable by at least the user who uploads it to CTS. If P is not empty, a fresh file identifier $\mathsf{ID_f}$ is assigned for the uploaded file by incrementing the global counter count_f. For each user $j \in P$, i.e., for all users' identifiers in the policy, the function $\mathsf{HE.Enc}$ is executed by the PQ DS with input the public key of the user with identify j and the file f, and that public key is stored in the jth entry of the database of keys. This indicates that the file f is already encrypted with a symmetric key k_j generated by the user j, hence the PQ DS encrypts the symmetric key k_j with the public key of the user at the current epoch t. The resulting ciphertext is stored together with the corresponding user identifier in the file database $\mathsf{DB_f}$ at index $\mathsf{ID_f}$.

More concretely, the pair (j, C), i.e., the user's identifier and ciphertext C, are appended to the database entry $\mathsf{DB_f}[\mathsf{ID_f}]$. Once all encryptions are computed, the list is stored, while the policy P is stored in the policy database $\mathsf{DB_P}$ and the epoch t in the database $\mathsf{DB_t}$, at the same index $\mathsf{ID_f}$. In particular, for a file identified by $\mathsf{ID_f}$ and associated with policy P, we have $\mathsf{DB_f}[\mathsf{ID_f}]$ containing all authorized encryptions of f, and $\mathsf{DB_P}[\mathsf{ID_f}] = P$. In addition, $\mathsf{DB_t}[\mathsf{ID_f}] = t$ is the epoch related to the file f with identifier $\mathsf{ID_f}$. The algorithm returns the file identifier $\mathsf{ID_f}$. The Upl procedure is summarized in the following steps:

$\mathsf{Upl}(f, P)$:

1 : **if** $P = \emptyset$ **then return** $\perp$
2 : $\mathsf{ID_f} \leftarrow \mathsf{count}_f + +$
3 : **for all** $j \in P$ **do**
4 : $\quad C \leftarrow \mathsf{HE.Enc}(\mathsf{DB_K}[j].\mathsf{pk}^t, f)$
5 : $\quad \mathsf{DB_f}[\mathsf{ID_f}] \xleftarrow{\cup} (j, C)$
6 : $\mathsf{DB_P}[\mathsf{ID_f}] \leftarrow P$
7 : $\mathsf{DB_t}[\mathsf{ID_f}] \leftarrow t$
8 : **return** $\mathsf{ID_f}$

The retrieval algorithm first verifies that the request is well-formed: the file identifier $\mathsf{ID_f}$ must correspond to an existing entry in $\mathsf{DB_f}$, and the user identifier $\mathsf{ID_U}$ must be valid. If either check fails, the algorithm returns $\perp$. If the checks succeed, the policy associated with the requested file is retrieved from $\mathsf{DB_P}$. Access control is enforced by verifying that $\mathsf{ID_U}$ is included in the policy. Only in this case does the

algorithm decrypt the ciphertext corresponding to user $\mathsf{ID_U}$ using the secret key $\mathsf{DB_K}[\mathsf{ID_U}].\mathsf{sk}$, and return the recovered file. Otherwise, an error is returned.

$\mathsf{Ret}(\mathsf{ID_U}, \mathsf{ID_f})$:

1 : **if** $\mathsf{ID_f} > |\mathsf{DB_f}|$ **or** $\mathsf{ID_U} > n$ **then return** $\perp$ //here, equivalently, can be checked if $\mathsf{ID_f}$ is a valid ID string
2 : $P \leftarrow \mathsf{DB_P}[\mathsf{ID_f}]$
3 : $e \leftarrow \mathsf{DB_t}[\mathsf{ID_f}]$
4 : **if** $\mathsf{ID_U} \notin P$ **then return** $\perp$
5 : **for** $tupl \in \mathsf{DB_f}[\mathsf{ID_f}]$ **do**
6 : **if** $tupl_1 == \mathsf{ID_U}$ **then**
7 : $f \leftarrow \mathsf{HE.Dec}(\mathsf{DB_K}[\mathsf{ID_U}].\mathsf{sk}^e, tupl_2)$
8 : **return** f
9 : **return** $\perp$

The Update procedure describes the current standard approach for UPKE schemes, where updating the pair of public and private keys requires the KP to re-execute the KeyGen function of the HE scheme n times, assuming n users. The following procedure describes this functionality, where a new pair of ML-KEM keys $(\mathsf{sk}_i^t, \mathsf{pk}_i^t)$ is updated at a new epoch t and stored in the database $\mathsf{DB_K}$. In particular, the entry $\mathsf{DB_K}[i][t]$, contains the pair of keys of the user with identity i, at epoch t.

$\mathsf{Update}()$:

1 : $t++$
2 : **for** $i = 1$ **to** n **do**
3 : $\mathsf{sk}_i^t, \mathsf{pk}_i^t \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$
4 : $\mathsf{DB_K}[i][t] \xleftarrow{\cup} (\mathsf{sk}_i^t, \mathsf{pk}_i^t)$

Note that the correctness of the instantiation follows directly from the correctness of the underlying HE encryption scheme.

5.3.2.3 Definition of Forward Secrecy for the Cloud Storage Scheme

We now present a first attempt to describe the security model of the simplified cloud storage scheme, capturing the forward secrecy property and based on the definition of the hybrid encryption scheme described in this section. The experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{forwSec}}(1^\lambda, n)$ aims to capture this property.

The setup phase is executed in order to initialize databases $\mathsf{DB_f}, \mathsf{DB_K}, \mathsf{DB_t}$ and $\mathsf{DB_P}$ of size n. In addition, a vector describing corrupted users is initialized, as well as the current epoch t which is set to 0. Then, the adversary provides two files to be uploaded along with a policy P^*. One of these two files is uploaded and stored, and the identification string $\mathsf{ID_f}^*$ of this file is returned to the adversary. The adversary, then, has access again to all the stored files, can further upload and retrieve files reflecting adaptive security. Throughout the entire game, the adversary can also query three oracles:

- A **compromise oracle** $\mathcal{O}_{\mathsf{comp}}$ that, on input an index identifying a user i, returns its secret key sk_i and marks the user as corrupted in the corruption vector;
- An **upload oracle** $\mathcal{O}_{\mathsf{Upl}}$, that on input a policy and a file runs the upload algorithm;
- A **retrieve oracle** $\mathcal{O}_{\mathsf{Ret}}$, that on input a file identity string and a user identity, runs the retrieve

algorithm, under the condition that, when the challenge identity string is provided, the user identity is not part of the policy related to challenge identity, that is $\mathsf{Ret}(i, \mathsf{ID_f}^*) = \perp$ for each $i \in P^*$.

- We allow the adversary to see encryptions and have descriptions under different epochs. In particular, the adversary can query an **update oracle** $\mathcal{O}_{\mathsf{Update}}$, which updates all the keys in the current epoch and increases the epoch counter e. Moreover, to model forward security, at the end of the game, the adversary is given access to secret keys in the current epoch (which is a different epoch compared to the keys generating the challenge ciphertext). This captures exactly the idea of forward security: a key compromise should not affect messages encrypted in the past.

The adversary can win the game if it can guess with probability more than random if one or the other file has been uploaded. Also, to avoid trivial wins, we check that the adversary has not corrupted any user in the policy of the challenge file.

$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{forwSec}}(1^\lambda, n)$:

1 : $\mathsf{DB_f}, \mathsf{DB_P}, \mathsf{DB_K}, \mathsf{DB_t} \leftarrow \mathsf{Setup}(1^\lambda, n)$
2 : $\mathsf{corr}[1, \ldots, n] = false$
3 : $t = 0$
4 : $(f_0, f_1, P^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Upl}}, \mathcal{O}_{\mathsf{Ret}}, \mathcal{O}_{\mathsf{comp}}, \mathcal{O}_{\mathsf{Update}}}(\mathsf{DB_f}, \mathsf{DB_P}, \mathsf{DB_t}, \mathsf{DB_K.pk})$
5 : $b \xleftarrow{\$} \{0, 1\}$
6 : $st \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Upl}}, \mathcal{O}_{\mathsf{Ret}}, \mathcal{O}_{\mathsf{comp}}, \mathcal{O}_{\mathsf{Update}}}(\mathsf{DB_f}, \mathsf{DB_P}, \mathsf{DB_t}, \mathsf{DB_K.pk})$
7 : $\mathsf{ID_f}^* \leftarrow \mathsf{Upl}(f_b, P^*)$
8 : $\mathsf{Update}(\mathsf{DB_K.pk}^t)$
9 : $b' \leftarrow \mathcal{A}(\mathsf{ID_f}^*, \mathsf{DB_f}, \mathsf{DB_P}, \mathsf{DB_t}, \mathsf{DB_K.pk}, \mathsf{DB_K.sk}^t)$
10 : **return** $b \stackrel{?}{=} b' \wedge \forall i \in P^*.\neg\mathsf{corr}[i]$

Definition 8 (Forward-Secure Cloud Storage)**.** *An updatable cloud storage server is said to be forward-secure if, for any probabilistic polynomial time algorithm $\mathcal{A}$, we have that*

$$\left| \mathbb{P}[\mathsf{Exp}_{\mathcal{A}}^{\mathit{forwSec}}(1^\lambda, n) = 1] - \frac{1}{2} \right| \leq negl(\lambda),$$

for a negligible function $negl$, and for any fixed n.

We note that the modelling presented in this section represents a preliminary stage, and serves as the basis towards the security modelling of the PiQASO cloud-based data management system with forward secrecy based on a predecessor UPKE scheme. At this initial phase, the model design focuses on establishing the core architecture, identifying the involved entities and their role, and validating the feasibility of the proposed approach. In subsequent phases of the project, this initial framework will be significantly refined and an extended security analysis will be presented. This analysis will aim to incorporate additional functionalities, capture a broader range of security properties, and evaluate the system against sophisticated attack vectors, thereby ensuring that it meets the envisioned security guarantees.

Chapter 6

Functional Encryption in PiQASO

Today, data privacy has become a major concern as personal and possibly sensitive information is stored and processed online. Traditional and widely-used cryptographic techniques, such as Public-Key Encryption (PKE) provide strong confidentiality guaranties by ensuring that only authorized parties can decrypt data.

However, the opaque nature of encrypted data limits its application -especially in scenarios where controlled computation over sensitive data is desirable, such as privacy-preserving data analytics.

Advanced cryptographic techniques such as Homomorphic Encryption (HE) and Functional Encryption (FE) attempt to bridge this gap by enabling computations over ciphertexts while allowing fine-grained access to encrypted data. In both HE and FE, specific functions can be computed directly on ciphertexts, revealing only the output of the computation without getting access to the individual values.

These techniques have gained increased attention in recent years, as they represent a significant step forward showing that data privacy can go hand in hand with utility.

This chapter is organised as follows. Section 6.2 recalls the Learning With Errors and its variants. Section 6.3 provides an overview of Homomorphic Encryption schemes. Section 6.4 and Section 6.5 present the construction of AoS in detail, including its algorithms, system model, performance benchmarks, and security analysis, culminating in its integration within the PiQASO framework.

6.1 Notations and Preliminaries

For a positive integer n, $[n]$ is the set $\{1, \ldots, n\}$. Vectors are denoted by bold lowercase letters, and matrices by bold uppercase letters. We denote the modulus operation by $[x]_q = x \; mod \; q$. We denote $y \leftarrow\!\!\$ \; \mathcal{Y}$ when y is chosen uniformly at random from a set $\mathcal{Y}$. We denote $\mathbb{Z}_q$ the set of integers in $(-q/2, q/2]$, $\mathbb{Z}_q[X]$ the set of polynomials in X and $\mathcal{R}_q = \mathbb{Z}_q[X]/P(X)$ the quotient ring, if $P(X)$ is irreducible in $\mathbb{Z}_q[X]$. An algorithm $\mathcal{A}$ is said to be (probabilistic) polynomial time ((P)PT) if it is a randomized algorithm such that there exists a polynomial P such that for any input y, the running time of $\mathcal{A}(y)$ is bounded by $|P(y)|$. A function $f : \mathbb{N} \mapsto \mathbb{R}$ is negligible if $\forall c \in \mathbb{N}, \exists \varepsilon_0 \in \mathbb{N}$ such that $\forall \epsilon \geq \epsilon_0$, $f(\epsilon) < \epsilon^{-c}$. An arbitrary negligible function is denoted $negl(\cdot)$. Two elements a, b are said to be indistinguishable, denoted $a \sim b$, if $\Pr[\beta' = \beta \,;\, c \leftarrow \beta \cdot a + (1-\beta) \cdot b \,|\, \beta \leftarrow\!\!\$ \; \{0,1\}] = 1/2 + negl(\cdot)$ where β' is the adversary's guess.

Co-funded by the European Union ECCC

6.2 Learning With Errors (LWE) and its variants

LWE showcases its comparable worst-case hardness properties through a quantum reduction [274]. With the diversification of cryptographic protocols, new variants of this problem have emerged, such as RLWE, MLWE and TLWE. These variants reduce to the same hardness assumption, which we refer to as *Generalized* Learning With Errors (GLWE). For the sake of clarity, we formally define only the GLWE problem.

It should be noted that the first version of the PiQASO Functional Encryption is based on the Ring LWE (RLWE) problem. However, considering recent advancements detailed in Chapter 4, PiQASO is also considering the transformation of this Functional Encryption construction to be based on the Module LWE (MLWE) problem.

6.2.1 Generalized Learning With Errors (GLWE)

Definition 9 (GLWE sample)**.** *Let $\mathcal{R}$ be a ring and denote $\mathcal{R}_N[X] := \mathcal{R}[X]/(X^N + 1)$ for N a power of 2. For a positive integer k, a vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N[X]^k$ and a polynomial $\mu \in \mathcal{R}_N[X]$, we define*

$$\texttt{GLWE}_{\mathbf{s}}(\mu) = (\mathbf{a}, \mu + \langle \mathbf{s}, \mathbf{a} \rangle + e)$$

where $\mathbf{a} \xleftarrow{\$} \mathcal{R}_N[X]^k$ and $e \xleftarrow{\chi} \mathcal{R}_N[X]$ for an error distribution χ.

Note that the trivial sample $\texttt{GLWE}_{\mathbf{s}}(0_{\mathcal{R}}) = (\mathbf{a}, \langle \mathbf{z}, \mathbf{a} \rangle + e)$ is called the GLWE distribution.

Definition 10 (GLWE problem)**.** *Let $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N[X]^k$ be a secret vector.*

- *Search GLWE: Given $\texttt{GLWE}_{\mathbf{s}}(0_{\mathcal{R}}) = (\mathbf{a}, \langle \mathbf{z}, \mathbf{a} \rangle + e)$ and the distribution χ, find $\mathbf{s}$;*
- *Decisional GLWE: Given a pair $(\mathbf{a}, b) \in \mathcal{R}_N[X]^{k+1}$ and the distribution χ, decide if b is chosen at random, i.e. $b \xleftarrow{\$} \mathcal{R}_N[X]$, or if it follows a $\texttt{GLWE}$ distribution that is $(\mathbf{a}, b) = \texttt{GLWE}_{\mathbf{s}}(0_{\mathcal{R}})$.*

The classical LWE problem stated by Regev corresponds to the case $N = 1$ and $\mathcal{R} = \mathbb{Z}_q$, for q a power of a prime. In short, an LWE sample is given by $\texttt{LWE}_{\mathbf{s}}(\mu) = ((a_1, \ldots, a_k), \mu + \langle \mathbf{s}, \mathbf{a} \rangle + e)$, for $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{Z}_q^k$, $\mathbf{s} \in \mathbb{Z}^k$, $\mu \in \mathbb{Z}_q$ and $e \xleftarrow{\chi} \mathbb{Z}_q$.

Definition 11 (Decisional LWE problem)**.** *Given the pair $(A, \mathbf{b})$, where $A \in \mathbb{Z}_q^{(m \times n)}$, $\mathbf{b} \in \mathbb{Z}_q^m$, where $m > n$; distinguishing the following cases for $\mathbf{b}$:*

- *$\mathbf{b}$ is chosen uniformly at random from $\mathbb{Z}_q$.*
- *$\mathbf{b} \leftarrow A \cdot \mathbf{s} + \mathbf{e} \pmod q$ where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\mathbf{e} \xleftarrow{\chi} \mathbb{Z}$.*

Definition 12 (Search LWE problem)**.** *Let q be a prime and $\mathcal{D}_{\alpha q}$ the discrete Gaussian of width αq for $\alpha < 1$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be a secret vector. Given m independent samples $(a_i, \langle a_i, \mathbf{s} \rangle + e_i)$ where $a_i \xleftarrow{\$} \mathbb{Z}_q^n$ and $e_i \xleftarrow{\$} \mathcal{D}_{\alpha q}$, find $\mathbf{s}$.*

Public-Key Encryption from LWE. We describe a public-key encryption scheme based on the Learning With Errors (LWE) problem.

Definition 13 (LWE-based Encryption Scheme)**.** *An LWE-based encryption scheme for message space $\mathcal{M} = \mathbb{Z}_2$, consists of three algorithms* LWE.Setup, LWE.Enc, LWE.Dec *defined as follows:*

- LWE.Setup$(\lambda) \rightarrow (\mathtt{pk}, \mathtt{sk})$
  select $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$*, and generate m vectors* $\mathbf{a_i} \xleftarrow{\$} \mathbb{Z}_q^n$*, and m error values* $e_i \xleftarrow{\$} \mathcal{D}$*. We set* $(b_i = \langle \mathbf{a_i}, \mathbf{s} \rangle - 2.e_i)$ *for* $i = 1, \ldots, m$*. Then output the* $\mathtt{sk} = \mathbf{s}$ *and* $\mathtt{pk} = ((\mathbf{a_1}, b_1), \ldots, (\mathbf{a_m}, b_m))$*.*
- LWE.Enc$(\mathtt{pk}, \mu) \rightarrow (\mathbf{c}, d)$
  to encrypt the single bit message $\mu \in \mathbb{Z}_2$*, first we select* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^m$*, the ciphertext is then a pair of* $\mathbf{c}$*, and an integer* d*, such that* $\mathbf{c} \leftarrow \sum_{i=1}^{m} r_i.\mathbf{a_i}$*, and* $d \leftarrow \mu - \sum_{i=1}^{m} r_i.b_i$*.*
- LWE.Dec$(\mathtt{sk}, (c, d)) \rightarrow \mu$
  decrypting the ciphertext is performed by calculating $\mu \leftarrow [[\langle \mathbf{c}, \mathbf{s} \rangle + d]_q]_2$

6.2.2 Learning with errors over a ring (RLWE)

For large values of N, encryption schemes based on LWE become too expensive for practical use in cryptography, especially in HE. To decrease computational complexity, most HE schemes rely on the Ring Learning With Errors (RLWE) problem, a variant of LWE that samples the keys in a polynomial ring.

This variant corresponds to the case $k = 1$, that is, the RLWE sample given by $\mathtt{RLWE}_{\mathbf{s}}(\mu) = (a(X), \mu + a(X) \cdot s(X) + e(X))$, for $a(X) \in \mathbb{Z}_{q,N}[X]$, $s(X) \in \mathbb{Z}_N[X]$, $\mu(X) \in \mathbb{Z}_{q,N}[X]$ and $e(X) \xleftarrow{\chi} \mathbb{Z}_{q,N}[X]$.

Consider the ring $\mathcal{R} = \mathbb{Z}[X]/\langle f(X) \rangle$, where $f(X) = X^N + 1$ is a cyclotomic polynomial for N a power of two, let q be a sufficiently large prime modulus such that $q \equiv 1 \mod 2N$, $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle f(X) \rangle$ be the ring of degree n polynomials modulo $f(X)$ with coefficients in $\mathbb{Z}_q$, and χ represent the error distribution over the ring $\mathcal{R}$. The RLWE problemis formally defined as follows:

Definition 14 (Ring-Learning With Errors (RLWE) distribution)**.** *Let the parameters be* $(n, m, q, \mathcal{X}_{\mathsf{sk}}, \mathcal{X}_e)$ *where* $n \geq 1$ *is an integer,* m *is the number of samples,* q *is the modulus,* $\mathcal{X}_{\mathsf{sk}}$ *is the secret distribution over* $\mathbb{Z}[X]/\left(X^N + 1\right)$ *and* $\mathcal{X}_e$ *is the error distribution over* $\mathbb{Z}_q[X]/\left(X^N + 1\right)$*.*

For a secret $\mathsf{sk}(x) \in \mathbb{Z}_q[X]/\left(X^N + 1\right)$ *chosen according to* $\mathcal{X}_{\mathsf{sk}}$*, the* RLWE *distribution samples* $a(x) \in \mathbb{Z}_q[X]/\left(X^N + 1\right)$ *uniformly at random, samples* $e(x) \in \mathbb{Z}_q[X]/\left(X^N + 1\right)$ *from* $\mathcal{X}_e$*, computes* $b(x) := a(x) \cdot \mathsf{sk}(x) + e(x) \mod q$ *and outputs* $(a(x), b(x))$*.*

Definition 15 (Search RLWE problem)**.** *Let* $\mathbf{a}, \mathbf{s} \in \mathcal{R}_q$*, and* $\mathbf{e} \xleftarrow{\$} \chi$*. Set* $\mathbf{b} \leftarrow \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e} \mod q$*. Given* $(\mathbf{a}, \mathbf{b})$*, find* $\mathbf{s}$*.*

Definition 16 (Decision RLWE problem)**.** *Given the pair* $(\mathbf{a}, \mathbf{b})$*, where* $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q$*, distinguishing the following cases for* $\mathbf{b}$

- $\mathbf{b}$ *is chosen uniformly at random from* $\mathcal{R}_q$*.*
- $\mathbf{b} \leftarrow \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e} \mod q$*, where* $\mathbf{s} \xleftarrow{\$} \mathcal{R}_q$ *and* $\mathbf{e} \xleftarrow{\$} \chi$*.*

The Search-RLWE and Decisional-RLWE problems are proven hard as standard hard lattice problems with properly chosen parameters, as shown in [215]

Definition 17 (RLWE-based Encryption)**.** *An RLWE-based encryption scheme for message space* $\mathcal{M} = \mathcal{R}_t$*, consists of three algorithms* RLWE.Setup, RLWE.Enc, RLWE.Dec *defined as follows:*

- RLWE.Setup $\left(1^\lambda, \mathsf{pp}\right)$*. On the input of a security parameter* λ*, outputs a secret key* sk *and a public key* pk*, where* $\mathsf{sk} \leftarrow_{\$} \mathbb{Z}_2[X]/(X^N + 1)$ *and* $\mathsf{pk} = (b, a)$ *for* $a \leftarrow_{\$} \mathcal{R}_q$ *and* $b = [-a \cdot \mathsf{sk} + t \cdot e_0]_q$*, with* $e_0 \leftarrow \mathcal{D}_{q,\sigma}$*;*

- RLWE.Enc (pk, x) . *On the input of a public key* pk *and a message x in the message space, samples* $u \leftarrow_{\$} \mathbb{Z}_2[X]/(X^N+1)$, $e_1, e_2 \leftarrow \mathcal{D}_{q,\sigma}$ *and outputs a ciphertext* $\mathbf{c} = (\mathsf{c}_1, \mathsf{c}_2)$*, where* $\mathsf{c}_1 = b \cdot u + t \cdot e_1 + x$ *and* $\mathsf{c}_2 = a \cdot u + t \cdot e_2$*;*
- RLWE.Dec $(\mathsf{sk}, \mathbf{c})$. *On the input of a decryption key* sk *and a ciphertext* $\mathbf{c} = (\mathsf{c}_1, \mathsf{c}_2)$*, computes* $x^* \leftarrow \mathsf{c}_1 + \mathsf{sk} \cdot \mathsf{c}_2$*, and outputs the message* $x \in \mathcal{M}$ *obtained by rounding from* x^**.*

6.2.3 Module-Learning With Errors (MLWE)

The Module-LWE problem generalizes the Ring-LWE problem by considering vectors of ring elements. Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, where N is typically a power of two. In the Module-LWE setting, the public samples are drawn from $\mathcal{R}_q^\kappa$, where κ denotes the rank of the module.

