IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/46SP/2025/4629825/
Vol. 2025 (37), Article ID 4629825

The Fujisaki–Okamoto Transform in the Context of Quantum Threats*

Alicja SALAMON and Mariusz JURKIEWICZ

Military University of Technology, 2 Gen. S. Kaliski St., Warsaw, Poland

Correspondence should be addressed to: Alicja SALAMON, alicja.salamon@wat.edu.pl

* Presented at the 46th IBIMA International Conference, 26-27 November 2025, Ronda, Spain

Abstract

This work examines the Fujisaki–Okamoto (FO) transform, the standard technique for upgrading a public key encryption scheme secure only against chosen plaintext attacks to one achieving full chosen ciphertext security (CCA2). The FO transform has been central to the design of key encapsulation mechanisms in the NIST Post Quantum Cryptography competition, and its security has been established in the classical Random Oracle Model (ROM). A crucial challenge arises in the post quantum setting, since the ROM differs fundamentally from the Quantum Random Oracle Model (QROM), where adversaries can issue superposition queries. This subtle distinction, often overlooked outside a narrow circle of experts, calls into question the validity of classical proofs when extended to quantum capable adversaries. Our study addresses this gap by analyzing the FO transform explicitly within the QROM framework.

On the practical side, we implement and evaluate two variants of the CRYSTALS-Kyber scheme: the CPA secure encryption scheme and the CCA secure key encapsulation mechanism derived via the FO transform. The experiments demonstrate that while the transform introduces negligible overhead in key generation and encapsulation, it substantially increases the cost of decapsulation due to the additional re encryption step. These results highlight the trade off between achieving strong CCA2 security and maintaining efficiency. As future work, we aim to explore refinements of CRYSTALS-Kyber that achieve full compatibility with the QROM, thereby contributing to the development of encryption schemes that combine rigorous post quantum security with practical performance.

Keywords: Fujisaki-Okamoto transform, quantum computational model, post-quantum cryptography

Introduction

Modern cryptography is at a critical juncture. For decades, the security of widely deployed public key systems has rested on the presumed intractability of certain number theoretic problems, most prominently integer factorization and discrete logarithms in both finite fields and elliptic curve groups. These hardness assumptions have withstood classical attacks and, as a consequence, have enabled the design of practical, efficient, and trusted cryptographic standards. However, the rapid progress in quantum computation challenges this long-standing foundation. Shor's algorithm demonstrates that a sufficiently powerful quantum computer would solve both integer factorization and discrete logarithms in polynomial time, thereby rendering RSA, Diffie–Hellman, and elliptic curve cryptography insecure.

Cite this Article as: Alicja SALAMON and Mariusz JURKIEWICZ, Vol. 2025 (37) "The Fujisaki–Okamoto Transform in the Context of Quantum Threats " Communications of International Proceedings, Vol. 2025 (37), Article ID 4629825, https://doi.org/10.5171/2025.4629825

Although large-scale, fault-tolerant quantum computers have not yet been realized, research in quantum engineering is advancing at an unprecedented pace. Leading technology companies and research consortia are investing heavily in the development of scalable architectures, error correction techniques, and quantum algorithms. This trajectory makes it prudent to anticipate the eventual deployment of quantum devices capable of breaking today's classical public key infrastructure. The prospect of a "cryptographic transition period" is particularly concerning, since adversaries may already be recording encrypted communications today in anticipation of future decryption once quantum resources become available, an attack scenario commonly referred to as harvest now, decrypt later.

In view of this threat, the research community has turned toward post-quantum cryptography, namely, cryptographic schemes that can withstand adversaries equipped with quantum computation. The U.S. National Institute of Standards and Technology (NIST) has recognized the urgency of this transition, initiating a multi-year standardization process aimed at identifying, analyzing, and eventually adopting quantum-resistant algorithms . The focus has largely been on problems believed to be resistant to quantum speedups, such as those arising from lattices, codes, multivariate polynomials, and hash-based constructions. Among these, lattice-based cryptography has emerged as a particularly strong candidate, offering both efficiency and conjectured hardness in the face of quantum algorithms.

Within this landscape, the question of how to achieve strong security guarantees in the quantum setting remains central. In public key encryption, indistinguishability under chosen ciphertext attack (CCA) is widely accepted as the appropriate security target, since it provides robustness against powerful adversaries who may interact adaptively with decryption oracles. However, many post-quantum proposals achieve security only under chosen plaintext attacks (CPA). To bridge this gap, the Fujisaki–Okamoto (FO) transform provides a generic and well-studied methodology for upgrading CPA secure schemes to CCA security.