In contrast to RLWE (which corresponds to $\kappa = 1$), Module-LWE allows tuning the hardness of the problem by adjusting the parameters N, κ, and q. In practice, implementations typically fix N and q to enable optimized arithmetic in $\mathcal{R}_q$, and increase κ to scale the security level.

Definition 18 (Search Module-LWE)**.** *Let* $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$*. A Module-LWE sample is a pair* $(\mathbf{a}, b) \in \mathcal{R}_q^\kappa \times \mathcal{R}_q$*, where* $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q^\kappa$ *is uniformly random, and*

$$b = \langle \mathbf{a}, \mathbf{s} \rangle + e,$$

with secret $\mathbf{s} \in \mathcal{R}_q^\kappa$ *sampled from a distribution* $\mathcal{X}_{\mathrm{sk}}$*, and error* $e \leftarrow \mathcal{X}_e$ *drawn from a small noise distribution over* $\mathcal{R}_q$*.*

The Search Module-LWE problem consists in recovering $\mathbf{s}$ *given access to* m *independent samples*

$$\{(\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=0}^{m-1}.$$

Notation. We denote an MLWE instance by $\mathrm{MLWE}(N, \kappa, m, q)$, where N is the ring dimension, κ the module rank, m the number of samples, and q the modulus.

Elements of $\mathcal{R}_q^\kappa$ are vectors of κ polynomials in $\mathcal{R}_q$. Addition and subtraction are performed component-wise, while the inner product $\langle \mathbf{a}, \mathbf{s} \rangle$ denotes the sum of component-wise products, yielding an element in $\mathcal{R}_q$.

6.2.4 Learning with errors over the torus (TLWE)

Though in the past decades, LWE or RLWE over integers have been in the spotlight recent workshave been extending this paradigm to another structure: the torus. We define the real torus as the ring $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, that is, the set of real numbers between 0 and 1. We justify using this new structure in Sect. 6.3, where we describe TFHE, a cryptosystem that relies on the following distribution.

Set $\mathcal{R} = \mathbb{T}$. A TLWE sample is defined as:

$$\mathtt{TLWE}_{\mathbf{s}}(\mu) = (\mathbf{a}, \mu + \langle \mathbf{s}, \mathbf{a} \rangle + e),$$

$\mathbf{a} \in \mathbb{T}_N[X]^k$, $\mathbf{s} \in \mathbb{Z}_N[X]^k$, $\mu \in \mathbb{T}_N[X]$ and $e \xleftarrow{\chi} \mathbb{T}_N[X]$. In classical LWE, one can define the classical LWE encryption over the torus by setting $N = 1$ or a ring version of TLWE by setting $k = 1$. These are the two types of ciphertexts used in torus-based cryptography.

6.2.5 General Gentry-Sahai-Waters construction (GGSW)

A GGSW encryption, named after Gentry *et al.* who first introduced it, is an extension of GLWE. In a nutshell, it is given by a vector of GLWE distributions. Consider an integer plaintext $\mu \in \mathbb{Z}_N[X]^k$ and a vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N[X]^k$. For $d > 0$ its *depth* and a basis $\beta > 0$, a GGSW sample is given by

$$\mathtt{GGLW}_{\mathbf{s}}(\mu) = \mathbf{Z} + \mu \cdot \mathbf{G}^{(\beta)} \in \mathbb{T}_N[X]^{d(k+1)\times(k+1)}$$

where $\mathbf{Z}$ is a matrix of $d(k+1)$ GLWE distributions and $\mathbf{G}^{(\beta)}$ is the gadget matrix in $\mathbb{T}_N[X]^{d(k+1)\times(k+1)}$ given by $G^{(\beta)}_{i,j} = \mathbf{I}_{k+1} \otimes \mathbf{g}$ for $\mathbf{I}_{k+1}$ the identity matrix, $\otimes$ the tensor product and $\mathbf{g}$ the vector of size d given by $g_i = \beta^{-i}, i \in [d]$.

We introduce a last version of LWE over the torus. We detail the construction of TFHE in section 6.3.

Consider the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ and $n, m > 0$ two integers. Let ξ be the error distribution over $\mathbb{T}$. Torus-LWE is defined as follows:

Definition 19 (Search TLWE problem)**.** *Let $\mathbf{s} \in \mathbb{Z}^n$ be a secret vector. Given m independent samples $(a_i, \langle a_i, \mathbf{s}\rangle + e_i)$ where $a_i \xleftarrow{\$} \mathbb{T}^n$ and $e_i \xleftarrow{\$} \xi$, find $\mathbf{s}$.*

Definition 20 (Decisional TLWE problem)**.** *Let $\mathbf{s} \in \mathbb{Z}^n$ be a secret vector. Given the pair $(A, \mathbf{b})$, where $A \in \mathbb{T}^{(m\times n)}, \mathbf{b} \in \mathbb{T}^m$; distinguishing the following cases for $\mathbf{b}$:*

- *$\mathbf{b}$ is chosen uniformly at random from $\mathbb{T}^m$.*
- *$\mathbf{b} \leftarrow A \cdot \mathbf{s} + \mathbf{e}$ where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}^n$ and $\mathbf{e} \xleftarrow{\$} \xi^m$.*

6.2.6 LWE Security and Resistance to Attacks

The security of LWE, RLWE, and MLWE is fundamentally rooted in their formal worst-case to average-case reductions. Regev [274] demonstrated that solving LWE on average is at least as hard as quantumly solving worst-case standard lattice problems (such as GapSVP and SIVP). This was later extended to the structured variants: Lyubashevsky, Peikert, and Regev [216] provided a reduction from worst-case problems on ideal lattices to average-case RLWE, and Langlois and Stehlé [200] established analogous reductions from module lattice problems to MLWE. These reductions guarantee that cracking a random instance of these structured and unstructured LWE problems is theoretically as difficult as solving the hardest instances of underlying lattice problems.

In practice, evaluating the concrete security of LWE-based schemes involves analysing their resistance to the most effective cryptanalytic techniques, predominantly lattice reduction algorithms like the Block Korkine-Zolotarev (BKZ) algorithm. The core subroutine of BKZ relies on solving the Shortest Vector Problem (SVP) in smaller projected sublattices, which is typically achieved through either enumeration or sieving approaches. Recent advancements in sieving, such as the Becker-Gama-Joux (BGJ) sieve, have highlighted the need to rigorously account for memory constraints and hardware optimisation when determining the practical complexity of real-world SVP solvers.

Notably, a recent breakthrough by Zhao, Ding, and Yang [323] introduced a highly optimised approach termed “Sieving with streaming memory access.” They significantly improved the memory behaviour of the BGJ sieve by minimizing RAM access overheads, demonstrating a roughly 40% reduction in RAM footprint and remarkable accelerations on modern CPUs. Their work refined the concrete cost estimation for solving SVP, revealing that parameters with excessively narrow security margins might provide less concrete security than initially estimated against heavily optimised, memory-aware sieving implementations.

Despite these advanced optimisations in sieving, cryptographic primitives based on RLWE and MLWE remain robustly secure against such attacks. The reasons for their resistance are multifold:

1. **Exponential Asymptotic Hardness:** The optimisations presented by Zhao et al. [323] primarily improve the constant factors and hardware-level memory efficiency of the sieve in practical limits. However, the underlying time and space complexities of sieving for SVP remain strictly exponential with respect to the lattice dimension even with idealised hardware. The conservative parameter choices in standardised RLWE and MLWE schemes ensure that the required lattice dimensions are large enough to keep the exponential cost entirely computationally prohibitive.

2. **Absence of Devastating Algebraic Attacks:** While RLWE and MLWE rely on structured lattices (ideal and module lattices), these structures have not yet yielded practical algorithmic advantages for sieving algorithms over general random lattices in the parameters used for cryptography. The improved BGJ sieve operates predominantly by viewing the lattice as an unstructured entity; thus, it does not exploit the underlying ring or module structure to achieve sub-exponential or polynomial-time breaches.

3. **Generous Security Margins:** Modern standards leveraging MLWE and RLWE are parameterised with substantial security buffers specifically intended to absorb such gradual, continuous improvements in classical and quantum lattice algorithms. The theoretical enhancements in practical cost scaling by Zhao et al., while significant for borderline cases or unstructured instances aiming for minimal sizes, fail to bridge the massive gap required to threaten the securely buffered design of contemporary MLWE and RLWE formulations.

Consequently, while streaming memory access optimisations provide critical insights into the real-world cost of lattice reduction bounds, the structural properties and conservative parameterisation of RLWE and MLWE ensure they safely withstand these and similar generalised sieving attacks.

6.3 Homomorphic Encryption

Over time, research has established three main types of Homomorphic Encryption (HE): Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption (FHE). These variants differ in functionality and computational cost. FHE enables arbitrary computations on ciphertexts, supporting both addition and multiplication as well as more complex operations. However, it remains computationally demanding and resource-intensive. PHE, by contrast, supports only one operation–either addition or multiplication–making it more efficient but considerably less flexible. SHE provides a compromise between these two extremes by permitting a limited number of additions and multiplications while maintaining more practical performance than FHE and greater expressiveness than PHE.

Following Gentry's groundbreaking work [137], numerous HE schemes have been developed to support diverse data types. Among the schemes endorsed by the Homomorphic Encryption Standard [17], the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [68] performs modular arithmetic over finite fields, enabling computations on integer vectors in $\mathbb{Z}_q$ for $q \geq 2$. Similarly, the Brakerski/Fan-Vercauteren (B/FV) scheme [67, 128] operates over finite fields and supports computations on integer vectors. The Cheon-Kim-Kim-Song (CKKS) scheme [80] extends these principles to approximate arithmetic on vectors of real and complex numbers. The Torus FHE (TFHE) scheme [81, 82, 83] adopts a different model, using boolean circuits and decision diagrams to operate on low-precision integers [84].

Each HE scheme is optimized for specific performance criteria. BGV, B/FV, and CKKS aim to minimize the multiplicative depth of their decryption circuits, whereas TFHE prioritizes reducing the number of

decryption gates. We now provide a generic definition of HE to help the reader follow the rest of this section.

An $\texttt{HE} := (\texttt{KeyGen}, \texttt{Enc}, \texttt{Eval}, \texttt{Dec})$ scheme is a tuple of four algorithms defined as follows:

1. $\texttt{KeyGen}\left(1^\lambda\right)$ given a security parameter λ and outputs the public key $\texttt{pk}$, the secret key $\texttt{sk}$ and the evaluation key $\texttt{evk}$;
2. $\texttt{Enc}\left(\texttt{pk}, \mathbf{m}\right)$ given the public key $\texttt{pk}$ and a message $\mathbf{m}$ and outputs a ciphertext $\mathbf{c_m}$;
3. $\texttt{Eval}\left(\texttt{evk}, f, (c_{m_1}, \ldots, c_{m_n})\right)$ given the evaluation key $\texttt{evk}$, a function f and an n-tuple of ciphertexts $(c_{m_1}, \ldots, c_{m_n})$ and outputs a ciphertext $\mathbf{c}^f$;
4. $\texttt{Dec}\left(\texttt{sk}, \mathbf{c}^f\right)$ given the secret key $\texttt{sk}$ and a ciphertext $\mathbf{c}^f$ and outputs $\mathbf{m}^f$.

Correctness: An HE scheme is correct if:

$$\Pr\Big[\texttt{Dec}(\texttt{sk}, \mathbf{c}^f) \neq f(m_1, \ldots, m_n) \;\Big|\; [(\texttt{pk}, \texttt{sk}, \texttt{evk}) \leftarrow \texttt{KeyGen}(1^\lambda)] \\ \wedge\; [\mathbf{c}^f \leftarrow \texttt{Eval}(\texttt{evk}, f, (c_{m_1}, \ldots, c_{m_n}))]\Big] = negl(\lambda).$$

6.3.1 Brakerski-Gentry-Vaikuntanathan (BGV)

BGV is an SHE scheme based on RLWE that supports modular arithmetic over finite fields [68]. BGV construction is scale-dependent, relying on a predetermined sequence of moduli $\mathcal{Q} = \{q_0, q_1, \ldots, q_L\}$. Each modulus corresponds to the ciphertext scale for a particular operation, and each multiplication requires a modulus switch. BGV is a scale-dependent scheme that defines multiple ciphertext modulo $\mathcal{Q} = \{p_0, p_1, \ldots, p_L\}$, for each level ciphertext's modulus is q_l, where $0 \leq l \leq L$. While performing HE computation, the ciphertext moves down from the freshly encrypted ciphertext at level L to level 1, the level immediately before decryption.

Setup: Given a security parameter λ, generate two integers $t, n > 0$ and $q_0 < \ldots < q_L$ a sequence of powers of prime q_ℓ with $\ell \in [L]$. Then set $\mathcal{P} = \mathcal{R}_t = \mathbb{Z}_t[X]/(X^n + 1)$ the plaintext space and $\mathcal{C} = \mathcal{R}_{q_l} \times \mathcal{R}_{q_l}$ the ciphertext space at level ℓ, for $\mathcal{R}_{q_l} = \mathbb{Z}_{q_l}[X]/(X^n + 1)$. Finally, denote $\mathcal{D}_{q_l,\sigma}$ the discrete Gaussian distribution over $\mathcal{R}_{q_l}$ of width σ. The BGV algorithms are as follows:

1. $\texttt{KeyGen}\left(1^\lambda\right)$ takes as input a security parameter λ. Sample $\texttt{sk} \xleftarrow{\$} \mathcal{R}$ with coefficients in $\{-1, 0, 1\}$, $a \xleftarrow{\$} \mathcal{R}_{q_L}$, and $e \leftarrow \mathcal{D}_{q_L,\sigma}$; set $\texttt{pk} = (b, a) \in \mathcal{R}^2_{q_L}$, where $b \leftarrow [-a \cdot \texttt{sk} + t \cdot e]_{q_L}$, and $\texttt{evk} = (b', a) \in \mathcal{R}^2_{q_L}$, where $b' \leftarrow [-(a \cdot \texttt{sk} + e) + \texttt{sk}^2]_{q_l}$; outputs $(\texttt{sk}, \texttt{pk}, \texttt{evk})$;

2. $\texttt{Enc}(\texttt{pk}, m)$ takes as input the public key $\texttt{pk}$ and a plaintext $m \in \mathcal{P}$. Sample $u \xleftarrow{\$} \mathcal{R}$ with coefficients in $\{-1, 0, 1\}$, and $e_1, e_2 \leftarrow \mathcal{D}_{q_\ell,\sigma}$. Set $c_1 \leftarrow [b \cdot u + t \cdot e_1 + m]_{q_\ell}$ and $c_2 \leftarrow [a \cdot u + t \cdot e_2]_{q_\ell}$; outputs $c = (c_1, c_2)$;

3. $\texttt{EvalAdd}\left(c^{(1)}, c^{(2)}\right)$ takes as input two ciphertexts $c^{(1)}, c^{(2)}$ and computes $c_1^{\texttt{add}} \leftarrow \left[c_1^{(1)} + c_1^{(2)}\right]_{q_\ell}$ and $c_2^{\texttt{add}} \leftarrow \left[c_2^{(1)} + c_2^{(2)}\right]_{q_\ell}$; outputs $c^{\texttt{add}} = (c_1^{\texttt{add}}, c_2^{\texttt{add}})$;

4. $\texttt{EvalMult}\left(c^{(1)}, c^{(2)}\right)$ takes as input two ciphertexts $c^{(1)}, c^{(2)}$ and computes $\overline{c} = (\overline{c_1}, \overline{c_2}, \overline{c_3}) \leftarrow \Big(\left[c_1^{(1)} \cdot c_1^{(2)}\right]_{q_\ell}, \left[c_1^{(1)} \cdot c_2^{(2)} + c_2^{(1)} \cdot c_1^{(2)}\right]_{q_\ell}, \left[c_2^{(1)} \cdot c_2^{(2)}\right]_{q_\ell}\Big)$. [1] Set $c^{\texttt{mult}} \leftarrow \texttt{Relinearize}\left(\texttt{evk}, \overline{c}\right)$; outputs $c^{\texttt{mult}}$;

[1] When we multiply two ciphertexts, the resulting ciphertext has one more ring element than the original ones, and its secret key is squared $(\texttt{sk}^2)$, which makes it challenging to decrypt. Thus, we need to perform Re-linearization to reduce the number of ring elements back to two and make the final result decryptable by $\texttt{sk}$. Also, the noise in c^f grows multiplicatively from the multiplication of noises in the input ciphertexts $(v = e \cdot e)$;

5. $\texttt{Relinearize}(\texttt{evk}, \overline{c})$ takes as input the evaluation key $\texttt{evk} = (b', a)$ and $\overline{c} = (\overline{c_1}, \overline{c_2}, \overline{c_3})$, compute $c_1 \leftarrow [\overline{c_1} + b' \cdot \overline{c_3}]_{q_l}$ and $c_2 \leftarrow [\overline{c_2} + a \cdot \overline{c_3}]_{q_l}$; outputs $c = (c_1, c_2)$;

6. $\texttt{ModSwitch}(\texttt{evk}, c)$ takes as input the evaluation key $\texttt{evk}$, a ciphertext c encrypted modulo q_ℓ; computes $c' \leftarrow \left\lfloor \frac{q_\ell}{q_{\ell-1}} \cdot c \right\rceil$; outputs c';

7. $\texttt{Dec}(\texttt{sk}, c)$ takes as input the secret key $\texttt{sk}$ and the ciphertext c. Compute $m' \leftarrow [c_1 + c_2 \cdot \texttt{sk}]_{q_\ell}$.

For simplicity we will not redefine functions for other HE schemes sharing the same principles as BGV. Instead, we will simply refer to these functions by name, highlighting the commonality in their inputs and outputs.

6.3.2 Brakerski/Fan-Vercauteren (B/FV)

B/FV is another SHE scheme that supports modular arithmetic over finite fields [67, 128]. B/FV is scale-independent, meaning the ciphertext modulus remains constant during homomorphic evaluation. The plaintext and the ciphertext spaces are defined over two polynomial rings denoted by $\mathcal{P} = \mathcal{R}_t = \mathbb{Z}_t[X]/(X^n + 1)$, and $\mathcal{C} = \mathcal{R}_q \times \mathcal{R}_q$, where $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$, respectively. Unlike the BGV scheme, in B/FV, the plaintext is placed on the most significant bits of the data structure. This is achieved by utilizing a scale factor, denoted as Δ, and adjusting the message's scale before encryption and after decryption. The B/FV is outlined as follows:

1. $\texttt{ScaleUp}\left(m, \Delta = \left\lfloor \frac{q}{t} \right\rfloor\right)$ takes as input the message m and scale factor Δ and outputs $m' = \Delta \cdot m$;

2. $\texttt{ScaleDown}\left(m', \Delta^{-1} = \left\lfloor \frac{t}{q} \right\rfloor\right)$ takes the scaled-up message m' and reverse scale factor Δ^{-1}, and outputs $m = \Delta^{-1} \cdot m'$.

3. $\texttt{KeyGen}\left(1^\lambda\right)$ outputs $(\texttt{sk}, \texttt{pk}, \texttt{evk})$;

4. $\texttt{Enc}\left(\texttt{pk}, m'\right)$ outputs c;

5. $\texttt{EvalAdd}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{add}}$;

6. $\texttt{EvalMult}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{mult}}$;

7. $\texttt{Relinearize}\left(\texttt{evk}, \overline{c}\right)$ outputs c;

8. $\texttt{Dec}\left(\texttt{sk}, c\right)$ outputs m'.

6.3.3 Cheon-Kim-Kim-Song (CKKS)

CKKS is an SHE scheme permitting approximate computation on vectors of real and complex numbers [115]. CKKS operates as a scale-dependent HE scheme, where the ciphertext modulus adapts throughout homomorphic evaluation, akin to the BGV scheme. The plaintext and ciphertext spaces are defined in the same way as $\mathcal{P} = \mathcal{R}$ and $\mathcal{C} = \mathcal{R}_q^2$, where $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$.

6.3.3.1 Encoding

We consider a message as a vector of complex numbers $\mathbf{z} \in \mathbb{C}^{n/2}$ and provide an encoding to convert $\mathbf{z}$ into a suitable plaintext. This method relies on a scaling factor Δ and the canonical embedding $\pi : \mathbb{R}^n \to \mathbb{C}^{n/2}$. This construction is detailed in [80].

1. $\texttt{Encode}\,(\mathbf{z}, \Delta)$ takes as input a vector $\mathbf{z} \in \mathbb{C}^n$ and the scale factor Δ. Maps $\mathbf{z}$ into element $w \in \mathcal{R}$, where $w \leftarrow \lfloor \Delta \cdot \pi^{-1}(\mathbf{z}) \rceil$; outputs w;
2. $\texttt{Decode}\,(w, \Delta)$ takes a plaintext ring element $w \in \mathcal{R}$, and computes $\mathbf{z} \leftarrow \pi\left(\Delta^{-1} \cdot w\right)$; outputs $\mathbf{z}$;
3. $\texttt{KeyGen}\left(1^\lambda\right)$ outputs $(\texttt{sk}, \texttt{pk}, \texttt{evk})$;
4. $\texttt{Enc}\,(\texttt{pk}, m)$ outputs c;
5. $\texttt{EvalAdd}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{add}}$;
6. $\texttt{EvalMult}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{mult}}$;
7. $\texttt{Relinearize}\,(\texttt{evk}, \overline{c})$ outputs c;
8. $\texttt{Rescale}\,(c_{q_\ell}, \Delta)$ scale the input ciphertext c down by Δ such that $c'_{q_{\ell-1}} = \Delta^{-1} \cdot [(c_1, c_2)]_{q_\ell}$. `Rescale` is similar to BGV `ModSwitch`, outputs $c'_{q_{\ell-1}}$;
9. $\texttt{Dec}\,(\texttt{sk}, c)$ outputs m'.

6.3.4 Torus FHE (TFHE)

TFHE is an FHE scheme founded on bootstrapping. Unlike the *leveled* HE schemes, TFHE is *fully* homomorphic, enabling support for any logical circuit and, consequently, any function. The particularity of TFHE is that it uses two types of encryption: the classical LWE ciphertext and the so-called General-GSW ciphertext. The latter permits efficient homomorphic multiplications.

6.3.4.1 Bootstrapping

To define a proper cryptosystem, one has to guarantee that any ciphertext can be decrypted correctly. As homomorphic operations tend to increase noise, it is necessary to provide a method to keep the noise of a ciphertext under a given bound. This can be achieved with a technique known as bootstrapping, which consists of refreshing the encryption of a message into a new ciphertext, hence resetting the noise to an acceptable level.

Let $\mathbb{B} = \{-1, 0, 1\}$, and $\mathbb{T}$ be the real Torus, *i.e.* the set of real numbers modulo one. For $q, N > 0$ two powers of 2, we denote $\mathbb{T}_N[X] = \mathbb{T}_N[X]/(X^N + 1)$, $\mathbb{B}_N[X] = \mathbb{B}[X]/(X^N + 1)$ and $\mathbb{Z}_{q,N}[X] = \mathbb{Z}_q[X]/(X^N + 1)$. Let $\beta > 0$ and $d \in \mathbb{Z}$ be two integers, called the base and the depth, respectively.

1. $\texttt{KeyGen}\left(1^\lambda\right)$ takes as input a security parameter λ. Sample $k = k(\lambda)$ polynomials $\texttt{sk} := (s_1, \ldots, s_k) \xleftarrow{\$} (\mathbb{B}_{q,N}[X])^k$. Set $\texttt{pk} := (\mathbf{a}, b) \leftarrow \texttt{GLWE}_{\texttt{sk}}(\mathbf{0})$, output the keys $(\texttt{pk}, \texttt{sk})$.
2. $\texttt{Enc}(\texttt{pk}, m)$ takes as input the public key $\texttt{pk}$ and a message $m \in \mathbb{Z}_{q,N}[X]$. Compute $\mathbf{c} \leftarrow (\mathbf{a}, b+m+e) = \texttt{GLWE}_{\texttt{sk}}(m) \in (\mathbb{Z}_{q,N}[X])^{k+1}$, for $e \xleftarrow{\chi} \mathbb{Z}_{q,N}[X]$ and outputs $\mathbf{c}$;

3. $\texttt{EvalMult}(\mathbf{G}, \mathbf{c})$ takes as input a GGSW ciphertext $\mathbf{G}$ and a GLWE ciphertext $\mathbf{c}$. First compute $(\mathbf{c})_\beta \leftarrow ((a_1)_\beta, \ldots, (a_k)_\beta, (b)_\beta) \in \mathbb{Z}_{q,N}[X]^{d(k+1)}$ where $(\cdot)_\beta$ is the decomposition in basis β. Compute the GLWE ciphertext $\mathbf{c}^{\texttt{mult}} \leftarrow (\mathbf{c})_\beta \cdot \mathbf{G}$; outputs $\mathbf{c}^{\texttt{mult}}$;

4. $\texttt{KeySwitch}(\texttt{sk}', \mathbf{c})$ takes as input a new secret key $\texttt{sk}'$ and a GLWE ciphertext $\mathbf{c} = (\mathbf{a}, b)$. Define the key-switching key $\mathbf{K} \in \mathbb{Z}_{q,N}[X]^{d \cdot k \times (k+1)}$ as the first $d \cdot k$ rows of $\texttt{GGSW}_{\texttt{sk}'}(1)$. Compute the GLWE ciphertext $\mathbf{c}' \leftarrow (\mathbf{0}, b) - \texttt{EvalMult}(\mathbf{K}, \mathbf{a})$ encrypted under $\texttt{sk}'$; outputs $\mathbf{c}'$;

5. $\texttt{CMux}(\mathbf{B}, \mathbf{c_0}, \mathbf{c_1})$ takes as input two GLWE ciphertexts $\mathbf{c_0}, \mathbf{c_1}$ of plaintexts m_0, m_1 and a GGSW ciphertext $\mathbf{B}$ of a bit b. Compute the GLWE ciphertext $\mathbf{c_3} \leftarrow \texttt{EvalMult}(\mathbf{B}, \mathbf{c_1} - \mathbf{c_0}) + \mathbf{c_0}$, which is an encryption of m_b with fresh noise; outputs $\mathbf{c_3}$;

6. $\texttt{Dec}(\texttt{sk}, \mathbf{c})$ takes as input the secret key $\texttt{sk}$ and a ciphertext $\mathbf{c}$. Compute $b - \sum_{i=1}^{n} a_i \cdot s_i = m + e$. Then, round to the nearest integer to output the message m.

Mark that a GLWE ciphertext mentioned earlier is an LWE ciphertext if $N = 1$ and an RLWE ciphertext if $k = 1$.

This method is usually too slow and, thus, impractical; TFHE introduces a novel idea that consists of running the bootstrapping together with the evaluation of a look-up table. Given as input $\mathbf{G}$, a GGSW encryption of an NLUT L, and $\mathbf{c}$, the LWE encryption of a message m, one can compute a fresh ciphertext $\mathbf{c}'$ of $L[m]$ directly. This method requires a heavy usage of the $\texttt{CMux}$ algorithm that can be optimized using *programmable bootstrapping*. We refer the reader to the articles of Chillotti *et al.* [83] and Cosseron *et al.* [92]

6.4 Functional Encryption (FE)

Like previously stated in Section 4.5.2, FE allows a user to compute specific functions on encrypted data without revealing the underlying plaintext and only obtains the result of that specific function. We now provide formal definitions for FE, Multi-Client Functional Encryption (MCFE) and ciphertext updatability as well as their correctness.

Definition 21 (Functional Encryption (FE))**.** *A functional encryption scheme* FE *for message space* $\mathcal{M}$ *and class of functions* $\mathcal{F} \subseteq \{f : \mathcal{M} \to \mathcal{Y}\}$*, consists of four PT algorithms* Setup, KDer, Enc, Dec *defined as follows:*

- $\mathsf{Setup}\left(1^\lambda\right)$*. The setup algorithm is a probabilistic algorithm. On the input of a security parameter* λ*, it outputs a secret key* sk *and a public key* pk*;*
- $\mathsf{Enc}\left(\mathsf{pk}, x\right)$*. The encryption algorithm is a probabilistic algorithm. On the input of the public key* pk *and a message* x *in the message space* $\mathcal{M}$*, it outputs a ciphertext* c*.*
- $\mathsf{KDer}\left(\mathsf{sk}, f\right)$*. The key derivation algorithm can be deterministic or probabilistic. On the input of the secret key* sk *and a function* $f \in \mathcal{F}$*, it outputs a decryption key* dk_f*.*
- $\mathsf{Dec}\left(\mathsf{dk}_f, c\right)$*. The decryption algorithm is a deterministic algorithm. On the input of a decryption key* dk_f *and a ciphertext* c*, it outputs an element* $y \in \mathcal{Y}$*.*

Correctness: *For all parameters* $\lambda \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{Setup}\left(1^{\lambda}\right)$, $f \in \mathcal{F}$ *and* $x \in \mathcal{M}$,

$$\mathbb{P}\left[\mathsf{Dec}\left(\mathsf{dk}_f, \mathsf{Enc}\left(\mathsf{pk}, x\right)\right) \neq f\left(x\right) \parallel \mathsf{dk}_f \leftarrow\!\!\$\ \mathsf{KDer}\left(\mathsf{sk}, f\right)\right]$$

is negligible in λ.

A functional encryption scheme only supports single-input functions, operating over $\mathcal{M}$. In many scenarios, it is necessary to extend the set of functions $\mathcal{F}$ to multi-inputs, over vectors of plaintexts in $\mathcal{M}^m$. Additionally, as previously mentioned, supporting multiple clients (MCFE) requires specific encryption and a key derivation algorithm taking an additional label α (that we can see as an identifier) as input.

Definition 22 (Multi-Client Functional Encryption (MCFE))**.** *An* m*-multi-client functional encryption scheme* MCFE *for message space* $\mathcal{M}$ *and class of functions* $\mathcal{F} \subseteq \{f : \mathcal{M}^m \to \mathcal{Y}\}$*, consists of four PT algorithms* Setup, Enc, KDer, Dec *defined as follows:*

- $\mathsf{Setup}\left(1^{\lambda}, m\right)$*. The setup algorithm is a probabilistic algorithm. On the input of a security parameter* λ *and an integer* $m \in \mathbb{N}$*, it outputs a secret key* sk *and a public key* $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_m)$.
- $\mathsf{Enc}\left(\mathsf{pk}_i, x_i, \alpha\right)$*. The encryption algorithm is a probabilistic algorithm. On the input of the* i*-th public key component* pk_i*, a message* $x_i \in \mathcal{M}$ *and a label* α*, it outputs a ciphertext* c_i.
- $\mathsf{KDer}\left(\mathsf{sk}, f, \alpha\right)$*. The key derivation algorithm can be deterministic or probabilistic. On the input of the secret key* sk*, a function* $f \in \mathcal{F}$ *and a label* α*, it outputs a decryption key* $\mathsf{dk}_{\alpha,f}$.
- $\mathsf{Dec}\left(\mathsf{dk}_{\alpha,f}, \mathbf{c}\right)$*. The decryption algorithm is a deterministic algorithm. On the input of a decryption key* $\mathsf{dk}_{\alpha,f}$ *and a ciphertext vector* $\mathbf{c} = (c_1, \ldots, c_m)$*, it outputs an element* $y \in \mathcal{Y}$.

Correctness: *For all parameters* $\lambda, m \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{Setup}\left(1^{\lambda}, m\right)$, $f \in \mathcal{F}$*, labels* $\alpha, \alpha_1, \ldots, \alpha_m$ *and message vector* $\mathbf{x} = (x_1, \ldots, x_m) \in \mathcal{M}^m$,

$$\mathbb{P}\left[\mathsf{Dec}\left(\mathsf{dk}_{\alpha,f}, \mathbf{c}\right) \neq f\left(\mathbf{x}\right) \parallel \begin{array}{l}\mathsf{dk}_{\alpha,f} \leftarrow\!\!\$\ \mathsf{KDer}\left(\mathsf{sk}, f, \alpha\right) \\ c_i \leftarrow\!\!\$\ \mathsf{Enc}\left(\mathsf{pk}_i, x_i, \alpha_i\right)\end{array}\right] \qquad (6.1)$$

is negligible if $\alpha = \alpha_1 = \cdots = \alpha_m$ *and overwhelming otherwise.*

Remark: For the sake of brevity and readability, we sometimes use the notation $\mathbf{c} \leftarrow\!\!\$\ \mathsf{Enc}\left(\mathsf{pk}, \mathbf{x}, \alpha\right)$ to denote the encryption of the full vector $\mathbf{x} = (x_1, \ldots, x_m)$ into a vector of ciphertexts $\mathbf{c} = (c_1, \ldots, c_m)$ for keys $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_m)$ and common label α. This notation is valid, since it corresponds to single-input FE for message space $\mathcal{M}^m \times \mathcal{X}$, where $\mathcal{X}$ would denote the space in which α lies.

Ciphertext Updatability is a feature allowing better management of the ciphertexts encrypted using labels δ or tags. In short, it permits us to update the label under which a certain ciphertext was encrypted. To perform this update, an authority can generate a token $\mathsf{tk}_{\delta \to \delta'}$ that can update any ciphertext encrypted with a label δ into a ciphertext encrypted with the label δ'.

Definition 23 (Ciphertext Updatability (CU))**.** *Let* MCFE *be an MCFE scheme for message space* $\mathcal{M}$ *and class of functions* $\mathcal{F} \subseteq \{f : \mathcal{M}^m \to \mathcal{Y}\}$*. Let* $(\mathsf{TokGen}, \mathsf{Upd})$ *be a pair of algorithms defined as follows:*

- $\mathsf{TokGen}\left(\mathsf{sk}, \delta, \delta'\right)$*. The token generation algorithm can be deterministic or probabilistic. On the input of the secret key* sk *and two labels* δ, δ'*, it outputs a token* $\mathsf{tk}_{\delta \to \delta'}$.
- $\mathsf{Upd}\left(\mathsf{tk}_{\delta \to \delta'}, \mathbf{c}_{\delta}\right)$*. The update algorithm is a probabilistic algorithm. On the input of a token* $\mathsf{tk}_{\delta \to \delta'}$ *and a ciphertext vector* $\mathbf{c}_{\delta}$*, it outputs a ciphertext* $\mathbf{c}_{\delta'}$.

Correctness: *For all parameters* $\lambda, m \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{Setup}\left(1^\lambda, m\right)$, $f \in \mathcal{F}$, *labels* δ, δ', *message vector* $\mathbf{x} = (x_1, \ldots, x_m) \in \mathcal{M}^m$, *ciphertext vector* $\mathbf{c}_\delta$ *where* $c_{i,\delta} \leftarrow\!\!\$\ \mathsf{Enc}\left(\mathsf{pk}_i, x_i, \delta\right)$ *and* $\mathsf{dk}_{\delta^*,f} \leftarrow\!\!\$\ \mathsf{KDer}\left(\mathsf{sk}, f, \delta^*\right)$, $\delta^* \in \{\delta, \delta'\}$,
$\mathsf{tk}_{\delta\to\delta'} \leftarrow \mathsf{TokGen}\left(\mathsf{sk}, \delta, \delta'\right)$ *and* $\mathbf{c}_{\delta'} \leftarrow\!\!\$\ \mathsf{Upd}\left(\mathsf{tk}_{\delta\to\delta'}, \mathbf{c}_\delta\right)$ *the following equality holds:*

$$\mathsf{Dec}\left(\mathsf{dk}_{\delta',f}, \mathbf{c}_{\delta'}\right) = \mathsf{Dec}\left(\mathsf{dk}_{\delta,f}, \mathbf{c}_\delta\right)$$

with overwhelming probability.

6.5 Functional Encryption scheme - ACE of SPADE (AoS)

We now describe the formal construction of AoS, a new lattice-based, construction that is based on RLWE and is quantum secure. This scheme introduces partial decryption in a dynamic multi-client setting, enabling flexible user registration and fine-grained access control. In addition, AoS supports the important feature of ciphertext updatability and extends FE towards qualitative data analysis. In Sect. 6.5.1 we provide the core algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KDer}, \mathsf{Dec})$ constituting the MCFE scheme. Then, in Sect. 6.5.2, we extend the core construction into a CUFE scheme, embedding AoS with a pair of algorithms $(\mathsf{TokGen}, \mathsf{Upd})$. Finally, Sect. 6.5.7 provide a formal security analysis, followed in Sect. 6.5.6 by extensive experimental results that show its practicality.

6.5.1 Core Construction of AoS

This section formally defines the partial decryption FE scheme we obtain from an RLWE scheme with public parameters $\mathsf{pp} = (n, m, q, \mathcal{X}_{\mathsf{sk}}, \mathcal{X}_e)$, as detailed in Definition 14. We parametrize pp by a security parameter λ, with $q = q(\lambda)$ such that $\log_2 \log_2 q \in \mathbb{N}^*$, $\mathcal{X}_{\mathsf{sk}}$ the uniform distribution over ternary polynomials in $\mathcal{R}$, denoted $\mathcal{R}_2 := \mathbb{Z}_2[X]/(X^n + 1)$, and $\mathcal{X}_e$ the Gaussian distribution in $\mathcal{D}_{\sigma,q}$ of standard derivation $\sigma = \sigma(\lambda)$ in $\mathcal{R}_q$. We denote $\mathcal{M} = \mathcal{R}_t = \mathbb{Z}_t[X]/(X^n + 1)$ the message space for a positive integer $t = t(\lambda)$, and $\mathcal{C} = \mathcal{R}_q \times \mathcal{R}_q$ the ciphertext space.

Setup During the setup step in algorithm 1, the secret key sk and the public key pk are generated as follows: sk_i is a ternary polynomial sampled randomly in $\mathbb{Z}_2[X]/(X^n + 1)$; a is a polynomial sampled randomly in $\mathbb{Z}_q[X]/(X^n + 1)$ and $\mathbf{e}_0$ and e_α are sampled from $\mathcal{D}^m_{q,\sigma}$ and $\mathcal{D}_{q,\sigma}$, respectively. A label α is a ternary polynomial in $\mathbb{Z}_2[X]/(X^n + 1)$.

The public key pk is constituted of m RLWE instances $\mathsf{pk}_i = (b_i, a) \in \mathcal{R}_q^2$, where $b_i = [-a \cdot \mathsf{sk}_i + t \cdot e_{0,i}]$, and a key $\mathsf{k}_\alpha = a \cdot \alpha + t \cdot e_\alpha$ associated to the label α.

Algorithm 1 $\mathsf{Setup}\left(1^\lambda, \mathsf{pp}\right)$

`Input:` Takes as input a security parameter λ and the public parameters pp.
`Output:` Outputs a secret key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_m)$ and an public key $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_m)$ and the k_α key.

1: Sample $a \leftarrow\!\!\$\ \mathcal{R}_q$ and $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_m)$ where $i \in [m], \mathsf{sk}_i \leftarrow\!\!\$\ \mathcal{R}_2$
2: Sample $e_0 = (e_{0,1} \cdots, e_{0,m})$ where $i \in [m], e_{0,i} \leftarrow \mathcal{D}^m_{q,\sigma}$ and $e_\alpha \leftarrow \mathcal{D}_{q,\sigma}$
3: Sample $\alpha = (\alpha_1, \cdots, \alpha_m)$ where $i \in [m], \alpha_i \leftarrow\!\!\$\ \mathcal{R}_2$ and set $\mathsf{k}_{\alpha_i} \leftarrow [a \cdot \alpha_i + t \cdot e_\alpha]_q$
4: Set $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_m)$, where $i \in [m]$, $\mathsf{pk}_i = (a, b_i) \in \mathcal{R}_q^2$, for $b_i \leftarrow [-a \cdot \mathsf{sk}_i + t \cdot e_{0,i}]_q$
5: **return** $(\mathsf{sk}, \mathsf{pk}, \mathsf{k}_\alpha)$

Encryption The encryption algorithm 2 is a probabilistic algorithm. On the input of a message $x \in \mathbb{Z}_t[X]/(X^n+1)$, the i-th public key $\mathsf{pk}_i = (b_i, a) \in \mathcal{R}_q^2$ and a label α, outputs a ciphertext $\mathbf{c} = (\mathsf{c}_1, \mathsf{c}_2)$.

In order to process that part, we sample u uniformly at random in $\mathcal{R}_2$ and the errors e_1, e_2 according to the Gaussian distribution $\mathcal{D}_{q,\sigma}$. The first component of the ciphertext is computed as $\mathsf{c}_1 = [b_i \cdot \alpha + t \cdot e_1 + u \cdot x]_q$ contains the message payload x, and its second part $\mathsf{c}_2 = [a \cdot \alpha + t \cdot e_2 - u]_q$ is a helping information for the decryption.

Algorithm 2 Enc $(\mathsf{pk}_i, x, \alpha)$

Input: Takes as input the i-th public key pk_i and a plaintext $x \in \mathcal{R}_t$.
Output: Outputs the ciphertext $\mathbf{c_x} = (\mathsf{c}_1, \mathsf{c}_2)$

1: Sample $u \leftarrow_\$ \mathcal{R}_2$, and $e_1, e_2 \leftarrow \mathcal{D}_{q,\sigma}$
2: Set $\mathsf{c}_1 \leftarrow [b_i \cdot \alpha + t \cdot e_1 + u \cdot x]_q$ and $\mathsf{c}_2 \leftarrow [a \cdot \alpha + t \cdot e_2 - u]_q$
3: **return** $\mathbf{c_x} = (\mathsf{c}_1, \mathsf{c}_2)$

Key Derivation The key derivation algorithm is a probabilistic algorithm that outputs a decryption key, depending on both an input value $v \in \mathcal{R}_t$ and a label α. This algorithm takes as input the secret key sk, the value v, and the key k_α. We sample the error e_3 in $\mathcal{D}_{q,\sigma}$ and compute the key $\mathsf{dk}_{\alpha,v} = (\mathsf{k}_1, \ldots, \mathsf{k}_m)$, where $\mathsf{k}_i \leftarrow \mathsf{k}_\alpha \cdot (\mathsf{sk}_i - v)$.

Algorithm 3 KDer $(\mathsf{sk}, \mathsf{k}_\alpha, v)$

Input: Takes as input the secret key sk, a key k_α, and v the value that we want to see in the ciphertext.
Output: Outputs the derivation key $\mathsf{dk}_{\alpha,v}$

1: Sample $e_3 \leftarrow \mathcal{D}_{q,\sigma}$
2: **for** $i \in [m]$ **do**
3: $\quad \mathsf{k}_i \leftarrow \mathsf{k}_\alpha \cdot (\mathsf{sk}_i - v) + t \cdot e_3$
4: **end for**
5: Set $\mathsf{dk}_{\alpha,v} = (\mathsf{k}_1, \ldots, \mathsf{k}_m)$
6: **return** $\mathsf{dk}_{\alpha,v}$

Decryption The decryption is done using the functional decryption key $\mathsf{dk}_{\alpha,v}$, without the knowledge of the secret key sk. This specificity guarantees partial decryption of the ciphertexts $\mathbf{c_x} = (\mathbf{c}_{x_1}, \ldots, \mathbf{c}_{x_m})$ for value v and label α.

Algorithm 4 Dec $(\mathsf{dk}_{\alpha,v}, \mathbf{c_x})$

Input: Take as input the derivation key $\mathsf{dk}_{\alpha,v}$, and a vector of m ciphertexts $\mathbf{c_x}$
Output: Outputs the partially decrypted $\mathbf{x}$ noted $\mathbf{y} = (y_1, \ldots, y_m)$.

1: **for** $i \in [m]$ **do**
2: $\quad$ Parse $(\mathsf{c}_1, \mathsf{c}_2) \leftarrow \mathbf{c}_{x_i}$ and $\mathsf{k}_i \leftarrow \mathsf{dk}_{\alpha,v}[i]$
3: $\quad$ Compute $y_i^* \leftarrow \mathsf{c}_1 + \mathsf{k}_i + v \cdot \mathsf{c}_2$
4: $\quad$ Reduce y_i^* into $[y_i^*]_t \in [0, t)$
5: $\quad$ Reduce $[y_i^*]_t$ into $y_i \in (-q/2, q/2]$
6: **end for**
7: **return** $\mathbf{y} = (y_1, \ldots, y_m)$

Correctness: Let $\lambda \in \mathbb{N}$ be an arbitrary security parameter, $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{Setup}\left(1^\lambda, \mathsf{pp}\right)$ the secret and public keys, $\alpha \in \mathcal{R}_2$ an arbitrary label and $\mathsf{dk}_{\alpha,v} \leftarrow \mathsf{KDer}(\mathsf{sk}, \mathsf{k}_\alpha, v)$. For any ciphertext $\mathbf{c}_{x_i} \leftarrow$

$\mathsf{Enc}(\mathsf{pk}_i, c_i, \alpha)$ of a message $x_i \in \mathcal{M}$, it follows that

$$y_i^* = \mathsf{c}_1 + \mathsf{k}_i + v \cdot \mathsf{c}_2 = (x_i - v) \cdot u + t \cdot E$$

where $E = e_{0,i} \cdot \alpha + e_1 + e_\alpha \cdot (\mathsf{sk}_i - v) + e_3 + v \cdot e_2$.

To guarantee that the decryption works correctly, we need to choose the parameters so that the noise overhead does not exceed the range $(-q/2, q/2]$, that is

$$\|y_i^*\| < q/2$$

We choose the parameters according to the article of Mono *et al.* [229], with $\sqrt{n\, V_{y_i^*}} < q/2$, where $V_{y_i^*}$ is the variance of y_i^*. First, remark that $V_{y_i^*} = V_{tE}$, as we only need to check the correctness in the case decryption works, that is $u\,(x_i - v) = 0$. It follows that

$$\begin{aligned} V_{y_i^*} = V_{tE} &= V_{t(e_{0i}\cdot\alpha + e_1 + e_\alpha\cdot(\mathsf{sk}_i - v) + e_3 + v\cdot e_2)} \\ &= t^2\, V_{e_{0i}\cdot\alpha + e_1 + e_\alpha\cdot(\mathsf{sk}_i - v) + e_3 + v\cdot e_2} \\ &= t^2\, \sigma^2\, (4n/3 + 2 + t^2/6) < q/2 \end{aligned}$$

where q is the ciphertext modulus, t the plaintext modulus, σ the derivation of the gaussian $\mathcal{D}_{q,\sigma}$ and n the degree of the quotient ring $\mathcal{R}$.

6.5.2 Updatable Functional Encryption

We now present the algorithms we designed to achieve ciphertext updatability, as defined in Definition 23. In our model, we assume that the token generation algorithm 5 is run by a trusted authority **KC**. It takes as input the secret key and two labels: δ_{in} is the original label, and δ_{out} is the label to which we want to make the update. It outputs a token $\mathsf{tk}_{\delta_{in}\to\delta_{out}}$.

Algorithm 5 $\mathsf{TokGen}(\mathsf{sk}, \delta_{in}, \delta_{out})$

Input: Takes as input the secret key sk and labels $\delta_{in}, \delta_{out}$
Outputs: Outputs the update token $\mathsf{tk}_{\delta_{in}\to\delta_{out}}$
1: Sample $\mathbf{e}_{\mathsf{tk}} \leftarrow_{\$} \mathcal{D}_{q,\sigma}^2$
2: Compute $\mathsf{tk}_{\delta_{in}\to\delta_{out}} \leftarrow \mathsf{pk} \cdot (\delta_{out} - \delta_{in}) + t \cdot \mathbf{e}_{\mathsf{tk}}$
3: Output $\mathsf{tk}_{\delta_{in}\to\delta_{out}}$

This update token can be used on a ciphertext to update its label using algorithm 6.