A further point about the Fujisaki–Okamoto transform deserves emphasis. The transform is often described as assumption neutral. It is a black box compiler that upgrades chosen plaintext security to chosen ciphertext security while remaining agnostic to the specific hardness assumption that underlies the base encryption scheme. In particular, the construction does not rely on the algebraic details of factorization, discrete logarithms, or lattice problems, and at first glance it appears unrelated to the quantum threat. This intuition, however, is misleading. The subtlety arises not from the computational assumption, but from the proof methodology itself. The transform fundamentally exploits the random oracle model in its classical form. The security proof programs and samples an ideal hash function and reasons about an adversary that submits classical queries to this oracle.

In the presence of a quantum adversary, the abstraction changes in a material way. The quantum random oracle model allows queries in superposition, which alters both what the adversary can learn and how a reduction can control the oracle. Classical proof techniques that are routine for Fujisaki–Okamoto, such as lazy sampling at single points, classical reprogramming of the oracle after the fact, and black box rewinding that assumes classical transcripts, do not directly carry over when the adversary can make coherent oracle queries. It has been established that the classical random oracle model and its quantum analogue differ in their power and in the admissible proof techniques. As a consequence, a proof that establishes security in the classical model does not automatically imply security against quantum adversaries.

The practical implication is clear. Even though the transform is independent of the particular computational assumption and even though its syntax looks unchanged in the quantum era, one must re-examine its security in a model that explicitly accounts for quantum access to the oracle. The literature has made progress by developing quantum-aware proof tools, for example measure-and-reprogram arguments and carefully structured decryption checks that limit how information can leak through the oracle. Yet a completely general and assumption-free theorem that covers all natural instantiations of Fujisaki–Okamoto in the quantum random oracle model remains out of reach. For this reason, any claim that a concrete construction achieves chosen ciphertext security in the presence of quantum adversaries should rest on an analysis within the quantum random oracle model, or on a security reduction that avoids oracle programmability altogether.

The present work contributes to this discussion by examining the Fujisaki–Okamoto transform both theoretically and practically as a tool for strengthening post-quantum encryption schemes. From a theoretical perspective, we investigate its status in the classical random oracle model (ROM) as well as in the quantum random oracle model (QROM), which more accurately captures the capabilities of quantum adversaries. While security in the ROM has been established, the situation in the QROM remains unsettled: a general proof of security is lacking, and only certain special cases have been analyzed. From a practical perspective, we assess the computational impact of applying the FO transform to a representative lattice-based scheme, CRYSTALS-

KYBER, which has been selected as part of the NIST post-quantum standardization effort. Our implementation compares CPA secure and CCA secure variants, with the goal of isolating the concrete efficiency costs attributable to the FO transform. The scheme itself is not the main subject of investigation; rather, it serves as a testbed to explore the trade-off between theoretical rigor and practical efficiency in the design of post-quantum secure systems.

Preliminaries

The LWE and MLWE Problems

One of the fundamental challenges in modern post-quantum cryptography is the **Learning With Errors** problem (LWE). This problem generalizes earlier constructions based on SIS and ISIS and plays a central role in the design of many post-quantum cryptographic schemes.

In its general form, an LWE instance is defined as follows. For a fixed matrix $A \in \mathbb{Z}_q^{n \times m}$, an unknown secret vector $s \in \mathbb{Z}_q^m$, and a random noise vector $e \in \mathbb{Z}^n$, one obtains

$$y \equiv As + e \pmod{q}.$$

The vector e introduces small errors, typically sampled from a discrete Gaussian or a discrete binomial distribution. Although the equation is linear, the added noise makes recovering s computationally intractable. *Computational variant.* In the computational version, the adversary is given a set of independent samples (A_i, y_i), each satisfying $y_i \equiv A_i s + e_i \pmod{q}$. The goal is to recover the secret vector s. *Decisional variant.* In the decisional version, the task is to distinguish between samples generated according to the above distribution and uniformly random pairs, where y is chosen uniformly at random from $\mathbb{Z}_q^n$.