Algorithm 6 $\mathsf{Upd}(\mathsf{tk}_{\delta_{in}\to\delta_{out}}, c_{\delta_{in}})$

Input: Takes as input a token $\mathsf{tk}_{\delta_{in}\to\delta_{out}}$ and a ciphertext $c_{\delta_{in}}$ encrypted under δ_{in}.
Outputs: Outputs $c_{\delta_{out}}$ encrypted under the label δ_{out}
1: Compute $c_{\delta_{out}} \leftarrow \mathsf{tk}_{\delta_{in}\to\delta_{out}} + c_{\delta_{in}}$

Correctness: Let $\lambda \in \mathbb{N}$ be an arbitrary security parameter. Generate $(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{Setup}(\lambda, n)$ and choose $\delta_{in}, \delta_{out}$ two labels. Let $x \in \mathcal{M}$ be an arbitrary message, $c_{\delta_{in}} \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}, x, \delta_{in})$ and $\mathsf{dk}_{f,\delta_{in}}, \mathsf{dk}_{f,\delta_{out}}$ for labels $\delta_{in}, \delta_{out}$ respectively. We show that

$$\mathsf{Dec}(\mathsf{dk}_{f,\delta_{out}}, c_{\delta_{out}}) = \mathsf{Dec}(\mathsf{dk}_{f,\delta_{in}}, c_{\delta_{in}})$$

with overwhelming probability. Since we assume that $\mathsf{dk}_{f,\delta_{out}}$ is a valid decryption key, it is sufficient to show that $c_{\delta_{out}}$ is an encryption of x under the label δ_{out}. For every $i \in [n]$, we have

$$\begin{aligned} c_{\delta_{out}} &= \mathsf{tk}_{\delta_{in}\to\delta_{out}} + c_{\delta_{in}} \\ &= (b \cdot \delta_{out} + (e_1 + e_{\mathsf{tk},1}) + u \cdot x \ , \ a \cdot \delta_{out} + (e_2 + e_{\mathsf{tk},2}) + u) \\ &\in \mathsf{Im}\left(\mathsf{Enc}\left(\mathsf{pk}, x, \delta_{out}\right)\right) \end{aligned}$$

if and only if $(e_1 + e_{\mathsf{tk},1})$ and $(e_2 + e_{\mathsf{tk},2})$ are valid errors, which is the case in all RLWE encryption schemes.

We now describe a protocol to demonstrate the applicability of the construction. In this scenario, we consider a group of data owners **DO** = $(d_1, \ldots, d_m)$. We assume the existence of a trusted authority, or key curator, **KC** that generates the master and functional decryption keys. The users **U** = $(u_1, \ldots, u_\ell)$, who makes use of the data, are external entities only interacting with **KC** to request the decryption keys.

6.5.3 Application Scenario and Protocol

We now describe a protocol to demonstrate the applicability of the construction. In this scenario, we consider a group of data owners **DO** = $(d_1, \ldots, d_m)$. We assume the existence of a trusted authority, or key curator **KC**, that generates the master and functional decryption keys. The users **U** = $(u_1, \ldots, u_\ell)$, who make use of the data, are external entities only interacting with **KC** to request the decryption keys.

6.5.4 System and Threat model

To describe our protocol, we formally define the capabilities of the three entities involved and the threat model that covers the level of trust in each entity.

System Model. The considered system model consists of the following four entities:

- **Key Curator (KC):** A trusted authority responsible for the generation of the keys. It initializes the system and generates the secret key, while providing the public key to registered data owners **DO**. Upon a user's request u_k for a value v and identifier α_j, it provides a functional derivation key $\mathsf{dk}_{j,v}$.
- **Data Owners** $(\mathbf{DO} = (d_1, \ldots, d_m))$**:** Each d_j owns a private identifier α_j and registers in the system by sending α_j to **KC**. The set **DO** is not bounded and a new owner of data d_{m+1} can dynamically register by sending their private identifier α_{m+1}. A data owner d_j is responsible for the encryption of their own data, using α_j and the public key.
- **Users** $(\mathbf{U} = (u_1, \ldots, u_\ell))$**:** Users wish to learn the positions of a value v in a message $\mathbf{x}$. To do so, a user u_k sends a request to **KC** for the corresponding ciphertext $\mathbf{c_x}$ and obtains a derivation key $\mathsf{dk}_{j,v}$.

Threat Model. In the considered threat model, $\mathbf{KC}$ is a trusted authority. As it owns the master secret key, it can generate an unbounded polynomial number of functional decryption keys and therefore has the ability to decrypt all the ciphertexts provided by **DO**. On the other side, we assume that it provides the right functional decryption key $\mathsf{dk}_{j,v}$ corresponding to **U**'s requests. The owners of the data are not required to trust each other. We make the natural assumption that they encrypt the correct plaintexts. Specifically, the security of a data owner d_j's data is guaranteed even if all other data owners are malicious. Users **U** are trusted only with the information they can recover from the ciphertexts and the decryption keys provided by **KC**. Finally, we assume that users do not collude, which is a standard assumption in access-control systems where each user possesses individual credentials.

6.5.5 Protocol Construction

The main building blocks of our protocol are the following:

- An secure public key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$;
- An EUF-CMA secure signature scheme $\mathfrak{S} = (\mathsf{sign}, \mathsf{verify})$;
- A first and second pre-image resistant cryptographic hash function H;
- A synchronized clock between all entities.

Setup. As a first step, **KC** runs the Setup algorithm to generate the keys $(\mathsf{sk}, \mathsf{pk})$ for the cryptosystem, defined in Sect. 6.5.1. On the other side, each data owner $d_j \in \mathbf{DO}$ generates their private identifier α_j. In the protocol overview Figure 6.1, they register during the setup phase. However, registration in the system can be made dynamically at any given time. Eventually, d_j identifies itself by sending a message $\mathbf{m_j}$ to **KC**, which answers with a message $\mathbf{m_1}$ containing the public key. Formally, $\mathbf{m_j}$ and $\mathbf{m_1}$ develop as follows: $\mathbf{m_j} = \langle t_j,\ c_{\alpha_j},\ \sigma_{d_j}(\mathsf{H}(t_j \parallel c_{\alpha_j}))\rangle$ and $\mathbf{m}_1 = \langle t_1,\ \mathsf{pk},\ \sigma_C(\mathsf{H}(t_1 \parallel \mathsf{pk}))\rangle$ where t_j, t_1 are timestamps, σ_{d_j}, σ_C are d_j's and **KC**'s signatures, respectively, and c_{α_j} is the PKE encryption of $j \| \alpha_j$ for **KC**'s public key. Upon reception, each party verifies the freshness and integrity of the messages by respectively checking the timestamp and the signature. **KC** stores (j, α_j) for each registered data owner d_j. **Note:** In Sect. 6.5.1, the identifier α is generate in algorithm 1. However, in a real-life scenario, data owners will generate their α and will then send their requests for access and keys to the **KC**.

User Registration. On completion of **KC**'s setup, a user $u_k \in \mathbf{U}$ sends a registration message reg to the key curator **KC** containing a set of one or more pairs (j, v). Each pair corresponds to a decryption key request for the partial decryption of data collected by d_j, for a value v. The message is formed as: $\mathbf{m_2} = \langle t_2,\ \mathsf{reg},\ \sigma_{u_k}(\mathsf{H}(t_2 \parallel \mathsf{reg}))\rangle$.

Upon reception, **KC** verifies the freshness and integrity of the message by respectively checking the timestamp and the signature. **KC** first computes the functional decryption key $\mathsf{dk}_{j,v} \leftarrow \mathsf{KDer}(\mathsf{sk}, v, \alpha_j)$ for each requested pair (j, v). Then, it encrypts $\{\mathsf{dk}_{j,v}\}$ into a ciphertext we denote c_{dk}, using u_k's public key for PKE. It forwards it to u_k via the following message: $\mathbf{m_3} = \langle t_3,\ \mathbf{c}_{\mathsf{dk}},\ \sigma_C(\mathsf{H}(t_3 \parallel \mathbf{c}_{\mathsf{dk}}))\rangle$.

Again, u_k verifies the freshness and integrity of $\mathbf{m_3}$. If the message is valid, u_k stores the decryption keys $\{\mathsf{dk}_{j,v}\}$ for further use.

Encryption. We assume d_j owns data $\mathbf{x}$ as plaintexts. Using the public key pk obtained from $\mathbf{m_1}$, d_j encrypts $\mathbf{x}$ into an RLWE ciphertext $\mathbf{c_{j,x}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{x}, \alpha_j)$. He then outsources the data to the key curator as a message $\mathbf{m_4} = \langle t_4,\ \mathbf{c_{j,x}},\ \sigma_{d_j}(\mathsf{H}(t_4 \parallel \mathbf{c_{j,x}}))\rangle$.

Upon reception, **KC** verifies the freshness and integrity of the messages by respectively checking the timestamp and signature.If the verification succeeds, **KC** stores the ciphertexts obtained.

Data Access. Next, u_k sends a request for the encrypted data to **KC** via: $\mathbf{m_5} = \langle t_5,\ \mathsf{req},\ \sigma_{u_k}(\mathsf{H}(t_5 \parallel \mathsf{req}))\rangle$, where req denotes a request for a set of encrypted data, for example, data collected by a specific owner d_j or data collected during a specific time frame. Upon reception, **KC** verifies the message's integrity and freshness, and returns $\{\mathbf{c_{j,x}}\}$ the corresponding set of such messages $\mathbf{c_{j,x}}$ with the following message: $\mathbf{m_6} = \langle t_6,\ \{\mathbf{c_{j,x}}\},\ \sigma_C(\mathsf{H}(t_6 \parallel \{\mathbf{c_{j,x}}\}))\rangle$.

Upon reception, u_k verifies the freshness and integrity of the message. She can then run the functional decryption algorithm as described in algorithm 4 using $\mathsf{dk}_{j,v}$ to recover the partial decryption of a ciphertext $\mathbf{c_{j,x}}$ with respect to the value v. A complete overview of this protocol is illustrated in Figure 6.1.

Ciphertext Updatability. At any moment during the protocol, **KC** has the capability of generating an update token $\mathtt{tk}$ to update a ciphertext $\mathbf{c}_{\delta,x}$ into a ciphertext $\mathbf{c}_{\delta',x}$ for a new label δ'. This feature is only possible because we assumed that **KC** is a trusted authority. Otherwise, a malicious adversary that corrupts **KC** and at least one data owner could update any ciphertext into a newly formed ciphertext for

Co-funded by the European Union

ECCC

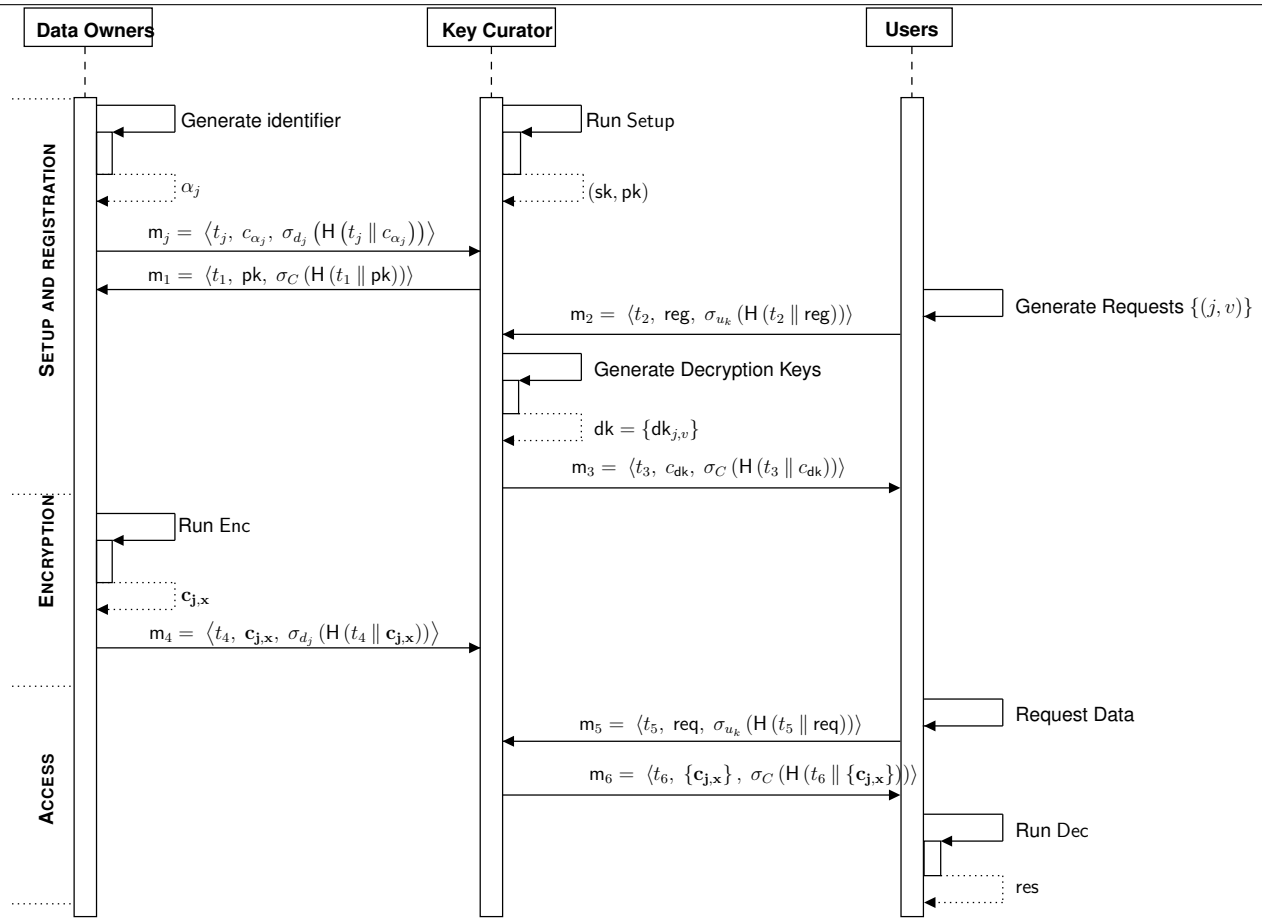

Figure 6.1: Use Case Protocol Overview

this corrupted user. In that case, they can access any data provided by any user. In this protocol, we assume that ciphertext updatability is inherent to the ACCESS step, as it involves updating data upon request of a user. It requires no additional security assumption, as **KC** already has access to all data.

In this section, we formally prove that the constructions we presented are secure in presence of an active adversary. We first provide a semantic security analysis, to prove the security of the underlying MCFE scheme on which the AoS protocol is built. The security of the latter is covered in Sect. 6.5.8, where we address its *robustness* and *secure access* security, two notions detailed below.

6.5.6 Semantic Security of AoS

Theorem 1. *The MCFE scheme AoS is IND-FE-CPA secure under the RLWE hardness assumption.*

Proof. To prove Theorem 1, we proceed by a game reduction from the AoS IND-FE-CPA game to the IND-CPA game for Definition 17. We introduce two PPT algorithms: $\mathcal{A}$ is the adversary in the RLWE game for challenger $\mathcal{C}$, and $\mathcal{B}$ is the adversary in the AoS game in which $\mathcal{A}$ is the challenger. We show that, if $\mathcal{B}$ has a non-negligible advantage in the AoS game, then $\mathcal{A}$ has a non-negligible advantage in the RLWE game.

Setup of RLWE game: The challenger $\mathcal{C}$ generates the secret and public key pair $(\mathsf{sk}, \mathsf{pk})$ for the RLWE game and forwards pk to $\mathcal{A}$. The latter submits two plaintexts x_0, x_1 to $\mathcal{C}$, who picks a bit $\beta \in \{0, 1\}$ at random and computes $\mathbf{c}_\beta \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, x_\beta)$, as:

$$\mathbf{c}_\beta = \begin{cases} \mathsf{c}_1 = b \cdot u + e_1 + x_\beta \\ \mathsf{c}_2 = a \cdot u + e_2 \end{cases}$$

where $u \leftarrow\!\!\$\ \mathcal{R}_2$ is a ternary polynomial, and $e_1, e_2 \leftarrow\!\!\$\ \mathcal{D}_{q,\sigma}$ are sampled from a discrete Gaussian. The challenger $\mathcal{C}$ returns $\mathbf{c}_\beta$ to $\mathcal{A}$.

Setup of AoS game: Then, $\mathcal{A}$ samples $u' \leftarrow\!\!\$\ \mathcal{R}_2$ and alters the ciphertext $\mathbf{c}_\beta = (\mathsf{c}_1, \mathsf{c}_2)$ into $\mathbf{c}'_\beta = (\mathsf{c}'_1, \mathsf{c}'_2)$ formed as follows: $\mathsf{c}'_1 \leftarrow \mathsf{c}_1 \cdot u'$ and $\mathsf{c}_2{}' \leftarrow \mathsf{c}_2 \cdot u' - u'$.

$$\mathbf{c}'_\beta = \begin{cases} \mathsf{c}_1{}' = b \cdot u \cdot u' + e_1 \cdot u' + x_\beta \cdot u' \\ \mathsf{c}_2{}' = a \cdot u \cdot u' + e_2 \cdot u' - u' \end{cases}$$

This alteration is valid, as it solely relies on the challenge ciphertext $\mathbf{c}_\beta$ and the polynomial u' generated by $\mathcal{A}$, and is hence independent from any private parameter held by the challenger $\mathcal{C}$.

Challenge: The adversary $\mathcal{A}$ forwards the altered ciphertext $\mathbf{c}'_\beta$ to the adversary $\mathcal{B}$ for the AoS game. Recall that this adversary is assumed to have a non-negligible advantage in this game. Therefore, the goal of $\mathcal{A}$ is to replicate this game for bit β and challenge plaintexts x_0, x_1, to have $\mathcal{B}$ acting as an oracle in the RLWE game against $\mathcal{C}$.

We remark that the ciphertext $\mathbf{c}'_\beta$ formed by $\mathcal{A}$ has the structure of a ciphertext for the MCFE scheme AoS, if $\mathcal{A}$ sets $\alpha \leftarrow (u \cdot u')$; $e'_1 \leftarrow (e_1 \cdot u')$; $e'_2 \leftarrow (e_2 \cdot u')$. Formally, $\mathbf{c}'_\beta$ is defined as:

$$\mathbf{c}'_\beta = \begin{cases} b \cdot \alpha + e'_1 + u' \cdot x_\beta \\ a \cdot \alpha + e'_2 + u' \end{cases}$$

$\mathcal{A}$ eventually sends $\mathsf{pk}, x_0, x_1, \mathbf{c}'_\beta$ to β. We note β' the guess made by $\mathcal{B}$ on the bit β chosen by $\mathcal{C}$. $\mathcal{B}$ forwards β' to $\mathcal{A}$, which submits this same guess β' to $\mathcal{C}$ in the RLWE game for which it is the adversary.

If we denote ϵ the advantage of $\mathcal{B}$ in the AoS game, $\mathcal{A}$ has at least an advantage ϵ in breaking the RLWE game against $\mathcal{C}$. It follows that, if $\mathcal{B}$ breaks AoS (*i.e.* it is not IND-FE-CPA secure), then $\mathcal{A}$ also breaks the RLWE IND-CPA game against $\mathcal{C}$. However, RLWE is IND-CPA secure by assumption, which concludes.

□

6.5.7 Semantic security of AoS

This section establishes the *semantic* security of the AoS construction described in earlier in the sense of ciphertext indistinguishability, as defined below, in Definition 24.

Definition 24 (IND-FE-CPA)**.** *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KDer}, \mathsf{Dec})$ *be a functional encryption scheme for message space* $\mathcal{M}$ *and functionality space* $\mathcal{F}$*. We say that* FE *is indistinguishable under chosen plaintext attack, or IND-FE-CPA, if for all PPT adversary* $\mathcal{A}$*, it holds that the advantage of* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\textit{ind-fe-cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda) = \Big| \mathbb{P}\left[\mathbf{Exp}^{\textit{ind-fe-cpa-0}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1\right] - \mathbb{P}\left[\mathbf{Exp}^{\textit{ind-fe-cpa-1}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1\right] \Big|$$

is negligible in λ*, where the experiment is defined as follows.*

$\mathbf{Exp}^{\text{ind-fe-cpa-}\beta}_{\mathsf{FE},\mathcal{A}}(\lambda)$

$\underline{\mathbf{Setup}(1^\lambda)}$
$L, V \leftarrow \emptyset$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\alpha \leftarrow\!\!\$\ \mathcal{R}_2$

$\underline{\mathbf{KeyDer}(v)}$
$V \leftarrow V \cup \{v\}$
$\mathsf{dk}_v \leftarrow \mathsf{KDer}(\mathsf{sk}, v, \alpha)$
return dk_v

$\underline{\mathbf{Challenge}(x_0, x_1)}$
$L \leftarrow L \cup \{(x_0, x_1)\}$
$c_\beta \leftarrow \mathsf{Enc}(\mathsf{pk}, x_\beta, \alpha)$

$\underline{\mathbf{Finalize}(\beta')}$
If $\exists\, v \in V, (x_0, x_1) \in L;$
$f_v(x_0) \neq f_v(x_1)$:
return $\perp$
return $\beta == \beta'$

Figure 6.2: Selective IND-FE-CPA security experiment parametrized by a bit $\beta \in \{0, 1\}$, a PPT adversary $\mathcal{A}$ and a security parameter λ.

To prove the IND-FE-CPA security of AoS, we implement a reduction to the security of the traditional RLWE from Definition 17.

Claim 1. The advantage of an adversary $\mathcal{A}$ to distinguish between $\mathbf{Exp}'^{\beta}_{\mathsf{FE},\mathcal{A}}$ and $\mathbf{Exp}^{\text{ind-fe-cpa-}\beta}_{\mathsf{FE},\mathcal{A}}$ is smaller than the advantage of $\mathcal{A}$ in the RLWE game.

Proof. To prove with a reduction using an adversary $\mathcal{B}$ that has a non-negligible advantage in AoS game. □

6.5.8 Security of the protocol

This section addresses the security of the AoS protocol described in Sect. 6.5.2 or, more accurately, proves that our protocol satisfies *robustness* and *secure access*.

Robustnessth guarantees that any party can access data they are authorized to retrieve. In its formal security Definition 25, an active PPT adversary $\mathcal{A}$ can interfere with the communications $\mathbf{m_j}$ and $\mathbf{m_1}, \ldots, \mathbf{m_6}$, but cannot influence the execution of the protocol. In this context, allowing $\mathcal{A}$ to corrupt users is meaningless, as they cannot influence the execution of **KC** and **DO**. However, it can impersonate **KC** or a data owner $d \in \mathbf{D}$ to send inappropriate initial messages to the parties and upload invalid contents. In the presence of such active adversary, we expect the protocol to guarantee that *all* authorized users can *correctly* retrieve partial decryption of the data.

Definition 25 (Robustness)**.** *A protocol Π for data owners $\mathbf{DO} = (d_1, \ldots, d_m)$, key curator **KC** and users $\mathbf{U} = (u_1, \ldots, u_\ell)$ is robust, if for all PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\textit{robust}}_{\Pi,\mathcal{A}}(\lambda) = \mathbb{P}\left[\mathbf{Exp}^{\textit{robust}}_{\Pi,\mathcal{A}}(\lambda) \rightarrow \mathtt{true}\right]$$

is negligible in λ, where the experiment $\mathbf{Exp}^{\textit{robust}}_{\Pi,\mathcal{A}}$ is defined as follows.

$\underline{\mathbf{Exp}^{\textit{robust}}_{\Pi,\mathcal{A}}(\lambda)}$

$L \leftarrow \emptyset$; forged $\leftarrow$ `false`
$(\mathsf{pp}, \mathsf{st}_{\mathbf{KC}}, \mathsf{st}_{\mathbf{D}}, \mathsf{st}_{\mathbf{U}}) \leftarrow \mathbf{Setup}(1^\lambda)$
$(u^*, \mathtt{req}^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \{\mathsf{msg}_d\}_{d\in\mathbf{D}} : \mathcal{O}_{\mathsf{robust}})$
Parse $\mathtt{req}^*$ *as* $(\mathsf{id}^*_{\mathbf{x}}, \mathsf{event})$
If $(\mathsf{forged} = \mathtt{true}) \vee$
$[\exists\, \mathbf{x}^*$ *such that* $(\mathbf{x}^*, \alpha^*, \mathbf{c}^*) \in L$
$\wedge\, c^*$ *is encrypted under* α^*
$\wedge\, \mathbf{x}^* \neq (\mathsf{res} \leftarrow (\mathsf{st}_{u^*}, \mathbf{c}^*))\,]$:
return `true`

where $\mathcal{O}_{\mathsf{robust}}$ *are the oracles in the robustness game, formally defined in Figure 6.3.*

Oracles $\mathcal{O}_{\mathsf{robust}}$

$\underline{\mathbf{Init}_{\mathbf{D}}(d_j, \mathsf{msg}_{d_j})}$
If $d_j \notin \mathbf{D}$: **return** $\perp$
If verification fails:
return $\perp$
Else: update st_{d_j} with msg_{d_j}
return `true`

$\underline{\mathbf{Init}_{\mathbf{U}}(u_k, \mathsf{msg}_{u_k})}$
If $u_k \notin \mathbf{U}$: **return** $\perp$
If verification fails:
return $\perp$
Else: update st_{u_k} with msg_{u_k}
return `true`

$\underline{\mathbf{UserReg}(\mathtt{reg})}$
Parse `reg` as $(u_k, \{v, \alpha\})$
If $u_k \notin \mathbf{U}$: **return** $\perp$
$\mathsf{st}_{u_k} \leftarrow \mathsf{KDer}(u_k, \{v, \alpha\})$
return msg_{u_k}

$\underline{\mathbf{DataUpload}\,(d_j, \mathbf{x}, \alpha_j)}$
If $d_j \notin \mathbf{D}$: **return** $\perp$
$\mathbf{c} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{st}_{d_j}, \mathbf{x}, \alpha_j)$
$\mathsf{msg} \leftarrow\!\!\$\ \mathbf{Upload}\,(\mathsf{st}_d, \mathbf{x}, \mathbf{c})$
$L \leftarrow L \cup \{\mathbf{x}, \alpha_j, \mathbf{c}\}$
return $(\mathbf{x}, \mathsf{msg})$

$\underline{\mathbf{Upload}(d_j, \mathsf{msg})}$
If $d_j \notin \mathbf{D}$: **return** $\perp$
Extract $(\mathbf{x}, \mathbf{c})$ from msg
If $\exists\, (\mathbf{x}, *, *) \in L$: **return** $\perp$
If $\mathbf{Init}_{\mathbf{D}}(d_j, \mathsf{msg}_{d_j}) = \mathtt{true}$:
$L \leftarrow L \cup \{\mathbf{x}, *, \mathbf{c}\}$
forged $= \mathtt{true}$
return $\mathbf{x}$
return $\perp$

$\underline{\mathbf{DataAccess}(\mathtt{req})}$
Parse `req` as $(\mathsf{id}_{\mathbf{x}}, \mathsf{event})$
If $\exists\, (\mathbf{x}, *, \mathbf{c}) \in L$: **return** $\mathbf{c}$
Else: **return** $\perp$

Figure 6.3: Oracles for the robustness game.

This security property is defined by an experiment involving a challenger $\mathcal{C}$, acting as $\mathbf{KC}$, and a PPT adversary $\mathcal{A}$ interacting with the protocol Π. $\mathcal{C}$ first runs $\mathbf{Setup}$, generating initial states for all parties and providing each data owner $d_j \in \mathbf{D}$ with an initialization message msg_{d_j}. Upon receiving msg_{d_j}, d_j verifies that msg_{d_j} is well formed and updates its local state. All initialization messages are revealed to $\mathcal{A}$, which can invoke the oracle $\mathbf{Init}_{\mathbf{D}}(d_j, \mathsf{msg}_{d_j})$ to further initialize owners, receiving a boolean response indicating success. User registration proceeds similarly: $\mathcal{A}$ may call $\mathbf{UserReg}\,(u_k, \{v, \alpha\}))$ to register a user u_k with request $\{v, \alpha\}$ and, upon success, an update message is returned. The user then updates its local state by calling $\mathbf{Init}_{\mathbf{U}}$ after verifying this update message.

The adversary can request a data owner d_j to upload a data $\mathbf{x}$ with identifier α by querying $\mathbf{DataUpload}$. The oracle runs Enc to produce a ciphertext $\mathbf{c}$, and executes $\mathbf{Upload}$ with d_j's local state and $(\mathbf{x}, \mathbf{c})$, to

Co-funded by the European Union
ECCC

output the upload message. A list L keeps record of every entry $\mathbf{x}, \alpha, \mathbf{c}$. The upload message is revealed to $\mathcal{A}$.

The adversary can also invoke $\mathbf{Upload}(d_j, \mathsf{msg})$ to upload arbitrary information. If msg is valid and no entry for $\mathbf{x}$ exists in L, the oracle returns $\mathbf{x}$ and sets a flag forged $=$ `true`, indicating successful forgery. To access encrypted data, $\mathcal{A}$ queries the $\mathbf{DataAccess}$ oracle with a data access token `req`. The token is structured as a specific identifier $\mathsf{id}_{\mathbf{x}}$ along with a potential event name event. Eventually, $\mathcal{A}$ receives the encrypted data $\mathbf{c}$ from the records in L.

There are two situations in which the adversary wins:

1. A data owner uploads $\mathbf{x}$ under α_j, but no authorized user (with decryption keys for α_j) can correctly recover $\mathbf{x}$.
2. A user gains access to data that was never uploaded by any data owner (i.e., the data is forged).

In either case, if there exists a user u^* and data identifier $\mathsf{id}^*_{\mathbf{x}}$ such that u^* fails to decrypt or the forged flag is set, $\mathcal{A}$ wins the game.

Secure Access: The second security definition we introduce ensures that unauthorized users cannot obtain any partial content of ciphertexts. This property is defined by a security game similar to the robustness game, but now the adversary $\mathcal{A}$ attempts to guess a random bit β chosen at the beginning of the game. The game maintains three lists: L for encrypted data, Ch for the challenges, and Cr for corrupted users. Unlike the robustness game, $\mathcal{A}$ may call a corruption oracle $\mathbf{Corrupt}$ to compromise users. Thus, $\mathbf{UserReg}$ is modified to permit corruption before and after user registration. However, $\mathcal{A}$ cannot run $\mathbf{Init}_{\mathbf{U}}$, since only $\mathcal{C}$ can initialize users. Honest users do not write data, so they cannot aid the adversary in leaking data. Similarly, the adversary has no capability to impersonate data owners.

The adversary issues a challenge via $\mathbf{Chall}$ by specifying a data owner d_j, two plaintexts $\mathbf{x_0}$ and $\mathbf{x_1}$, and α. The challenger encrypts and stores $\mathbf{x}_\beta$ in L. To prevent trivial wins, every corrupted user's registration requests $\{v, \alpha'\}$ must be such that $\alpha' \neq \alpha$. The game ends when $\mathcal{A}$ outputs a guess β', and the adversary wins if $\beta' = \beta$.

Definition 26 (Secure Access)**.** *A protocol Π for data owners $\mathbf{DO} = (d_1, \ldots, d_m)$, key curator **KC** and users $\mathbf{U} = (u_1, \ldots, u_\ell)$ is secure with respect to secure access, if for all PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\textit{s-acc}}_{\Pi,\mathcal{A}}(\lambda) = \left| \mathbb{P}\left[\mathbf{Exp}^{\textit{s-acc}}_{\Pi,\mathcal{A}}(\lambda) \to \mathtt{true}\right] - \frac{1}{2} \right|$$

is negligible in λ, where the experiment $\mathbf{Exp}^{\textit{s-acc}}_{\Pi,\mathcal{A}}$ is defined as follows.

where $\mathcal{O}_{\mathsf{robust}}$ are the oracles in the robustness game, formally defined in Figure 6.4.

$\underline{\mathbf{Exp}^{\textit{s-acc}}_{\Pi,\mathcal{A}}(\lambda)}$
$L, \mathsf{Ch}, \mathsf{Cr} \leftarrow \emptyset$; $\beta \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{pp}, \mathsf{st}_{\mathbf{KC}}, \mathsf{st}_{\mathbf{D}}, \mathsf{st}_{\mathbf{U}}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\beta' \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \{\mathsf{msg}_d\}_{d \in \mathbf{D}} : \mathcal{O}_{\mathsf{s-acc}})$
Return $(\beta = \beta')$

Theorem 2. *If the signature scheme $\mathfrak{S}$ is EUF-CMA secure, the construction AoS is robust in the random oracle model.*

Theorem 3. *If* PKE *is IND-CPA secure, $\mathfrak{S}$ is EUF-CMA secure, and the AoS scheme is IND-FE-CPA secure, the construction AoS is secure with respect to secure access in the random oracle model.*

Proof. The formal proof of Theorem 2 and Theorem 3 are rather straightforward, as they essentially rely on the unforgeability of the signature scheme $\mathfrak{S}$. This security assumption guarantees that no corrupted entity is able to forge a message to impersonate another party of the system. □

Oracles $\mathcal{O}_{\text{s-acc}}$

$\mathbf{Init_D}(d_j, \mathsf{msg}_{d_j})$

If $d_j \notin \mathbf{D}$: **return** $\perp$
If verification fails:
  return $\perp$
Else: update st_{d_j} with msg_{d_j}
return `true`

$\mathbf{UserReg}(\mathtt{reg})$

Parse `reg` as $(u_k, \{v, \alpha'\})$
If $u_k \notin \mathbf{U}$: **return** $\perp$
If $(u_k, \alpha') \in \mathsf{Cr}$:
  For all $\alpha \in \mathsf{Ch}$:
    If $\alpha = \alpha'$: **return** $\perp$
$\mathsf{st}_{u_k} \leftarrow \mathsf{KDer}(u_k, \{v, \alpha'\})$
If $u_k \in \mathsf{Cr}$: **return** st_{u_k}
Else **return** u_k

$\mathbf{DataUpload}(d_j, \mathbf{x}, \alpha_j)$

If $d_j \notin \mathbf{D}$: **return** $\perp$
$(\mathbf{x}, \mathbf{c}) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{st}_{d_j}, \mathbf{x}_\beta, \alpha_j)$
$L \leftarrow L \cup \{\mathbf{x}, \alpha_j, \mathbf{c}\}$
return $\mathbf{x}$

$\mathbf{DataAccess}(\mathtt{req})$

Parse `req` as $(\mathsf{id}_{\mathbf{x}}, \mathsf{event})$
If $\exists\ (\mathbf{x}, \alpha, \mathbf{c}) \in L$:
return $\mathbf{c}$

$\mathbf{Corrupt}(u_k)$

If $u_k \notin \mathbf{U}$: **return** $\perp$
For all $\alpha \in \mathsf{Ch}$:
  If $\alpha \in \mathsf{reg}_{u_k}$: **return** $\perp$
$\mathsf{Cr} \leftarrow \mathsf{Cr} \cup \{u_k\}$
return st_{u_k}

$\mathbf{Chall}(d, \mathbf{x}_0, \mathbf{x}_1, \alpha)$

If $d \notin \mathbf{D}$: **return** $\perp$
For all $u_k \in \mathsf{Cr}$:
  If $\alpha \in \mathsf{reg}_{u_k}$: **return** $\perp$
$\mathbf{c} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{st}_d, \mathbf{x}_\beta, \alpha)$
$L \leftarrow L \cup \{(\mathbf{x}_\beta, \alpha, \mathbf{c})\}$
$\mathsf{Ch} \leftarrow \mathsf{Ch} \cup \{\alpha\}$
return $\mathbf{c}$

Figure 6.4: Oracles for the secure access game.

To evaluate the practicality of our scheme, we implemented a proof-of-concept of AoS using the *Go* programming language and *Lattigo* library [174]. For testing and benchmarking, we followed best practices by using the Golang standard test package. Our experiments were conducted in a single-threaded environment on a PC with an Intel Core i5-1235 CPU @ 4.4GHz and 16GB of memory. Table 6.2 summarizes our benchmarks in terms of running time and memory usage.

6.5.9 AoS Benchmarks

To benchmark AoS, we follow the notation and algorithms in Sect. 6.5.1, using a modular approach for implementing different functions of $AoS = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KDer}, \mathsf{Dec}, \mathsf{TokGen}, \mathsf{Upd}\}$. As shown in Table 6.1, we define four parameter sets - **Params** $= \{\mathbf{S}, \mathbf{M}, \mathbf{L}, \mathbf{XL}\}$ - to evaluate performance across different system scales. For example, the smallest set (**S**) uses input vectors m of size 100 with a polynomial ring degree of $log_2 N = 12$ and ciphertext modulus $log_2 q = 39$. Additionally, for all the parameter sets, we have used the same plaintext modulus t equal to 2^{16}.

As shown in Table 6.2, the **running time** and **memory usage** of AoS increases with the change of parameters. The comparison between the smallest (**B1**) and largest benchmarks (**B4**), which involves increasing the number of samples by $10\times$, using a larger ring degree, and a larger ciphertext modulus, shows that the average running time is $25\times$ longer, and memory usage is $43\times$ higher.

Table 6.1: Parameters for Benchmarking

Params	$\mathbf{m}$	$\mathbf{log_2 N}$	$\mathbf{log_2 q}$	$\mathbf{log_2 t}$
Small	100	12	39	16
Medium	200	13	42	16
Large	500	14	43	16
XLarge	1000	15	47	16

Table 6.2: Ace of SPADE Benchmarks

Component	**Time** (ms/op)	**Memory Allocations** (KB/op)
B1 with parameter set S		
Setup	22.4327	25998
Encryption	0.4933	353
KeyDerivation	4.8752	6439
Decryption	3.5918	6410
TokenGen	0.4691	193
Update	0.0244	64
B2 with parameter set M		
Setup	74.4276	103384
Encryption	1.0265	706
KeyDerivation	7.1626	25678
Decryption	9.3621	25619
TokenGen	0.9092	386
Update	0.0406	128

Component	**Time** (ms/op)	**Memory Allocations** (KB/op)
B3 with parameter set L		
Setup	372.7643	515183
Encryption	1.1618	1412
KeyDerivation	39.7671	128164
Decryption	46.1236	128047
TokenGen	1.1352	772
Update	0.0564	256
B4 with parameter set XL		
Setup	1485.6854	2058371
Encryption	3.6025	2824
KeyDerivation	123.7577	512327
Decryption	150.0868	512095
TokenGen	2.1088	1544
Update	0.1498	512

6.5.10 The use of AoS in the framework of PiQASO

In healthcare, professionals analyse large volumes of sensitive patient data daily. Encryption must therefore protect this data while enabling selective access and computation when needed for analysis.

However, traditional cryptographic schemes permit only “all-or-nothing” decryption. As a result, they fail to support fine-grained access control and offer no protection against quantum attacks.

The AoS scheme addresses these limitations. It enables partial decryption in a post-quantum setting based on RLWE, allowing users to reveal only selected values while concealing other sensitive parts. Ciphertext updatability further permits changes to ciphertext properties without re-encryption.

In scenarios involving healthcare systems such as the *TelluCare Remote Patient Monitoring System* or *Smart Ambulance*, the AoS scheme allows authorised medical staff to decrypt only relevant data through partial decryption.

This capability arises from key derivation tied to specific attributes or identifiers of the data owner. Decryption keys are then distributed to authorised entities.

Thus, the AoS scheme suits applications like *UC TelluCare* and *UC Smart Ambulance*, where partial decryption, dynamic access control, and post-quantum security are essential.

The framework illustrated in Section 5.2 can be used to demonstrate the different interactions between the actors needed.

Chapter 7

Algorithmic Optimization and HW-based Acceleration

7.1 State-of-the-Art in Optimization and Acceleration

Beyond the design and implementation of PQC primitives, one additional target of PiQASO entails the *investigation of their efficiency and the provision of optimization and acceleration capabilities* in order to facilitate their integration in the far edge, thus fulfilling the strict latency and memory constraints of practical embedded systems. To this end, PiQASO employs a **SW/HW co-design approach**, where algorithmic SW optimizations aim to improve the performance of the algorithms themselves, while HW accelerations target the underlying mathematical operations that have been identified as bottlenecks.

The transition to PQC is driven by two key dimensions: cryptanalysis maturity (including countermeasures against side-channel attacks) and performance. While SW-based implementations offer ease of integration, they often face efficient challenges. HW-based solutions, including memory-mapped cryptographic peripherals, enhance performance and security by keeping secret variables out of vulnerable microcontroller registers, albeit at the cost of HW dependence and fragmentation. This underscores the value of SW/HW co-design, balancing efficiency, security, and interoperability. Co-designs can be categorized as **Tightly-Coupled Accelerations (TCAs)** and **Loosely-Coupled Accelerations (LCAs)** [COT15], depending on CPU modification requirements. Several lattice-based crypto approaches rely on tightly coupled HW modules [301], or combinations of loosely- and tightly- coupled designs for NTT and SHAKE acceleration [190]. Tightly coupled HW acceleration has also been used for SLH-DSA [211] and HQC [288]. Consideration of countermeasures is limited, with only masking applied for HQC [296]. SW/HW co-design is also crucial for establishing PQ RoTs, as the OpenTitan RoT [212] has been employed in loosely coupled instruction sets [300] and tightly coupled Keccak acceleration [4] in lattice-based crypto, though it cannot yet be used for resource-constrained devices due to computational demands. QR-TPMs have been examined for ML-KEM, ML-DSA [130], and SLH-DSA [248], but lack integration of countermeasures. TEEs have been considered for mitigating side-channel vulnerabilities [205], though without practical implementations.

The methodology employed by PiQASO entails performing a quantitative analysis - which has already been conducted - for identifying performance-critical mathematical operations and accelerate subroutines for all investigated PQ families, through the optimal composition of loosely- and tightly-coupled extensions. As will be demonstrated in Section 7.3, in the next version of this deliverable (D3.2), we will present the PiQASO approach for accelerating two core arithmetic operations: (i) **NTT**, which is essentially the polynomial multiplication which lies at the core of lattice-based constructions, and (ii) **Karatsuba**, which is the operation that lies at the core of HQC constructions.

Co-funded by the European Union
ECCC

In the context of PiQASO, this will be achieved through targeted performance optimizations, hardware accelerators, resource-constrained implementations (microcontrollers/IoT), and robust CMs against side-channel and physical attacks. Core PQ cryptosystems will run on CPUs, while performance-critical operations will be offloaded to HW accelerators in FPGAs. Note that, to this end, the **AMD Zynq UltraScale+ MPSoC ZCU104** FPGA module has been selected to in order to implement the envisioned accelerations. In principle, these can exist at all layers of the software stack: ranging from the low-level field arithmetic over the polynomial multiplications and logical operations. *The choice of the acceleration layer determines the boundary between hardware and software and enables different trade-offs between flexibility and efficiency*: accelerators can be re-used for different cryptosystems that are based on the same sub-routine. Note that **PiQASO is the first to employ a hybrid approach combining TCAs and LCAs, and memory-mapped cryptographic peripherals with Direct Memory Access (DMA)**. *LCAs will target long-running operations* (e.g., arithmetic in large prime fields requiring thousands of cycles, such as NTT), while *TCAs will handle faster operations* (e.g., 512–1024-bit arithmetic, such as Karatsuba), and DMA peripherals will accelerate resource-intensive tasks like matrix algebra and polynomial computations. This *HW/SW co-design strategy ensures universal acceleration for diverse PQC families by leveraging shared arithmetic operations*. The process involves quantitative analysis and design space exploration to classify performance-critical sub-routines for hardware acceleration versus those suited for software execution.

Note that all aforementioned accelerations will be integrated with the PiQASO PQC ensemble, incorporating validated state-of-the-art techniques (e.g., masking) and novel approaches like threshold implementations for code-based schemes. Thus, PiQASO will enable secure, efficient, and flexible implementations compatible with both legacy systems and future cryptographic advancements.

7.2 PIQASO algorithmic optimization

Practical Deployment and Efficiency Drivers The transition to quantum-resistant standards necessitates the practical deployment of lattice-based Post-Quantum Cryptography (PQC) across diverse platforms. Beyond theoretical security proofs, real-world adoption depends on highly optimized implementations that can operate within the stringent latency and memory constraints of embedded systems and high-throughput servers alike. Since these primitives are rooted in polynomial arithmetic over finite fields, the efficiency of the underlying multiplication algorithms serves as the primary engine for overall system performance. While the Number Theoretic Transform (NTT) provides asymptotic efficiency, its concrete software implementation is often hampered by the high frequency of modular reduction operations—such as Montgomery or Barrett reductions—which introduce significant computational overhead. In order to apply NTT in Kyber it is required that the arithmetic field has a primitive N-th root of unity.

To mitigate frequent modular reductions, we propose a domain-shift optimization that transposes the arithmetic burden from the modular finite field to the complex domain via fixed-point arithmetic. This approach results in faster multiplication while maintaining the constant-time execution that is essential for side-channel resistance. In addition, this construction does not require the existence of a primitive N-th root of unity in the arithmetic field. Therefore, the modulus q could be selected solely on the basis of security and not the existence of a primitive N-th root of unity.

Context and Mathematical Foundations Lattice-based PQC schemes, such as those targeting Ring Learning With Errors (RLWE) problem or its module variant, rely on the hardness of finding short vectors in structured lattices. These schemes achieve a favorable balance between cryptographic strength and communication overhead (e.g., ciphertext and public key sizes). Operations are typically defined over the

cyclotomic quotient ring:

$$\mathcal{R}q = \mathbb{Z}q[X]/(X^N + 1), \tag{7.1}$$

where N is a power of two (typically $N \in \{256, 512, 1024\}$) and q is a prime modulus. The core computational bottleneck is the negacyclic convolution of two polynomials $a(x), b(x) \in \mathcal{R}_q$. For a product $c(x) = a(x) \cdot b(x) \pmod{X^N + 1}$, the resulting coefficients c_k are determined by:

$$c_k = \sum i = 0^k a_i b_{k-i} - \sum_{i=k+1}^{N-1} a_i b_{N+k-i} \quad \text{for } k = 0, \ldots, N-1. \tag{7.2}$$

A direct implementation of this summation entails $O(N^2)$ scalar multiplications and additions. In high-security contexts where N is large, this quadratic complexity becomes a prohibitive barrier to real-time performance.

Standard NTT Approach vs. Computational Bottlenecks To overcome the $O(N^2)$ barrier, standard PQC implementations typically utilize the NTT, see [284] for a review. By selecting parameters such that $q \equiv 1 \pmod{2N}$, a primitive $2N$-th root of unity exists in $\mathbb{Z}_q$, allowing the transformation of polynomials into a "frequency" domain. In this domain, multiplication is reduced to a point-wise $O(N)$ operation, bringing the total complexity to $O(N \log_2 N)$. However, the NTT's reliance on modular arithmetic introduces a "reduction tax." In software, every "butterfly" operation in the transform requires modular reductions to prevent integer overflow and keep coefficients within the range $[0, q-1]$. These reductions involve multiple instructions (multiplications, shifts, and subtractions) that accumulate into a significant performance penalty on general-purpose CPUs.

Proposed Optimization: Complex Domain Mapping Our proposed optimization circumvents the "reduction tax" by shifting the arithmetic domain. We map the integer coefficients of the polynomials in $\mathbb{Z}_q$ to the complex field $\mathbb{C}$. Moreover, to circumvent the requirment for a Floating-Point Unit (FPU) and non-constant time operations between floats, we use fixed point arithmetic. The strategy leverages the fact that for specific ranges of N and q, the convolution can be computed using fixed-point arithmetic without loss of information. By performing the FFT, point-wise multiplication, and Inverse FFT (IFFT) in the complex domain, we replace expensive modular reductions with standard fixed-point instructions. The final result is only mapped back to the modular domain $\mathbb{Z}_q$ at the very end of the computation, significantly reducing the time duration of the multiplication. Crucially, by utilizing fixed-point arithmetic with a deterministic execution flow, we preserve the constant-time property required to thwart timing side-channel attacks.