Generalization: the MLWE problem. A widely used extension of LWE in practical post-quantum schemes (e.g., KYBER) is the **Module Learning With Errors** problem (MLWE). Here, given a parameter $k \in \mathbb{N}$, a ring R, the problem is to distinguish pairs

$$(\mathbf{A}, \mathbf{t} = \mathbf{As} + \mathbf{e}) \in R^{k \times k} \times R^k,$$

where $\mathbf{s}, \mathbf{e} \in R^k$ are secret and error vectors, from uniformly random pairs $(\mathbf{A}, \mathbf{t})$ sampled from $R^{k \times k} \times R^k$. In this work, we consider the polynomial space $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ where $\mathbb{Z}_q$ is a ring.

The advantage of MLWE is that it retains the hardness of LWE while improving efficiency and compactness of representation, which makes it especially suitable for practical cryptographic schemes.

Reduction from MLWE to LWE. It is important to note that MLWE generalizes LWE and can be reduced to it. Specifically, an MLWE instance can be transformed into a larger LWE instance by expanding the coefficients in R_q into coefficient vectors over $\mathbb{Z}_q^n$. Consequently, if there exists an efficient algorithm for solving LWE for certain parameters, it can also be applied to solve MLWE with appropriately scaled parameters.

Conversely, the hardness of MLWE inherits from that of LWE. The absence of efficient algorithms for LWE implies that MLWE instances with suitable parameters are also hard. From a practical perspective, this enables the construction of more efficient schemes while preserving the same security foundation.

Random Oracle Model

The Random Oracle Model (ROM) is a widely used abstraction in cryptography, introduced to facilitate security proofs that would be difficult or even intractable to carry out in the standard model. In this framework, a hash function H is idealized as a truly random function, accessible only through queries to an oracle. For each fresh query, the oracle returns a value chosen uniformly at random from its range, while ensuring consistency for repeated queries. The oracle can therefore be regarded as a "black box" mechanism: its internal structure is hidden, but its responses are consistent and appear unpredictable.

The strength of the ROM lies in its ability to enable security reductions under the strong assumption that H behaves as a perfect source of randomness. Proofs in this model typically rely on three fundamental properties of the oracle:

1. **Uniformity:** until a query x is made, the value $\mathsf{H}(x)$ is completely undetermined and distributed uniformly at random.

2. **Consistency:** once $\mathsf{H}(x)$ has been defined, the same value is always returned for subsequent queries on x.

3. **Programmability:** in reduction based proofs, the simulator can assign values to $\mathsf{H}(x)$ provided the resulting distribution remains consistent with uniformity.

These properties make the ROM a versatile and powerful tool for the design and analysis of cryptographic primitives. At the same time, the model has inherent limitations. In practice, the random oracle must be instantiated with a concrete hash function, and no real function can perfectly emulate the behavior of a true random oracle. Consequently, it is not guaranteed that security proven in the ROM always translates to the standard model. Indeed, there exist constructions that are provably secure in the ROM but demonstrably insecure under every instantiation with an actual hash function.

Despite these shortcomings, the ROM has been extremely influential. It is best viewed as a heuristic framework: security in the ROM provides strong evidence that a scheme has no obvious structural weaknesses, even if it does not constitute a proof in the strictest sense. For this reason, the ROM continues to serve as the foundation for the security analysis of many widely deployed protocols, including signature schemes and encryption systems, and remains one of the most important tools in modern cryptographic practice.

Quantum Random Oracle Model

Most modern cryptographic schemes are proven secure in the classical ROM, where the adversary is restricted to queries consisting of classical inputs. Concretely, given access to a random oracle H, the adversary can only evaluate $y = \mathsf{H}(x)$ for some classical input $x \in \{0,1\}^n$.

In practice, however, the random oracle is instantiated by a fixed hash function that is known both to the honest participants of the protocol and to the adversary. Within this setting, a quantum adversary gains a decisive advantage: the ability to query the oracle on a superposition of inputs rather than on a single classical string. This capability fundamentally distinguishes the quantum setting from the classical one. Formally, a quantum query is represented by the state

$$\sum_{x\in\{0,1\}^n} a_x \,|x\rangle|0\rangle,$$

where the amplitudes $a_x \in \mathbb{C}$ satisfy the normalization condition

$$\sum_{x\in\{0,1\}^n} |\,a_x|^2 = 1.$$

The quantum oracle implements the unitary transformation

$$|x\rangle|y\rangle \mapsto |x\rangle|y \oplus \mathsf{H}(x)\rangle,$$

so that its response to the superposition query is

$$\sum_{x\in\{0,1\}^n} a_x \,|x\rangle|\mathsf{H}(x)\rangle.$$

In this way, the adversary obtains a coherent superposition of all pairs $(x, \mathsf{H}(x))$, thereby accessing information that would require exponentially many queries in the classical model. It is essential to stress, however, that the cryptographic scheme itself remains entirely classical: both the interaction between protocol participants and the formulation of their queries are restricted to classical bit strings. The quantum capability lies exclusively with the adversary's access to the oracle.