Precision, Correctness, and Error Bound Analysis The validity of this mapping depends on ensuring that the numerical errors inherent in fixed-point approximations do not alter the discrete result. We employ a rigorous dual-selection strategy to guarantee mathematical equivalence to the NTT-based approach:

1. **Dynamic Range and Integer Width:** We analyze the maximum possible growth of coefficients during the convolution (the "worst-case" bit growth) to select an appropriate container size (e.g., 32-bit vs. 64-bit). This prevents overflows during the accumulation of N terms.
2. **Fractional Precision Allocation:** We calculate the required number of fractional bits needed to keep the cumulative quantization noise below 0.5. This ensures that when the final result is rounded to the nearest integer, it matches the exact value of the theoretical negacyclic convolution.

By satisfying these precision bounds, we achieve a high-performance implementation that is functionally identical to the standard modular approach but benefits from the highly optimized computational pipelines of modern processors.

7.3 PIQASO HW Acceleration Roadmap

As outlined in Section 7.1, the focus of the HW acceleration approach to be investigated in PiQASO is *to accelerate the mathematical operations which have been identified as bottlenecks in the core functionalities of KEMs and signature algorithms*, and beyond. We first focus on these two core functionalities present in each public-key infrastructure system but all tighlty- and loosely0coupled acceleration units will then be examined for integration in the more advanced crypto-suite of PiQASO in the context of Functional Encryption (FE) and Attribute-based Encryption (ABE). These are related to both **polynomial multiplication** for NTT, especially in the context of lattice-based crypto, and **binary multiplication**, especially the Karatsuba operation used in code-based crypto (e.g., HQC). Note that, through this approach, less focus is placed on acceleration of operations that have already been achieved through tightly-coupled accelerations, such as hashing functions (e.g., Keccak). In the following, we provide further details on the approach for the two types of operations targeted through HW accelerations.

Polynomial multiplication is typically employed to establish secured systems under the PQC principle. However, the challenge for most cryptosystems is in attaining a computing speed similar to unsecured systems (or secured under traditional crypto primitives). For a critical transactional activity, the encrypting process lag may allow cyber miscreants to leak information and have irreparable consequences. Hence faster cryptosystems are in demand to serve the purpose of providing security at no computational cost. The Number theoretic transform (NTT) allows compute-latency enhancements by multiplying higher-order polynomials in the transformed domain and achieve the necessary output. The hardware NTT design further tightens the security aspect and additionally offers power-performance benefits to the system. In PiQASO, the Longa-Naehrig reduction technique accompanied with the benefits of Montgomery (and possibly Barett) reduction will be adopted at different stages of NTT sequence generation, to design FPGA based hardware accelerator NTT design referred to as FastNTT.

7.3.1 HW Acceleration of NTT/INTT

As aforementioned, a core bottleneck of lattice-based crypto which will be targeted for HW-based acceleration in the context of PiQASO are **NTT** and **inverse NTT (INTT)**, which are core operations in lattice-based crypto (e.g., ML-KEM, ML-DSA). In the literature, SW/HW co-design approaches have been proposed [221], using high-speed pipelined HW modules dedicated to NTT/INTT, pointwise multiplication/addition, and for SHAKE, in order to accelerate the time-consuming operations in ML-DSA, as shown in Figure 7.1. In this approach, the modules are not connected sequentially and the data is moved through Module Control Logic (through an FSM). Based on a profiling on the ML-DSA algorithm, some blocks were implemented in a HW-based manner, while the rest of the implementation was performed in the software. However, such approaches may introduce performance issues, as the need to transfer data between software and hardware may introduce latency issues, and pure HW designs may have a higher degree of parallelism.

Thus, as part of the PiQASO approach, we aim to develop HW-based solutions in order to accelerate the underlying operations themselves. Specifically, we will first profile the **polynomial multiplication** performed as part of NTT/INTT and we will investigate all building blocks of FPGA-based NTT and INTT, such as **Butterfly Units**, which are responsible for performing addition, subtraction, and multiplication for each NTT stage. Based on this, we will identify the appropriate number of Butterfly Units in order to optimize these operations, when also considering appropriate key optimization mechanisms. These may include the **Montgomery Modular Multiplier** which reduces complex divisions to bit shifts and additions and uses DSP slices or Lookup Table (LUT)-based multipliers for efficiency. To ensure efficient resource usage, we will explore combining **Cooley-Tukey (CT)** and **Gentleman-Sande (GS)** Butterfly Units into a single module capable of performing both operations using in-place memory updates. For

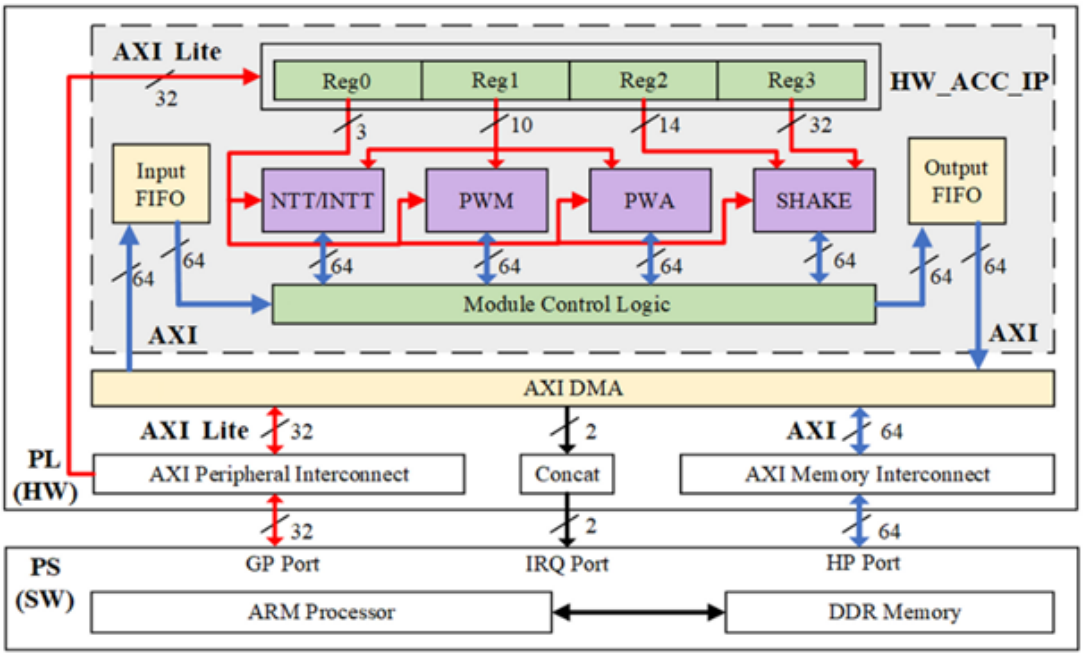

Figure 7.1: SW/HW co-design for accelerating ML-DSA

the core arithmetic, we will investigate the use of Montgomery reduction, along with modular addition and subtraction, to achieve low-latency processing. Additionally, we will investigate **on-the-fly twiddle factor generation**, initialized by an ARM processor, to minimize memory storage requirements. Finally, we will examine the **system-level integration of this accelerator**, specifically utilizing AXI memory-mapped interfaces for ARM configuration and AXI Stream interfaces for high-throughput data transfers. Additional optimization and reduction mechanisms will also be investigated, such as **Toom-Cook Reduction** and **KRED Modular Reduction**.

7.3.2 HW Acceleration of Karatsuba

PiQASO will be one of the first projects of its kind to investigate the acceleration of the binary multiplication performed in the context of code-based crypto, and especially HQC (which has recently been standardized by NIST). Specifically, we investigate the **Karatsuba** operation, which is essentially a divide-and-conquer algorithm that reduces the multiplication of two n-digit numbers to three multiplications of n/2-digit numbers. In this case, as Karatsuba is a binary multiplication and not pointwise, the acceleration methods outlined in the previous Section (e.g., Montgomery, KRED) are not applicable in this case. Thus, PiQASO will investigate techniques focusing on how to structure the high-degree coefficient of HQC for accelerating the Karatsuba operation, and will identify ways to further optimize the critical workflow of the binary multiplication.

As part of this approach, our research will investigate hardware acceleration techniques for binary polynomial multiplication, focusing on **minimizing latency** and **optimizing resource utilization** to achieve a **high operating frequency (Fmax)**. Specifically, in the case of the Karatsuba algorithm, the accelerator will be integrated into the broader system using **AXI4-Lite** for configuration and **AXI4-Stream** for data transfer. To optimally feed the central Karatsuba engine, we will focus on leveraging suitable data processing algorithms within either the **Processing System (PS)** or the **Programmable Logic (PL)**. Furthermore, we will investigate techniques for the efficient reconstruction of the final product. A key objective of this study is to explore the architectural design space of this pipeline. We will evaluate different Karatsuba engine configurations, analyzing the trade-offs of using narrower versus wider foundational

bit-widths, as well as test various chunk construction and reconstruction methodologies. While optimizing this specific Karatsuba-based dataflow is the primary focus, our efforts will also briefly review other potential acceleration methods for binary polynomial multiplication to contextualize our performance and resource metrics against alternative approaches.

Chapter 8

Conclusions

As outlined throughout this deliverable, a core target of PiQASO is to provide post-quantum security capabilities across heterogeneous systems covering the entire compute continuum, ranging from the cloud–based backend to the far edge, offering capabilities for converting embedded devices to their quantum secure equivalents. In this regard, D3.1 is the first in a series of PiQASO deliverables intended to provide all details pertaining to the **PiQASO PQC Ensemble**, offering all functionalities and primitives required for providing quantum secure capabilities to service graph chains belonging to various application domains, so that it can then be integrated in the context of the use case demonstrators in order to evaluate their applicability and performance, through the construction of an appropriate testbed.

To this end, in Chapter 2, we first provided a detailed analysis on the need for modern and legacy infrastructures - which currently rely on traditional cryptographic methods - to adopt quantum-secure solutions, considering the advent of quantum computers. To this end, we detail three core aspects of the need for PQC migration, namely (i) **crypto agility** for enabling the efficient adoption of different PQC schemes in case vulnerabilities are identified, thus providing the capability to adapt under uncertainty, (ii) **hybridization** during the transition period as a disciplined mechanism to preserve security guarantees and operational interoperability during the coexistence period of legacy and PQC-enabled components, and (iii) **interoperability with legacy infrastructures**, which determines whether PQC can be deployed at scale without deploying operational capability. Then, in Chapter 3, we provided a high-level overview of the main building blocks of PiQASO to be developed in order to fulfil the defined PQC transition roadmap.

Next, in continuation to the description of the PiQASO vision, in Chapter 4 we provided a detailed description of the state-of-the-art in the design of the types of PQC algorithms and constructions targeted by PiQASO, with particular focus on lattice-based constructions. Specifically, we first described the **Learning With Errors (LWE)** problem, which constitutes a core building block of post-quantum cryptography, and supports a wide range of cryptographic operations. We also provide descriptions of other LWE variants, such as RLWE and MLWE, which will be used in the context of PiQASO cryptographic constructions. We also provided detailed a state-of-the-art of all types of constructions and components targeted by PiQASO, namely **Key Encapsulation Mechanisms**, **Digital Signatures**, **QR Trusted Computing architectures**, integration into **high-end security protocols**, and **lattice trapdoors and gadgets**. We also provide an initial definition of the threat model considered by PiQASO, as a baseline to be expanded in the next version of this deliverable.

Next, we provided a detailed description of the foundations and the mathematical constructions behind core components of the PiQASO PQC Ensemble. In Chapter 5, we defined various fundamental concepts behind UPKE, which extends classical PKE notions to account for key evolution over time and constitutes a suitable approach for guaranteeing forward security in the encryption provided by PiQASO. We also provided a formal security model for the cloud that captures forward security and proves that

the cloud enriched with the update procedure satisfies forward security requirements. Then, in Chapter 6, we presented the mathematical foundations behind the Functional Encryption capabilities of PiQASO, specifically the LWE problem and its variants. We then presented the tools needed to construct the **ACE of SPADE (AoS)** scheme of PiQASO, which allows partial decryption and features ciphertext updatability.

Finally, in Chapter 7, we provided an overview of the **hybrid SW/HW co-design approach** of PiQASO, intended to enable the provision of PQC capability to embedded far-edge devices with limited computational capabilities, through the construction of advanced optimization and acceleration capabilities. Specifically, from a software perspective, we proposed a domain-shift optimization that transposes the arithmetic burden from the modular finite field to the complex domain via fixed-point arithmetic, thus resulting in faster multiplication, while also maintaining the constant-time execution that is essential for sode-channel resistance. From a hardware perspective, we set the foundations for the HW optimizations of the NTT operation used in lattice-based constructions, as well as the Karatsuba operation used in HQC constructions.

Compounding all the above, D3.1 is the first deliverable in a series of PiQASO deliverables aimed towards the design and implementation of the **PiQASO PQC Ensemble**, which also sets the foundation for the evaluation of the PiQASO offerings in the context of the envisioned use cases, in order to evaluate their performance and practical applicability.

Bibliography

[1] Information security, cybersecurity and privacy protection — evaluation criteria for it security — methodology for it security evaluation, 2022.

[2] Information security, cybersecurity and privacy protection — evaluation criteria for it security — part 1: Introduction and general model, 2022.

[3] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Amber Sprenkels. Faster kyber and dilithium on the cortex-m4. Cryptology ePrint Archive, Paper 2022/112, 2022.

[4] Amin Abdulrahman, Felix Oberhansl, Hoang Nguyen Hien Pham, Jade Philipoom, Peter Schwabe, Tobias Stelzer, and Andreas Zankl. Towards ML-KEM and ML-DSA on OpenTitan. Cryptology ePrint Archive, Paper 2024/1192, 2024.

[5] Calvin Abou Haidar, Benoît Libert, and Alain Passelègue. Updatable Public Key Encryption from DCR: Efficient Constructions With Stronger Security. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Los Angeles, United States, November 2022. ACM.

[6] Calvin Abou Haidar, Alain Passelègue, and Damien Stehlé. Efficient updatable public-key encryption from lattices. In Advances in Cryptology – ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4–8, 2023, Proceedings, Part V, page 342–373, Berlin, Heidelberg, 2023. Springer-Verlag.

[7] Melchior Aelmans, Gert Grammel, Siji Joseph, Sabyasachi Mukhopadhyay, Priyabrata Saha, Ranjan Sinha, and Aswin Surendran. Day one: Quantum-safe ipsec vpns. 2024.

[8] Agence nationale de la sécurité des systèmes d'information (ANSSI). Anssi views on the post-quantum cryptography transition. Position paper, ANSSI, January 2022.

[9] Shweta Agrawal, Rajarshi Biswas, Ryo Nishimaki, Keita Xagawa, Xiang Xie, and Shota Yamada. Cryptanalysis of boyen's attribute-based encryption scheme in tcc 2013. Des. Codes Cryptography, 90(10):2301–2318, October 2022.

[10] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. 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 17, pages 21–40. Springer, 2011.

[11] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II, page 224–255, Berlin, Heidelberg, 2021. Springer-Verlag.

Co-funded by the European Union
ECCC

[12] Carlos Aguilar-Melchor, Jean-Christophe Deneuville, Arnaud Dion, James Howe, Romain Malmain, Vincent Migliore, Mamuri Nawan, and Kashif Nawaz. Towards automating cryptographic hardware implementations: a case study of HQC. Cryptology ePrint Archive, Paper 2022/1425, 2022.

[13] Aikata Aikata, Ahmet Can Mert, Malik Imran, Samuel Pagliarini, and Sujoy Sinha Roy. Kali: A crystal for post-quantum security using kyber and dilithium. IEEE Transactions on Circuits and Systems I: Regular Papers, 70(2):747–758, 2023.

[14] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97, page 284–293, New York, NY, USA, 1997. Association for Computing Machinery.

[15] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pages 53–63, 2022.

[16] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the third round of the nist post-quantum cryptography standardization process. NIST Interagency/Internal Report NIST IR 8413-upd1, National Institute of Standards and Technology, July 2022.

[17] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. Protecting privacy through homomorphic encryption, pages 31–62, 2021.

[18] Martin Albrecht and contributors. lattice-estimator: Estimation of lattice attack complexities. https://github.com/malb/lattice-estimator, 2023. Accessed: 2025-01-05.

[19] Martin R. Albrecht, Benjamin Benčina, and Russell W. F. Lai. Hollow lwe: A new spin. In Serge Fehr and Pierre-Alain Fouque, editors, Advances in Cryptology – EUROCRYPT 2025, pages 363–392, Cham, 2025. Springer Nature Switzerland.

[20] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Journal of Mathematical Cryptology, 9(3):169–203, 2015.

[21] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Krämer, Patrick Longa, and Jefferson E. Ricardini. The lattice-based digital signature scheme qtesla. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, Applied Cryptography and Network Security, pages 441–460, Cham, 2020. Springer International Publishing.

[22] Joël Alwen, Benedikt Auerbach, Miguel Cueto Noval, Karen Klein, Guillermo Pascual-Perez, Krzyzstof Pietrzak, and Michael Walter. Cocoa: Concurrent continuous group key agreement. In Orr Dunkelman and Stefan Dziembowski, editors, Advances in Cryptology – EUROCRYPT 2022, pages 815–844, Cham, 2022. Springer International Publishing.

[23] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the ietf mls standard for group messaging. In Daniele Micciancio and Thomas Ristenpart, editors, Advances in Cryptology – CRYPTO 2020, pages 248–277, Cham, 2020. Springer International Publishing.

[24] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Modular design of secure group messaging protocols and the security of mls. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21, page 1463–1483, New York, NY, USA, 2021. Association for Computing Machinery.

[25] Joel Alwen, Sandro Coretti, Daniel Jost, and Marta Mularczyk. Continuous group key agreement with active security. In Rafael Pass and Krzysztof Pietrzak, editors, Theory of Cryptography, pages 261–290, Cham, 2020. Springer International Publishing.

[26] Joël Alwen, Georg Fuchsbauer, and Marta Mularczyk. Updatable public-key encryption, revisited. In Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part VII, page 346–376, Berlin, Heidelberg, 2024. Springer-Verlag.

[27] Joël Alwen, Marta Mularczyk, and Yiannis Tselekounis. Fork-resilient continuous group key agreement. In Helena Handschuh and Anna Lysyanskaya, editors, Advances in Cryptology – CRYPTO 2023, pages 396–429, Cham, 2023. Springer Nature Switzerland.

[28] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. Theor. Comp. Sys., 48(3):535–553, April 2011.

[29] Joël Alwen, Georg Fuchsbauer, Marta Mularczyk, and Doreen Riepel. Lattice-based updatable public-key encryption for group messaging. Cryptology ePrint Archive, Paper 2025/365, 2025.

[30] Dorian Amiet, Andreas Curiger, and Paul Zbinden. Fpga-based accelerator for post-quantum signature scheme sphincs-256. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(1):18–39, Feb. 2018.

[31] Dorian Amiet, Lukas Leuenberger, Andreas Curiger, and Paul Zbinden. Fpga-based sphincs+ implementations: Mind the glitch. In 2020 23rd Euromicro Conference on Digital System Design (DSD), pages 229–237, 2020.

[32] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. In 2018 IEEE symposium on security and privacy (SP), pages 962–979. IEEE, 2018.

[33] ANSSI. Anssi views on the post-quantum cryptography transition, March 2022. Accessed 2026-02-27.

[34] ANSSI. Anssi views on the post-quantum cryptography transition (2023 follow up), December 2023. Accessed 2026-02-27.

[35] Kyoichi Asano and Yohei Watanabe. Updatable public key encryption with strong cca security: Security analysis and efficient generic construction. In Topics in Cryptology – CT-RSA 2025: Cryptographers' Track at the RSA Conference 2025, San Francisco, CA, USA, April 28–May 1, 2025, Proceedings, page 223–246, Berlin, Heidelberg, 2025. Springer-Verlag.

[36] Benedikt Auerbach, Miguel Cueto Noval, Boran Erol, and Krzysztof Pietrzak. Continuous group-key agreement: Concurrent updates without pruning. In Yael Tauman Kalai and Seny F. Kamara, editors, Advances in Cryptology – CRYPTO 2025, pages 141–172, Cham, 2025. Springer Nature Switzerland.

[37] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation. 2017.

[38] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation (version 3.01). https://pq-crystals.org/kyber/, 2021.

[39] Erman Ayday, Jean Louis Raisaro, Urs Hengartner, Adam Molyneaux, and Jean-Pierre Hubaux. Privacy-preserving processing of raw genomic data. In Joaquin Garcia-Alfaro, Georgios Lioudakis, Nora Cuppens-Boulahia, Simon Foley, and William M. Fitzgerald, editors, Data Privacy Management and Autonomous Spontaneous Security, pages 133–147, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[40] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Markus Schönauer, Tobias Schneider, François-Xavier Standaert, and Christine van Vredendaal. Protecting dilithium against leakage: Revisited sensitivity analysis and improved implementations. Cryptology ePrint Archive, Paper 2022/1406, 2022.

[41] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabeand Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium – Algorithm Specifications and Supporting Documentation (Version 3.1). https://pq-crystals.org/dilithium/, 2021.

[42] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, Advances in Cryptology – CRYPTO 2017, pages 67–98, Cham, 2017. Springer International Publishing.

[43] Aritra Banerjee, Tirumaleswar Reddy.K, Dimitrios Schoinianakis, Tim Hollebeek, and Mike Ounsworth. Post-Quantum Cryptography for Engineers. Internet-Draft draft-ietf-pquip-pqc-engineers-14, Internet Engineering Task Force, August 2025. Work in Progress.

[44] Utsav Banerjee, Tenzin S. Ukyab, and Anantha P. Chandrakasan. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(4):17–61, Aug. 2019.

[45] Elaine Barker. Considerations for achieving cryptographic agility. NIST Cybersecurity White Paper (CSWP) 39, December 2025.

[46] Elaine Barker, Lily Chen, David Cooper, Dustin Moody, Andrew Regenscheid, Murugiah Souppaya, William Newhouse, Russ Housley, Sean Turner, William Barker, and Karen Kent. Considerations for achieving cryptographic agility: Strategies and practices. NIST Cybersecurity White Paper NIST CSWP 39, National Institute of Standards and Technology, December 2025.

[47] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The messaging layer security (mls) protocol rfc 9420, 2024.

[48] Carsten Baum, Olivier Blazy, Jan Camenisch, Jaap-Henk Hoepman, Eysa Lee, Anja Lehmann, Anna Lysyanskaya, René Mayrhofer, Hart Montgomery, Ngoc Khanh Nguyen, Bart Preneel, abhi shelat, Daniel Slamanig, Stefano Tessaro, Søren Eller Thomsen, and Carmela Troncoso. Cryptographers' feedback on the eu digital identity's arf. rlhttps://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/discussions/211, June 2024. available online at rlhttps://github.com/user-attachments/files/15904122/cryptographers-feedback.pdf.

[49] Hanno Becker, Vincent Hwang, Matthias J. Kannwischer, Bo-Yin Yang, and Shang-Yi Yang. Neon NTT: Faster dilithium, kyber, and saber on cortex-a72 and apple m1. Cryptology ePrint Archive, Paper 2021/986, 2021.

[50] Hanno Becker and Matthias J. Kannwischer. Hybrid scalar/vector implementations of keccak and SPHINCS+ on AArch64. Cryptology ePrint Archive, Paper 2022/1243, 2022.

[51] Luke Beckwith, Abubakr Abdulgadir, and Reza Azarderakhsh. A flexible shared hardware accelerator for nist-recommended algorithms crystals-kyber and crystals-dilithium with sca protection. In Topics in Cryptology – CT-RSA 2023: Cryptographers' Track at the RSA Conference 2023, San Francisco, CA, USA, April 24–27, 2023, Proceedings, page 469–490, Berlin, Heidelberg, 2023. Springer-Verlag.

[52] Luke Beckwith, Jens-Peter Kaps, and Kris Gaj. Fpga energy consumption of post-quantum cryptography. In Fourth PQC Standardization Conference. NIST, 2022.

[53] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. High-performance hardware implementation of crystals-dilithium. In 2021 International Conference on Field-Programmable Technology (ICFPT), pages 1–10, 2021.

[54] Daniel J. Bernstein, Karthikeyan Bhargavan, Shivam Bhasin, Anupam Chattopadhyay, Tee Kiah Chia, Matthias J. Kannwischer, Franziskus Kiefer, Thales B. Paiva, Prasanna Ravi, and Goutam Tamvada. Kyberslash: Exploiting secret-dependent division timings in kyber implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2025.

[55] Quentin Berthet, Andres Upegui, Laurent Gantel, Alexandre Duc, and Giulia Traverso. An area-efficient sphincs+ post-quantum signature coprocessor. In 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pages 180–187, 2021.

[56] Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting intermediate value leakage in dilithium: A template-based approach. Cryptology ePrint Archive, Paper 2023/050, 2023.

[57] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, pages 440–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[58] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Advances in Cryptology — EUROCRYPT '97, volume 1233 of Lecture Notes in Computer Science, pages 37–51. Springer, 1997.

[59] Dan Boneh, Saba Eskandarian, and Ben Fisch. Post-quantum epid signatures from symmetric primitives. In Topics in Cryptology – CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings, page 251–271, Berlin, Heidelberg, 2019. Springer-Verlag.

[60] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, Advances in Cryptology – CRYPTO 2013, pages 410–428, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[61] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pages 353–367, 2018.

[62] Joppe W. Bos, Brian Carlson, Joost Renes, Marius Rotaru, Daan Sprenkels, and Geoffrey P. Waters. Post-quantum secure boot on vehicle network processors. Cryptology ePrint Archive, Paper 2022/635, 2022.

[63] 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, pages 553–570, 2015.

[64] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(4):173–214, Aug. 2021.

[65] Botan Project. Post-quantum cryptography in botan. https://botan.randombit.net, 2024.

[66] Bouncy Castle Project. Latest nist pqc standards support in bouncy castle. https://www.bouncycastle.org/resources/latest-nist-pqc-standards-and-more-bouncy-castle-java-1-79/, 2024.

[67] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Advances in Cryptology – CRYPTO 2012, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[68] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[69] Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes, and Tobias Schneider. Exploiting small-norm polynomial multiplication with physical attacks: Application to crystals-dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024(2):359–383, Mar. 2024.

[70] Maxime Buser, Joseph K. Liu, Ron Steinfeld, Amin Sakzad, and Shi-Feng Sun. Dgm: A dynamic and revocable group merkle signature. In Computer Security – ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I, page 194–214, Berlin, Heidelberg, 2019. Springer-Verlag.

[71] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, Advances in Cryptology — EUROCRYPT 2003, pages 255–271, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[72] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT 2010, pages 523–552, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[73] Laurent Castelnovi, Ange Martinelli, and Thomas Prest. Grafting trees: A fault attack against the sphincs framework. In Tanja Lange and Rainer Steinwandt, editors, Post-Quantum Cryptography, pages 165–184, Cham, 2018. Springer International Publishing.

[74] E Chen, Yan Zhu, Kaitai Liang, and Hongjian Yin. Secure remote cloud file sharing with attribute-based access control and performance optimization. IEEE Transactions on Cloud Computing, 11(1):579–594, 2023.

[75] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 1223–1237, 2018.

[76] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1243–1255, 2017.

[77] Keng-Yu Chen and Jiun-Peng Chen. Masking floating-point number multiplication and addition of falcon: First- and higher-order implementations and evaluations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024(2):276–303, Mar. 2024.

[78] Liqun Chen, Changyu Dong, Nada El Kassem, Christopher J. P. Newton, and Yalan Wang. Hash-based direct anonymous attestation. In Post-Quantum Cryptography: 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16–18, 2023, Proceedings, page 565–600, Berlin, Heidelberg, 2023. Springer-Verlag.

[79] Liqun Chen, Changyu Dong, Christopher J. P. Newton, and Yalan Wang. Sphinx-in-the-head: Group signatures from symmetric primitives. ACM Trans. Priv. Secur., 27(1), February 2024.

[80] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23, pages 409–437. Springer, 2017.

[81] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. 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 22, pages 3–33. Springer, 2016.

[82] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In International Conference on the Theory and Application of Cryptology and Information Security, pages 377–408. Springer, 2017.

[83] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology, 33(1):34–91, 2020.

[84] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In Cyber Security Cryptography and Machine Learning, pages 1–19, Cham, 2021. Springer International Publishing.

[85] Valerio Cini, Sebastian Ramacher, Daniel Slamanig, Christoph Striecks, and Erkan Tairi. (inner-product) functional encryption with updatable ciphertexts. J. Cryptol., 37(1), December 2023.

[86] Cloudflare. Post-quantum cryptography (pqc) for ssl/tls, October 2025.

[87] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On post-compromise security. In IEEE 29th Computer Security Foundations Symposium, CSF 2016.

[88] Common Criteria Development Board. Common methodology for information technology security evaluation (cem): 2022 release 1, November 2022.

[89] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 1135–1150, 2021.

[90] Deirdre Connolly. Ml-kem post-quantum key agreement for tls 1.3. Internet-Draft draft-ietf-tls-mlkem-04, Internet Engineering Task Force, July 2025. Work in Progress.

[91] Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order polynomial comparison and masking lattice-based encryption. Cryptology ePrint Archive, Paper 2021/1615, 2021.

[92] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. Towards case-optimized hybrid homomorphic encryption. In Advances in Cryptology – ASIACRYPT 2022, pages 32–67, Cham, 2022. Springer Nature Switzerland.

[93] Emilio G. Cota, Paolo Mantovani, Giuseppe Di Guglielmo, and Luca P. Carloni. An analysis of accelerator coupling in heterogeneous architectures. In 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pages 1–6, 2015.

[94] Eric Crockett, Christian Paquin, and Douglas Stebila. Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. Cryptology ePrint Archive, Paper 2019/858, 2019.

[95] CRYSTALS-Dilithium Team. pq-crystals/dilithium: Reference and Optimized Implementations. https://github.com/pq-crystals/dilithium, 2023.

[96] Flo D, Michael P, and Britta Hale. Terminology for post-quantum traditional hybrid schemes. RFC 9794, June 2025.

[97] Fynn Dallmeier, Jan Peter Drees, Kai Gellert, Tobias Handirk, Tibor Jager, Jonas Klauke, Simon Nachtigall, Timo Renzelmann, and Rudi Wolf. Forward-secure 0-rtt goes live: Implementation and performance analysis in QUIC. In Stephan Krenn, Haya Schulmann, and Serge Vaudenay, editors, Cryptology and Network Security - 19th International Conference, CANS.

[98] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, Progress in Cryptology – AFRICACRYPT 2018, pages 282–305, Cham, 2018. Springer International Publishing.

[99] David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, Advances in Cryptology - EUROCRYPT 2018.

[100] David Derler, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Fine-grained forward secrecy: Allow-list/deny-list encryption and applications. In Nikita Borisov and Claudia Díaz, editors, Financial Cryptography and Data Security - 25th International Conference, FC 2021.

[101] Emir Dervisevic and Miralem Mehic. Overview of quantum key distribution technique within ipsec architecture. 05 2021.

[102] Sanjay Deshpande, Chuanqi Xu, Mamuri Nawan, Kashif Nawaz, and Jakub Szefer. Fast and efficient hardware implementation of HQC. Cryptology ePrint Archive, Paper 2022/1183, 2022.

[103] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of fiat-shamir with aborts. In Helena Handschuh and Anna Lysyanskaya, editors, Advances in Cryptology – CRYPTO 2023, pages 327–357, Cham, 2023. Springer Nature Switzerland.

[104] Whitfield Diffie and Martin E Hellman. New directions in cryptography. In Democratizing cryptography: the work of Whitfield Diffie and Martin Hellman, pages 365–390. 2022.

[105] Françoise Levy dit Vehel and Maxime Roméas. A composable look at updatable encryption. Cryptology ePrint Archive, Paper 2021/538, 2021.

[106] Yevgeniy Dodis, Matt Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. Intrusion-resilient public-key encryption. In Marc Joye, editor, Topics in Cryptology — CT-RSA 2003, pages 19–32, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[107] Yevgeniy Dodis, Matt Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. A generic construction for intrusion-resilient public-key encryption. In Tatsuaki Okamoto, editor, Topics in Cryptology – CT-RSA 2004, pages 81–98, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[108] Yevgeniy Dodis, Harish Karthikeyan, and Daniel Wichs. Updatable public key encryption in the standard model. In Kobbi Nissim and Brent Waters, editors, Theory of Cryptography, pages 254–285, Cham, 2021. Springer International Publishing.

[109] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, Advances in Cryptology — EUROCRYPT 2002, pages 65–82, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[110] Xingting Dong, Yupu Hu, Baocang Wang, Momeng Liu, and Wen Gao. Lattice-based revocable attribute-based encryption with decryption key exposure resistance. IET Information Security, 15(6):428–441, 2021.

[111] Ronny Döring and Marc Geitz. Post-quantum cryptography in use: Empirical analysis of the tls handshake performance. In NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, pages 1–5, 2022.

[112] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In Jonathan Katz and Hovav Shacham, editors, Advances in Cryptology – CRYPTO 2017, pages 537–569, Cham, 2017. Springer International Publishing.

[113] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, Public-Key Cryptography – PKC 2018, pages 3–31, Cham, 2018. Springer International Publishing.

[114] F Driscoll, M Parsons, and B Hale. Rfc 9794 terminology for post-quantum traditional hybrid schemes. 2025.

[115] Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, Advances in Cryptology – EUROCRYPT 2015, pages 617–640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[116] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(1):238–268, Feb. 2018.

[117] Max Duparc, Tako Boris Fouotsa, and Serge Vaudenay. Silbe: An updatable public key encryption scheme from lollipop attacks. In Selected Areas in Cryptography – SAC 2024: 31st International Conference, Montreal, QC, Canada, August 28–30, 2024, Revised Selected Papers, Part I, page 151–177, Berlin, Heidelberg, 2025. Springer-Verlag.

[118] Simone Dutto, Davide Margaria, Carlo Sanna, and Andrea Vesco. Toward a post-quantum zero-knowledge verifiable credential system for self-sovereign identity. Cryptology ePrint Archive, Paper 2022/1297, 2022.

[119] Edward Eaton, David Jao, Chelsea Komlo, and Youcef Mokrani. Towards post-quantum key-updatable public-key encryption via supersingular isogenies. In Riham AlTawy and Andreas Hülsing, editors, Selected Areas in Cryptography, pages 461–482, Cham, 2022. Springer International Publishing.

[120] Rachid El Bansarkhani and Rafael Misoczki. G-merkle: A hash-based group signature scheme from standard assumptions. In Tanja Lange and Rainer Steinwandt, editors, Post-Quantum Cryptography, pages 441–463, Cham, 2018. Springer International Publishing.

[121] Ali El Kaafarani and Shuichi Katsumata. Attribute-based signatures for unbounded circuits in the rom and efficient instantiations from lattices. In Michel Abdalla and Ricardo Dahab, editors, Public-Key Cryptography – PKC 2018, pages 89–119, Cham, 2018. Springer International Publishing.

[122] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, Advances in Cryptology, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[123] Mohamed ElGhamrawy, Melissa Azouaoui, Olivier Bronchain, Joost Renes, Tobias Schneider, Markus Schönauer, Okan Seker, and Christine van Vredendaal. From mlwe to rlwe: A differential fault attack on randomized and deterministic dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023(4):262–286, Aug. 2023.

[124] Encryption Consulting. Current landscape of post-quantum cryptography migration, 2025. Accessed 2026-02-27.

[125] ENISA. Application of attack potential to smartcards, January 2024.

[126] European Commission. Commission implementing regulation (eu) 2024/482 of 31 january 2024 laying down rules for the application of regulation (eu) 2019/881 as regards the adoption of the european common criteria-based cybersecurity certification scheme (eucc), January 2024.

[127] European Union. Regulation (eu) 2024/1183 of the european parliament and of the council of 11 april 2024 amending regulation (eu) no 910/2014 as regards establishing the european digital identity framework, April.

[128] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch., 2012:144, 2012.

[129] Shisen Fang, Shaojun Yang, and Yuexin Zhang. Inner product encryption from ring learning with errors. Cybersecurity, 3:1–11, 2020.

[130] Luís Fiolhais and Leonel Sousa. Qr tpm in programmable low-power devices, 2023.

[131] 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. 2019.

[132] Pierre-Alain Fouque, Paul Kirchner, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Key recovery from gram-schmidt norm leakage in hash-and-sign signatures over NTRU lattices. Cryptology ePrint Archive, Paper 2019/1180, 2019.

[133] Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. Masked accelerators and instruction set extensions for post-quantum cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(1):414–460, Nov. 2021.

[134] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. Swoosh: Efficient lattice-based non-interactive key exchange. Cryptology ePrint Archive, Paper 2023/271, 2023.

[135] Lydia Garms, Taofiq K. Paraïso, Neil Hanley, Ayesha Khalid, Ciara Rafferty, James Grant, James Newman, Andrew J. Shields, Carlos Cid, and Maire O'Neill. Experimental integration of quantum key distribution and post-quantum cryptography in a hybrid quantum-safe cryptosystem. Advanced Quantum Technologies, 7(4):2300304, 2024.

[136] Stefan-Lukas Gazdag, Sophia Grundner-Culemann, Tobias Guggemos, Tobias Heider, and Daniel Loebenberger. A formal analysis of ikev2's post-quantum extension. In Proceedings of the 37th Annual Computer Security Applications Conference, ACSAC '21, page 91–105, New York, NY, USA, 2021. Association for Computing Machinery.

[137] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[138] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC '08, page 197–206, New York, NY, USA, 2008. Association for Computing Machinery.

[139] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, Advances in Cryptology — ASIACRYPT 2002, pages 548–566, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[140] Aymeric Genêt. On protecting SPHINCS+ against fault attacks. Cryptology ePrint Archive, Paper 2023/042, 2023.

[141] Aymeric Genêt, Matthias J. Kannwischer, Hervé Pelletier, and Andrew McLauchlan. Practical fault injection attacks on SPHINCS. Cryptology ePrint Archive, Paper 2018/674, 2018.

[142] Lewis Glabush, Patrick Longa, Michael Naehrig, Chris Peikert, Douglas Stebila, and Fernando Virdia. FrodoKEM: A CCA-secure learning with errors key encapsulation mechanism. IACR Communications in Cryptology, 2(3), 2025.

[143] GlobalPlatform. Tee system architecture. GlobalPlatform Specification, February 2019. GlobalPlatform Device Technology.

[144] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Annual International Cryptology Conference, pages 112–131. Springer, 1997.

[145] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, Advances in Cryptology – EUROCRYPT 2014, pages 578–602, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[146] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information, page 173–201. Association for Computing Machinery, New York, NY, USA, 2019.

[147] Ruben Gonzalez and Thom Wiggers. KEMTLS vs. post-quantum TLS: Performance on embedded systems. Cryptology ePrint Archive, Paper 2022/1712, 2022.

[148] Google BoringSSL Team. Ml-kem support in boringssl. https://commondatastorage.googleapis.com/chromium-boringssl-docs/mlkem.h.html, 2024.

[149] Christian Göth, Sebastian Ramacher, Daniel Slamanig, Christoph Striecks, Erkan Tairi, and Alexander Zikulnig. Optimizing 0-rtt key exchange with full forward security. In Francesco Regazzoni and Apostolos P. Fournaris, editors, Proceedings of the 2023 on Cloud Computing Security Workshop, CCSW 2023.

[150] Guillaume Goy, Antoine Loiseau, and Philippe Gaborit. A new key recovery side-channel attack on hqc with chosen ciphertext. 09 2022.

[151] Guillaume Goy, Antoine Loiseau, and Phlippe Gaborit. A New Key Recovery Side-Channel Attack on HQC with Chosen Ciphertext. Lecture Notes in Computer Science, 13512(978-3-031-17233-5):353 – 371, 2022. International Conference on Post-Quantum Cryptography - PQCrypto 2022.

[152] Guillaume Goy, Antoine Loiseau, and Phlippe Gaborit. Estimating the strength of horizontal correlation attacks in the hamming weight leakage model: A side-channel analysis on HQC KEM. In WCC 2022: The Twelfth International Workshop on Coding and Cryptography, page WCC_2022_paper_48, Rostock, Germany, March 2022.

[153] Guillaume Goy, Julien Maillard, Philippe Gaborit, and Antoine Loiseau. Single trace HQC shared key recovery with SASCA. Cryptology ePrint Archive, Paper 2023/1590, 2023.

[154] Denisa O. C. Greconici, Matthias J. Kannwischer, and Amber Sprenkels. Compact dilithium implementations on cortex-m3 and cortex-m4. Cryptology ePrint Archive, Paper 2020/1278, 2020.

[155] Morgane Guerreau, Ange Martinelli, Thomas Ricosset, and Mélissa Rossi. The hidden parallelepiped is back again: Power analysis attacks on falcon. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(3):141–164, Jun. 2022.

[156] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-rtt key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, Advances in Cryptology - EUROCRYPT 2017.

[157] Naina Gupta, Arpan Jati, Anupam Chattopadhyay, and Gautam Jha. Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-dilithium. Cryptology ePrint Archive, Paper 2022/496, 2022.

[158] Tim Güneysu, Markus Krausz, Tobias Oder, and Julian Speith. Evaluation of lattice-based signature schemes in embedded systems. In 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pages 385–388, 2018.

[159] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data. Cryptology ePrint Archive, 2018.

[160] Darrel Hankerson, Scott Vanstone, and Alfred Menezes. Guide to elliptic curve cryptography. Springer, 2004.

[161] Thomas Hanson, Qian Wang, Santosh Ghosh, Fernando Virdia, Anne Reinders, and Manoj R. Sastry. Optimization for SPHINCS+ using intel secure hash algorithm extensions. Cryptology ePrint Archive, Paper 2022/1726, 2022.

[162] Mohammad Zahidul Hasan, Md Safiur Rahman Mahdi, Md Nazmus Sadat, and Noman Mohammed. Secure count query on encrypted genomic data. Journal of Biomedical Informatics, 81:41–52, 2018.

[163] Shiyang He, Hui Li, Fenghua Li, and Ruhui Ma. A lightweight hardware implementation of crystals-kyber with fpga and asic evaluation. Journal of Integrated Information Systems, 12(11):1–19, 2024.

[164] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. First-order masked kyber on ARM cortex-m4. Cryptology ePrint Archive, Paper 2022/058, 2022.

[165] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. Cryptology ePrint Archive, Paper 2023/098, 2023.

[166] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, Algorithmic Number Theory, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[167] Tim Hollebeek, Sophie Schmieg, and Bas Westerbaan. Use of ml-dsa in tls 1.3. Internet-Draft draft-ietf-tls-mldsa-00, Internet Engineering Task Force, May 2025. Work in Progress.

[168] James Howe and Bas Westerbaan. Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM cortex m7. Cryptology ePrint Archive, Paper 2022/405, 2022.

[169] Senyang Huang, Rui Qi Sim, Chitchanok Chuengsatiansup, Qian Guo, and Thomas Johansson. Cache-timing attack against hqc. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023(3):136–163, Jun. 2023.

[170] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. Xmss: extended merkle signature scheme. RFC 8391, May 2018.

[171] IETF. Hybrid key exchange in tls 1.3, April.

[172] IETF. Draft new supplement itu-t y.supp.qkdn-uc use cases of quantum key distribution networks. 2022.

[173] IETF. Hybrid key exchange in tls 1.3. Active Internet-Draft (tls WG), 2025.

[174] Tune Insight. Lattigo v5. Online: https://github.com/tuneinsight/lattigo, November 2023. EPFL-LDS, Tune Insight SA.

[175] IonQ, Inc. Ionq technical roadmap. https://www.ionq.com/roadmap, 2026. Accessed: 2026-02-13.

[176] ISO/IEC. Iso/iec 17825:2016 — information technology — security techniques — testing methods for the mitigation of non-invasive attack classes against cryptographic modules, 2016.

[177] ISO/IEC. Iso/iec 17825:2024 — information technology — security techniques — testing methods for the mitigation of non-invasive attack classes against cryptographic modules, 2024.

[178] Malika Izabachène, Lucas Prabel, and Adeline Roux-Langlois. Identity-based encryption from lattices using approximate trapdoors. In Information Security and Privacy: 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5–7, 2023, Proceedings, page 270–290, Berlin, Heidelberg, 2023. Springer-Verlag.

[179] Pratima Jana and Ratna Dutta. UPKE and UKEM schemes from supersingular isogenies. Cryptology ePrint Archive, Paper 2025/1010, 2025.

[180] Arpan Jati, Naina Gupta, Anupam Chattopadhyay, and Somitra Kumar Sanadhya. A configurable crystals-kyber hardware implementation with side-channel protection. Cryptology ePrint Archive, Paper 2021/1189, 2021.

[181] Huiwen Jia, Yupu Hu, and Chunming Tang. Lattice-based hash-and-sign signatures using approximate trapdoor, revisited. IET Information Security, 16:41–50, 07 2021.

[182] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2019, pages 159–188, Cham, 2019. Springer International Publishing.

[183] Panos Kampanakis and Michael Kallitsis. Faster post-quantum tls handshakes without intermediate ca certificates. In Shlomi Dolev, Jonathan Katz, and Amnon Meisels, editors, Cyber Security, Cryptology, and Machine Learning, pages 337–355, Cham, 2022. Springer International Publishing.

[184] Panos Kampanakis, Peter Panburana, Ellie Daw, and Daniel Van Geest. The viability of post-quantum x.509 certificates. IACR Cryptol. ePrint Arch., 2018:63, 2018.

[185] Tendayi Kamucheka, Alexander Nelson, David Andrews, and Miaoqing Huang. A masked pure-hardware implementation of kyber cryptographic algorithm. Cryptology ePrint Archive, Paper 2022/1547, 2022.

[186] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3):243–268, Jun. 2020.

[187] Emre Karabulut and Aydin Aysu. Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks. Cryptology ePrint Archive, Paper 2021/772, 2021.

[188] Emre Karabulut and Aydin Aysu. A hardware-software co-design for the discrete gaussian sampling of FALCON digital signature. Cryptology ePrint Archive, Paper 2023/908, 2023.

[189] Emre Karabulut and Aydin Aysu. Masking falcon's floating-point multiplication in hardware. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024(4):483–508, Sep. 2024.

[190] Patrick Karl, Jonas Schupp, Tim Fritzmann, and Georg Sigl. Post-quantum signatures on risc-v with hardware acceleration. ACM Transactions on Embedded Computing Systems, 23, 01 2023.

[191] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, page 525–537, New York, NY, USA, 2018. Association for Computing Machinery.

[192] Youngbeom Kim, Jingyo Song, and Seog Chung Seo. Accelerating falcon on armv8. IEEE Access, 10:44446–44460, 2022.

[193] Youngbeom Kim, Jingyo Song, Taek-Young Youn, and Seog Chung Seo. Crystals-dilithium on armv8. Security and Communication Networks, 2022(1):5226390, 2022.

[194] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Proceedings of the 19th Annual International Cryptology Conference (CRYPTO'99), volume 1666 of Lecture Notes in Computer Science, pages 388–397. Springer, 1999.

[195] Elisabeth Krahmer, Peter Pessl, Georg Land, and Tim Güneysu. Correction fault attacks on randomized CRYSTALS-dilithium. Cryptology ePrint Archive, Paper 2024/138, 2024.