Security Proofs in the QROM

Proof techniques in the QROM often rely on *quantum-accessible pseudorandom functions* (QPRFs). These allow an efficient simulation of a quantum random oracle while maintaining security against a quantum-enabled

adversary. The approach assumes the existence of a QPRF, which can be used by the simulator to construct a reduction that emulates the random oracle in a way that is indistinguishable from real quantum oracle access.

A **quantum-accessible pseudorandom function** is an efficiently computable function satisfying the following condition for all efficient quantum algorithms $\mathcal{D}$:

$$|\Pr[\mathcal{D}^{\mathsf{QPRF}}(1^n) = 1] - \Pr[\mathcal{D}^{\mathsf{H}}(1^n) = 1]| \leq \mathsf{negl}(n).$$

One method of constructing such functions for use in security proofs is based on **t-wise independent functions**. A function $f: \{0,1\}^n \to \{0,1\}^n$ is called **t-wise independent** if, for t distinct inputs $x_1, x_2, \dots, x_t$, the outputs $f(x_1), f(x_2), \dots, f(x_t)$ are mutually independent and uniformly distributed.

The simplest example of such a function is a polynomial of degree $t-1$:

$$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1},$$

where $a_i \in GF(2^n), i \in \{0, \dots, t-1\}$. (Zhandry, 2012) showed that $2q$-wise independent functions are indistinguishable from a truly random oracle by quantum algorithms making at most q queries. Below we recall two key lemmas commonly used in QROM proofs, which will later be applied in the analysis of the modified Fujisaki–Okamoto transform:

Lemma 1 (O2H, *One Way to Hiding Lemma*, see (Unruh, 2014)). *Let* $\mathsf{H}: \{0,1\}^n \to \{0,1\}^m$ *be a random oracle. Consider an algorithm* $\mathcal{A}$ *making at most* q_{H} *queries to* H. *Define:*

$$P_{\mathcal{A}}^1 = \Pr\left[\mathrm{b}' = 1 : x \xleftarrow{\$} \{0,1\}^n, \mathrm{b}' \leftarrow \mathcal{A}^{\mathsf{H}}\big(x, \mathsf{H}(x)\big)\right],$$
$$P_{\mathcal{A}}^2 = \Pr\left[\mathrm{b}' = 1 : x \xleftarrow{\$} \{0,1\}^n, y \xleftarrow{\$} \{0,1\}^m, \mathrm{b}' \leftarrow \mathcal{A}^{\mathsf{H}}(x, y)\right],$$

where $\mathcal{A}^{\mathsf{H}}$ *indicates that* $\mathcal{A}$ *may query* H.

Now consider an algorithm $\mathcal{B}$ *which, given input* x, *operates as follows:*

- *choose* $i \xleftarrow{\$} \{1, \dots, q_{\mathsf{H}}\}$,
- *choose* $y \xleftarrow{\$} \{0,1\}^m$,
- *provide* x, y *to* $\mathcal{A}^{\mathsf{H}}(x, y)$,
- *just before the* i*-th query of* $\mathcal{A}$ *to* H, *measure the query register and record* x',
- *if* $\mathcal{A}$ *makes fewer than* i *queries, output* $\perp \notin \{0,1\}^n$,
- *define the event that* $\mathcal{B}$ *wins if* $x = x'$, *with probability*

$$P_{\mathcal{B}} = \Pr\left[x = x' : x \xleftarrow{\$} \{0,1\}^n, x' \leftarrow \mathcal{B}(x)\right].$$

Then,

$$|P_{\mathcal{A}}^1 - P_{\mathcal{A}}^2| \leq 2 q_{\mathsf{H}} \sqrt{P_{\mathcal{B}}}.$$

This lemma is analogous to the use of the event in the classical ROM, where an adversary queries the oracle at a critical point x. In the quantum setting, the measurement performed by $\mathcal{B}$ plays a similar role, and bounding $P_{\mathcal{B}}$ allows us to control the difference $|P_{\mathcal{A}}^1 - P_{\mathcal{A}}^2|$.