[196] Yen-Ting Kuo and Atsushi Takayasu. A lattice attack on CRYSTALS-kyber with correlation power analysis. Cryptology ePrint Archive, Paper 2023/1781, 2023.

[197] Georg Land, Pascal Sasdrich, and Tim Güneysu. A hard crystal - implementing dilithium on reconfigurable hardware. Cryptology ePrint Archive, Paper 2021/355, 2021.

[198] Adam Langley et al. The tls post-quantum experiment, October 2019.

[199] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 75(3):565–599, 2015.

[200] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 75(3):565–599, 2015.

[201] Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In International Conference on Machine Learning, pages 12403–12422. PMLR, 2022.

[202] Yongseok Lee, Jonghee Youn, Kevin Nam, Heon Hui Jung, Myunghyun Cho, Jimyung Na, Jong-Yeon Park, Seungsu Jeon, Bo Gyeong Kang, Hyunyoung Oh, and Yunheung Paek. An efficient hardware/software co-design for falcon on low-end embedded systems. IEEE Access, 12:57947–57958, 2024.

[203] Chen Li, Suwen Song, Jing Tian, Zhongfeng Wang, and Çetin Kaya Koç. An efficient hardware design for fast implementation of hqc. In 2023 IEEE 36th International System-on-Chip Conference (SOCC), pages 1–6, 2023.

[204] Lu Li, Qi Tian, Guofeng Qin, Shuaiyu Chen, and Weijia Wang. Compact instruction set extensions for dilithium. ACM Trans. Embed. Comput. Syst., 23(2), 2024.

[205] Xiaoguo Li, Bowen Zhao, Guomin Yang, Tao Xiang, Jian Weng, and Robert H. Deng. A survey of secure computation using trusted execution environments, 2023.

[206] Yamin Li, Jianghong Wei, Fuchun Guo, Willy Susilo, and Xiaofeng Chen. Robust decentralized multi-client functional encryption: Motivation, definition, and inner-product constructions. In Jian Guo and Ron Steinfeld, editors, Advances in Cryptology – ASIACRYPT 2023, pages 134–165, Singapore, 2023. Springer Nature Singapore.

[207] Benoît Libert and Radu Ţiţiu. Multi-client functional encryption for linear functions in the standard model from lwe. In Steven D. Galbraith and Shiho Moriai, editors, Advances in Cryptology – ASIACRYPT 2019, pages 520–551, Cham, 2019. Springer International Publishing.

[208] Yi-Kai Liu, Matthew Campagna, Sean Turner, Lily Chen, Andrew Regenscheid, Ray Perlner, William Polk, and Quynh Dang. Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters. NIST Special Publication 800-186, National Institute of Standards and Technology, January 2023.

[209] Alex Lombardi, Vinod Vaikuntanathan, and Thuy-Duong Vuong. Lattice trapdoors and IBE from middle-product LWE. IACR Cryptol. ePrint Arch., page 1067, 2019.

[210] Chris M. Lonvick and Tatu Ylonen. The Secure Shell (SSH) Protocol Architecture. RFC 4251, January 2006.

[211] Jonathan Lopez-Valdivieso and Rene Cumplido. Design and implementation of hardware-software architecture based on hashes for sphincs+. ACM Trans. Reconfigurable Technol. Syst., 17(4), October 2024.

[212] lowRISC. Opentitan, 2026.

[213] Fucai Luo and Saif Al-Kuwari. Attribute-based signatures from lattices: unbounded attributes and semi-adaptive security. Des. Codes Cryptography, 90(5):1157–1177, May 2022.

[214] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. Journal of the ACM, 60(6):1–35, November 2013.

[215] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. J. ACM, 60(6), November 2013.

[216] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. J. ACM, 60(6), November 2013.

[217] Suraj Mandal and Debapriya Basu Roy. Kid: A hardware design framework targeting unified ntt multiplication for crystals-kyber and crystals-dilithium on fpga. In 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID), pages 455–460, 2024.

[218] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, 2007.

[219] Mark Manulis, Daniel Slamanig, and Federico Valbusa. SoK: Updatable Public-Key Encryption. Cryptology ePrint Archive, Paper 2026/583, 2026.

[220] Gaoyu Mao, Donglong Chen, Guangyan Li, Wangchen Dai, Abdurrashid Ibrahim Sanka, Çetin Kaya Koç, and Ray C. C. Cheung. High-performance and configurable sw/hw co-design of post-quantum signature crystals-dilithium. ACM Trans. Reconfigurable Technol. Syst., 16(3), June 2023.

[221] Gaoyu Mao, Donglong Chen, Guangyan Li, Wangchen Dai, Abdurrashid Ibrahim Sanka, Çetin Kaya Koç, and Ray C. C. Cheung. High-performance and configurable sw/hw co-design of post-quantum signature crystals-dilithium. ACM Trans. Reconfigurable Technol. Syst., 16(3), June 2023.

[222] Sarah McCarthy, James Howe, Neil Smyth, Seamus Brannigan, and Máire O'Neill. Bearz attack falcon: Implementation attacks with countermeasures on the falcon signature scheme. pages 61–71, 01 2019.

[223] David McGrew, Michael Curcio, and Scott Fluhrer. Leighton-micali hash-based signatures. RFC 8554, April 2019.

[224] Jonas Meers and Doreen Riepel. Cca secure updatable encryption from non-mappable group actions. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, Post-Quantum Cryptography, pages 137–169, Cham, 2024. Springer Nature Switzerland.

[225] Peihan Miao, Sikhar Patranabis, and Gaven Watson. Unidirectional updatable encryption and proxy re-encryption from ddh. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, Public-Key Cryptography – PKC 2023, pages 368–398, Cham, 2023. Springer Nature Switzerland.

[226] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology – EUROCRYPT 2012, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[227] Microsoft Security Response Center. Microsoft's quantum-resistant cryptography is here. https://techcommunity.microsoft.com/t5/microsoft-security-blog/microsoft-s-quantum-resistant-cryptography-is-here/ba-p/4238780, 2024.

[228] Vincent Migliore, Benoit Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium: Efficient implementation and side-channel evaluation. Cryptology ePrint Archive, Paper 2019/394, 2019.

[229] Johannes Mono, Chiara Marcolla, Georg Land, Tim Güneysu, and Najwa Aaraj. Finding and evaluating parameters for bgv. In Progress in Cryptology - AFRICACRYPT 2023: 14th International Conference on Cryptology in Africa, Sousse, Tunisia, July 19–21, 2023, Proceedings, page 370–394, Berlin, Heidelberg, 2023. Springer-Verlag.

[230] Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Transition to post-quantum cryptography standards. NIST Interagency/Internal Report NIST IR 8547 (Initial Public Draft), National Institute of Standards and Technology, November 2024.

[231] Pietro Nannipieri, Stefano Di Matteo, Luca Zulberti, Francesco Albicocchi, Sergio Saponara, and Luca Fanucci. A risc-v post quantum cryptography instruction set extension for number theoretic transform to speed-up crystals algorithms. IEEE Access, 9:150798–150808, 2021.

[232] National Institute of Standards and Technology. Cryptographic algorithm validation program (cavp), October 2016.

[233] National Institute of Standards and Technology. Security requirements for cryptographic modules. Federal Information Processing Standards Publication (FIPS) 140-3, March 2019.

[234] National Institute of Standards and Technology. Digital signature standard (dss). Federal Information Processing Standards Publication FIPS 186-5, National Institute of Standards and Technology, February 2023.

[235] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Technical Report NIST FIPS 204, U.S. Department of Commerce, Washington, D.C., August 2024.

[236] National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard, 2024.

[237] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report NIST FIPS 203, U.S. Department of Commerce, Washington, D.C., August 2024.

[238] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard, 2024.

[239] National Institute of Standards and Technology. Nist releases first 3 finalized post-quantum encryption standards, August 2024. Accessed 2026-02-27.

[240] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Technical Report NIST FIPS 205, U.S. Department of Commerce, Washington, D.C., August 2024.

[241] National Institute of Standards and Technology. Fips 140-3 cryptographic module validation program (cmvp) management manual, October 2025.

[242] National Institute of Standards and Technology. Post-quantum cryptography standardization, 2025. Accessed 2026-02-27.

[243] National Institute of Standards and Technology. Fft-based lattice digital signature standard (fndsa). Technical Report NIST FIPS 206 (Draft), U.S. Department of Commerce, 2026. Forthcoming Standard.

[244] National Institute of Standards and Technology. Hamming quasi-cyclic (hqc) key encapsulation mechanism. Technical Report NIST FIPS 207 (Draft), U.S. Department of Commerce, 2026. Forthcoming Standard (Code-Based KEM).

[245] Duc Tri Nguyen, Viet B. Dang, and Kris Gaj. A high-level synthesis approach to the software/hardware codesign of ntt-based post-quantum cryptography algorithms. In 2019 International Conference on Field-Programmable Technology (ICFPT), pages 371–374, 2019.

[246] Duc Tri Nguyen and Kris Gaj. Fast falcon signature generation and verification using armv8 neon instructions. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, Progress in Cryptology - AFRICACRYPT 2023, pages 417–441, Cham, 2023. Springer Nature Switzerland.

[247] Ruben Niederhagen, Johannes Roth, and Julian Wälde. Streaming sphincs+ for embedded devices using the example of tpms. In Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18–20, 2022, Proceedings, page 269–291, Berlin, Heidelberg, 2022. Springer-Verlag.

[248] Ruben Niederhagen, Johannes Roth, and Julian Wälde. Streaming sphincs+ for embedded devices using the example of tpms. In Lejla Batina and Joan Daemen, editors, Progress in Cryptology - AFRICACRYPT 2022, pages 269–291, Cham, 2022. Springer Nature Switzerland.

[249] NIS Cooperation Group. A coordinated implementation roadmap for the transition to post-quantum cryptography. Official document, European Commission, July 2025.

[250] Camille Nuoskala, Hossein Abdinasibfar, and Antonis Michalas. Spade: Digging into selective and partial decryption using functional encryption. In 20th EAI International Conference on Security and Privacy in Communication Networks (SecureComm'24), SecureComm 2024, Cham, 2024. Springer Nature Switzerland.

[251] Tobias Oder, Julian Speith, Kira Höltgen, and Tim Güneysu. Towards practical microcontroller implementation of the signature scheme falcon. In Jintai Ding and Rainer Steinwandt, editors, Post-Quantum Cryptography, pages 65–80, Cham, 2019. Springer International Publishing.

[252] Open Quantum Safe. oqs-provider releases (openssl 3 provider for post-quantum and hybrid tls/x.509), 2025. Accessed 2026-02-27.

[253] Open Quantum Safe. Tls (open quantum safe applications guide), 2025. Accessed 2026-02-27.

[254] Open Quantum Safe Project. liboqs Performance Benchmarks. Technical report, Open Quantum Safe, 2024.

[255] Open Quantum Safe Project. Open quantum safe (liboqs). https://github.com/open-quantum-safe/liboqs, 2024.

[256] Open Quantum Safe Project. oqs-provider: Post-quantum algorithms for openssl 3. https://github.com/open-quantum-safe/oqs-provider, 2024.

[257] OpenSSL Foundation. The features of 3.5: Post-quantum cryptography, April 2025.

[258] OpenSSL Project. Evp_kem ml-kem documentation. https://docs.openssl.org/master/man7/EVP_KEM-ML-KEM/, 2024.

[259] Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from lwe. In International Conference on Cryptology and Information Security in Latin America, pages 127–148. Springer, 2021.

[260] T. Pauly, P. Wouters, et al. Mixing preshared keys in the internet key exchange protocol version 2 (ikev2) for post-quantum security. RFC 8784, June 2020.

[261] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Optimal efficiency of optimistic contract signing. In Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing, pages 113–122, 1998.

[262] Guru-Vamsi Policharla, Bas Westerbaan, Armando Faz-Hernández, and Christopher A Wood. Post-quantum privacy pass via post-quantum anonymous credentials. Cryptology ePrint Archive, Paper 2023/414, 2023.

[263] Post-Quantum Cryptography Coalition (PQCC). Heatmap: Current state of pqc standards and adoption, 2025. Accessed 2026-02-27.

[264] PQ Code Package Consortium. Pq code package: Verified and optimized post-quantum cryptography. https://github.com/pq-code-package, 2024.

[265] PQ Code Package Contributors. mlkem-native: High-performance and secure ml-kem implementations. https://github.com/pq-code-package/mlkem-native, 2024.

[266] QUBIP. Post-quantum cryptography in openpgp. March 2023.

[267] QUBIP. Hybridization for quantum-secure ipsec. 2024.

[268] Prasanna Ravi, Sourav Gupta, Anupam Chattopadhyay, and Shivam Bhasin. Improving Speed of Dilithium's Signing Procedure, pages 57–73. 03 2020.

[269] Prasanna Ravi, Thales Paiva, Dirmanto Jap, Jan-Pieter D'Anvers, and Shivam Bhasin. Defeating low-cost countermeasures against side-channel attacks in lattice-based encryption: A case study on crystals-kyber. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024(2):795–818, Mar. 2024.

[270] Tirumaleswar Reddy and Hannes Tschofenig. Guidance for migration to composite, dual, or pqc-only authentication. Internet-Draft draft-reddy-pquip-pqc-signature-migration-00, Internet Engineering Task Force, September 2025. Work in Progress.

[271] Tirumaleswar Reddy and Hannes Tschofenig. Post-quantum cryptography recommendations for tls-based applications. Internet-Draft draft-ietf-uta-pqc-app-01, Internet Engineering Task Force, February 2026. Work in Progress.

[272] Tirumaleswar K. Reddy, Dan Wing, and Yaroslav Rosomakho. Guidance for migration to composite, dual, or pqc authentication. Internet-Draft draft-reddy-pquip-pqc-signature-migration-01, Internet Engineering Task Force, October 2025. Work in Progress.

[273] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 56(6):1–40, 2009.

[274] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6):34:1–34:40, 2009.

[275] Joshua Renckens, Peter B. Rønne, Johann Großschädl, and Peter Y. A. Ryan. An evaluation of post-quantum and hybrid noise protocol variants on mobile devices. In Luciana Morogan, Peter Roenne, and Ion Bica, editors, Innovative Security Solutions for Information Technology and Communications, pages 149–169, Cham, 2025. Springer Nature Switzerland.

[276] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[277] Sara Ricci, Lukas Malina, Petr Jedlicka, David Smékal, Jan Hajny, Peter Cibik, Petr Dzurenda, and Patrik Dobias. Implementing CRYSTALS-Dilithium Signature Scheme on FPGAs. In Proceedings of the 16th International Conference on Availability, Reliability and Security. ACM, 2021.

[278] Sara Ricci, Lukas Malina, Petr Jedlicka, David Smekal, Jan Hajny, Petr Cibik, and Patrik Dobias. Implementing CRYSTALS-dilithium signature scheme on FPGAs. Cryptology ePrint Archive, Paper 2021/108, 2021.

[279] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms, 1978.

[280] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2):120–126, February 1978.

[281] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. Cryptology ePrint Archive, Paper 2022/878, 2022.

[282] Markku-Juhani O. Saarinen. Accelerating slh-dsa by two orders of magnitude with a single hash unit. In Leonid Reyzin and Douglas Stebila, editors, Advances in Cryptology – CRYPTO 2024, pages 276–304, Cham, 2024. Springer Nature Switzerland.

[283] Pakize Sanal, Emrah Karagoz, Hwajeong Seo, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. Kyber on ARM64: Compact implementations of kyber on 64-bit ARM cortex-a processors. Cryptology ePrint Archive, Paper 2021/561, 2021.

[284] Ardianto Satriawan and Rella Mareta. A complete beginner guide to the number theoretic transform (ntt). Cryptology ePrint Archive, Paper 2024/585, 2024. https://eprint.iacr.org/2024/585.

[285] Peter Schwabe, Douglas Stebila, and Thom Wiggers. More efficient post-quantum kemtls with pre-distributed public keys. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, Computer Security – ESORICS 2021, pages 3–22, Cham, 2021. Springer International Publishing.

[286] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum tls without handshake signatures. January 2022. Full version.

[287] Maximilian Schöffel, Johannes Feldmann, and Norbert Wehn. Code-based cryptography in iot: A hw/sw co-design of hqc, 2023.

[288] Maximilian Schöffel, Johannes Feldmann, and Norbert Wehn. Efficient hardware implementation of constant time sampling for hqc, 2025.

[289] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.

[290] Masoumeh Shafieinejad and Navid Nasr Esfahani. A scalable post-quantum hash-based group signature. Cryptology ePrint Archive, Paper 2019/1377, 2019.

[291] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science, pages 124–134. Ieee, 1994.

[292] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5):1484–1509, October 1997.

[293] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. Assessing the overhead of post-quantum cryptography in tls 1.3 and ssh. In Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '20, page 149–156, New York, NY, USA, 2020. Association for Computing Machinery.

[294] Daniel Slamanig and Christoph Striecks. Revisiting updatable encryption: Controlled forward security, constructions and a puncturable perspective. In Theory of Cryptography: 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 – December 2, 2023, Proceedings, Part II, page 220–250, Berlin, Heidelberg, 2023. Springer-Verlag.

[295] D. Soni, K. Basu, M. Nabeel, N. Aaraj, M. Manzano, and R. Karri. Hardware Architectures for Post-Quantum Digital Signature Schemes. Springer International Publishing, 2021.

[296] Maxime Spyropoulos, David Vigilant, Fabrice Perion, Renaud Pacalet, and Laurent Sauvage. Masked vector sampling for HQC. Cryptology ePrint Archive, Paper 2024/1106, 2024.

[297] Douglas Stebila, Scott Fluhrer, Shay Gueron, et al. Hybrid key exchange in tls 1.3. Internet-Draft draft-ietf-tls-hybrid-design-09, Internet Engineering Task Force, September 2023. Work in Progress.

[298] Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. In Roberto Avanzi and Howard Heys, editors, Selected Areas in Cryptography – SAC 2016, pages 14–37, Cham, 2017. Springer International Publishing.

[299] Tobias Stelzer, Felix Oberhansl, Jonas Schupp, and Patrick Karl. Enabling Lattice-Based Post-Quantum Cryptography on the OpenTitan Platform. 2023.

[300] Tobias Stelzer, Felix Oberhansl, Jonas Schupp, and Patrick Karl. Enabling lattice-based post-quantum cryptography on the opentitan platform. In Proceedings of the 2023 Workshop on Attacks and Solutions in Hardware Security, ASHES '23, page 51–60, New York, NY, USA, 2023. Association for Computing Machinery.

[301] Tobias Stelzer, Felix Oberhansl, Jonas Schupp, Patrick Karl, and Horia Turcuman. Extended version: enabling lattice-based post-quantum cryptography on the opentitan platform. Journal of Cryptographic Engineering, 15, 04 2025.

[302] George Tasopoulos, Jinhui Li, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. Performance evaluation of post-quantum tls 1.3 on resource-constrained embedded systems. In Chunhua Su, Dimitris Gritzalis, and Vincenzo Piuri, editors, Information Security Practice and Experience, pages 432–451, Cham, 2022. Springer International Publishing.

[303] The OpenSSL Project. Openssl: The open source toolkit for ssl/tls. www.openssl.org, April 2003.

[304] Emily Stark Thomson et al. Advancing our amazing bet on asymmetric cryptography, May 2024.

[305] Trusted Computing Group. What is post quantum cryptography, and how is tcg implementing it?, August 2025. Accessed 2026-02-27.

[306] Alexander Wagner, Felix Oberhansl, and Marc Schink. To be, or not to be stateful: Post-quantum secure boot using hash-based signatures. In Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security, ASHES'22, page 85–94, New York, NY, USA, 2022. Association for Computing Machinery.

[307] Ruize Wang, Kalle Ngo, Joel Gärtner, and Elena Dubrova. Single-trace side-channel attacks on CRYSTALS-dilithium: Myth or reality? Cryptology ePrint Archive, Paper 2023/1931, 2023.

[308] Tengfei Wang, Chi Zhang, Pei Cao, and Dawu Gu. Efficient implementation of dilithium signature scheme on fpga soc platform. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 30(9):1158–1171, 2022.

[309] Paul Webster, Lucas Berent, Omprakash Chandra, Evan T. Hockings, Nouédyn Baspin, Felix Thomsen, Samuel C. Smith, and Lawrence Z. Cohen. The pinnacle architecture: Reducing the cost of breaking rsa-2048 to 100 000 physical qubits using quantum ldpc codes, 2026.

[310] Daan Weller and Ruud Schellekens. Incorporating post-quantum cryptography in a microservice environment. 2020.

[311] Carolyn Whitnall and Elisabeth Oswald. A critical analysis of iso 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). IACR Cryptology ePrint Archive, Report 2019/1013, 2019.

[312] wolfSSL Inc. Support for nist post-quantum standards ml-kem and ml-dsa. https://www.wolfssl.com/support-for-the-official-post-quantum-standards-ml-kem-and-ml-dsa/, 2024.

[313] IAM World. Post quantum oauth2.x. June 2025.

[314] P. Wouters, T. Pauly, et al. Multiple key exchanges in the internet key exchange protocol version 2 (ikev2). RFC 9370, December 2023.

[315] Kang Yang, Guohua Wu, Chengcheng Dong, Xingbing Fu, Fagen Li, and Ting Wu. Attribute based encryption with efficient revocation from lattices. Int. J. Netw. Secur., 22:161–170, 2020.

[316] Niao Yang, Shaojun Yang, Yong Zhao, and Wei Wu. Inner product encryption from middle-product learning with errors. In International Symposium on Security and Privacy in Social Networks and Big Data, pages 94–113. Springer, 2022.

[317] Mahmoud Yehia, Riham AlTawy, and T. Aaron Gulliver. Gmmt: A revocable group merkle multi-tree signature scheme. In Cryptology and Network Security: 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings, page 136–157, Berlin, Heidelberg, 2021. Springer-Verlag.

[318] Mahmoud Yehia, Riham AlTawy, and T. Aaron Gulliver. Security analysis of DGM and GM group signature schemes instantiated with XMSS-t. Cryptology ePrint Archive, Paper 2021/1496, 2021.

[319] Shiduo Zhang, Xiuhan Lin, Yang Yu, and Weijia Wang. Improved power analysis attacks on falcon. Cryptology ePrint Archive, Paper 2023/224, 2023.

[320] Yanhua Zhang, Ximeng Liu, Yupu Hu, Qikun Zhang, and Huiwen Jia. Attribute-based signatures for inner-product predicate from lattices. In Jaideep Vaidya, Xiao Zhang, and Jin Li, editors, Cyberspace Safety and Security, pages 173–185, Cham, 2019. Springer International Publishing.

[321] Cankun Zhao, Neng Zhang, Hanning Wang, Bohan Yang, Wenping Zhu, Zhengdong Li, Min Zhu, Shouyi Yin, Shaojun Wei, and Leibo Liu. A compact and high-performance hardware architecture for crystals-dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(1):270–295, Nov. 2021.

[322] Yifan Zhao, Ruiqi Xie, Guozhu Xin, and Jun Han. A high-performance domain-specific processor with matrix extension of risc-v for module-lwe applications. IEEE Transactions on Circuits and Systems I: Regular Papers, 69(7):2871–2884, 2022.

[323] Ziyu Zhao, Jintai Ding, and Bo-Yin Yang. Sieving with streaming memory access. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2025(2):362–384, 2025.

[324] Zhen Zhou, Debiao He, Zhe Liu, Min Luo, and Kim-Kwang Raymond Choo. A software/hardware co-design of crystals-dilithium signature scheme. ACM Trans. Reconfigurable Technol. Syst., 14(2), June 2021.

Keep in touch

PiQASO

ECCC EUROPEAN CYBERSECURITY COMPETENCE CENTRE

PiQASO Linkedin

PiQASO Facebook

PiQASO Youtube

ECCC Newsletter

ECCC LinkedIn

ECCC Twitter/X

ECCC Website

EUROPEAN PARTNERSHIP

Co-funded by the European Union

ECCC EUROPEAN CYBERSECURITY COMPETENCE CENTRE