Lemma 2 (Adaptive O2H, *One Way to Hiding Lemma, Adaptive*, see (Unruh, 2014)). *Let* $\mathsf{H}: \{0,1\}^* \to \{0,1\}^n$ *be a random oracle. Consider an algorithm* $\mathcal{A}_0$ *making at most* q_0 *queries to* H *and an algorithm* $\mathcal{A}_1$, *which uses the final state of* $\mathcal{A}_0$ *and makes at most* q_1 *queries to* H. *Define:*

$$P_{\mathcal{A}}^{1} = \Pr\left[\mathrm{b}' = 1 : \mathrm{m} \leftarrow \mathcal{A}_0^{\mathsf{H}}, x \xleftarrow{\$} \{0,1\}^{l}, \mathrm{b}' \leftarrow \mathcal{A}_1^{\mathsf{H}}(x, \mathsf{H}(x \parallel \mathrm{m}))\right],$$
$$P_{\mathcal{A}}^{2} = \Pr\left[\mathrm{b}' = 1 : \mathrm{m} \leftarrow \mathcal{A}_0^{\mathsf{H}}, x \xleftarrow{\$} \{0,1\}^{l}, \mathsf{C} \xleftarrow{\$} \{0,1\}^{n}, \mathrm{b}' \leftarrow \mathcal{A}_1^{\mathsf{H}}(x, \mathsf{C})\right].$$

Now define an algorithm $\mathcal{B}$, which, given (j, C, x), runs $\mathcal{A}_1^{\mathsf{H}}(x, \mathsf{C})$ and measures the query register just before the j-th query of $\mathcal{A}_1$ to H*, returning $x' \parallel \mathrm{m}'$. If $\mathcal{A}_1$ makes fewer than j queries, $\mathcal{B}$ outputs $\perp \notin \{0,1\}^{l}$. Define the event $x = x' \wedge \mathrm{m} = \mathrm{m}'$ with probability*

$$P_{\mathcal{B}} = \Pr[x = x' \wedge \mathrm{m} = \mathrm{m}' : \mathrm{m} \leftarrow \mathcal{A}_0^{\mathsf{H}}, x \xleftarrow{\$} \{0,1\}^{l}, \mathsf{C} \xleftarrow{\$} \{0,1\}^{n}, j \xleftarrow{\$} \{1, \ldots, q_1\}, x' \parallel \mathrm{m}' \leftarrow \mathcal{B}^{\mathsf{H}}(j, \mathsf{C}, x)],$$

then

$$|P_{\mathcal{A}}^{1} - P_{\mathcal{A}}^{2}| \leq 2q_1\sqrt{P_{\mathcal{B}}} + q_0 \cdot 2^{-l/2+2}.$$

The Fujisaki–Okamoto Transform in the ROM

Hybrid Encryption Scheme

The Fujisaki–Okamoto transform, introduced in the following subsection, is an example of a **hybrid encryption scheme**. Such schemes combine both asymmetric and symmetric cryptography, thereby leveraging the advantages of each approach. Hybrid encryption techniques are widely used, as they naturally address the inherent limitations of the individual paradigms.

Asymmetric schemes, while providing a high level of security, are computationally expensive, making them inefficient for encrypting long messages. On the other hand, symmetric schemes are significantly faster but require a secure key exchange between the communicating parties. Hybrid schemes mitigate these drawbacks by integrating both methods into a single construction.

The simplest example of a hybrid scheme is a construction that combines an asymmetric encryption scheme $\mathcal{B}_{\mathsf{asy}}$ and a symmetric encryption scheme $\mathcal{B}_{\mathsf{sym}}$, where the key spaces satisfy $\mathcal{M}^{\mathsf{asy}}=\mathcal{K}^{\mathsf{sym}}$. The resulting hybrid scheme $\mathcal{B}_{\mathsf{hy}} = (\mathsf{KGen}^{\mathsf{hy}},\mathsf{Enc}^{\mathsf{hy}},\mathsf{Dec}^{\mathsf{hy}})$ is defined as follows:

- $\mathsf{KGen}^{\mathsf{hy}}(1^n)$:
    1. $(\mathsf{sk}^{\mathsf{asy}},\mathsf{pk}^{\mathsf{asy}}) \xleftarrow{\$} \mathsf{KGen}^{\mathsf{asy}}(1^n)$,
    2. $\mathsf{sk} \leftarrow \mathsf{sk}^{\mathsf{asy}}$, $\mathsf{pk} \leftarrow \mathsf{pk}^{\mathsf{asy}}$,
    3. output the key pair $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{hy}}(1^n)$.
- $\mathsf{Enc}^{\mathsf{hy}}(\mathsf{pk},\mathsf{m})$:
    1. $\mathsf{k} \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$,
    2. $\mathsf{c} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$,
    3. $\mathsf{e} \xleftarrow{\$} \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\mathsf{k})$,
    4. output the ciphertext $(\mathsf{c}, \mathsf{e}) \leftarrow \mathsf{Enc}^{\mathsf{hy}}(\mathsf{m}, \mathsf{pk})$.
- $\mathsf{Dec}^{\mathsf{hy}}(\mathsf{sk}, (\mathsf{c}, \mathsf{e}))$:
    1. $\mathsf{k}' \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{e})$,
    2. $\mathsf{m}' \leftarrow \mathsf{Dec}^{\mathsf{sym}}(\mathsf{k}, \mathsf{c})$,
    3. output the message $m' \leftarrow \mathsf{Dec}^{\mathsf{hy}}(\mathsf{sk}, (\mathsf{c}, \mathsf{e}))$.

The commonly accepted security standard in modern public-key encryption is chosen ciphertext security (CCA). In this context, a significant limitation of the above construction becomes apparent. For the hybrid scheme $\mathcal{B}_{\mathsf{hy}}$ to achieve CCA security, it is necessary that both $\mathcal{B}_{\mathsf{asy}}$ and $\mathcal{B}_{\mathsf{sym}}$ are themselves CCA-secure. This requirement substantially restricts the range of feasible constructions.

The Fujisaki–Okamoto Transform

Consider a scenario where we have an asymmetric encryption scheme that is secure against chosen plaintext attacks (CPA), and a symmetric encryption scheme that provides semantic security. Applying the Fujisaki–Okamoto transform enables the construction of a hybrid scheme that achieves chosen ciphertext security (CCA) in the random oracle model.
Let the asymmetric encryption scheme be defined as $\mathcal{B}_{\mathsf{asy}} = (\mathsf{KGen}^{\mathsf{asy}}, \mathsf{Enc}^{\mathsf{asy}}, \mathsf{Dec}^{\mathsf{asy}})$ over $(\mathcal{M}^{\mathsf{asy}}, \mathcal{C}^{\mathsf{asy}}, \mathcal{R}^{asy})$, the symmetric encryption scheme as $\mathcal{B}_{\mathsf{sym}} = (\mathsf{KGen}^{\mathsf{sym}}, \mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ over $(\mathcal{K}^{\mathsf{sym}}, \mathcal{M}^{\mathsf{sym}}, \mathcal{C}^{\mathsf{sym}})$, and let there be two hash functions:

- $\mathsf{G}: \mathcal{M}^{\mathsf{asy}} \to \mathcal{K}^{\mathsf{sym}}$,
- $\mathsf{H}: \mathcal{M}^{\mathsf{asy}} \times \mathcal{C}^{\mathsf{sym}} \to \mathcal{R}^{asy}$.

We construct the hybrid scheme $\mathcal{B}_{\mathsf{fo}} = (\mathsf{KGen}^{\mathsf{fo}}, \mathsf{Enc}^{\mathsf{fo}}, \mathsf{Dec}^{\mathsf{fo}})$ over $(\mathcal{M}^{\mathsf{fo}}, \mathcal{C}^{\mathsf{fo}})$ as follows:

- $\mathsf{KGen}^{\mathsf{fo}}(1^n)$ is the key generation algorithm for producing an asymmetric key pair $(\mathsf{pk},\mathsf{sk})$, given the security parameter. The algorithm proceeds as follows:
    1. $(\mathsf{sk}^{\mathsf{asy}}, \mathsf{pk}^{\mathsf{asy}}) \leftarrow \mathsf{KGen}^{\mathsf{asy}}(1^n)$,
    2. $\mathsf{sk} \leftarrow \mathsf{sk}^{\mathsf{asy}}$, $\mathsf{pk} \leftarrow \mathsf{pk}^{\mathsf{asy}}$,
    3. output the key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{fo}}(1^n)$.
- $\mathsf{Enc}^{\mathsf{fo}}(\mathsf{pk}, \mathsf{m})$ is the encryption algorithm, which takes as input a public key pk and a message $\mathsf{m} \in \mathcal{M}^{\mathsf{fo}} (:= \mathcal{M}^{\mathsf{sym}})$. The algorithm proceeds as follows:
    1. $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$,
    2. $\mathsf{k} \leftarrow \mathsf{G}(\sigma)$,
    3. $\mathsf{c} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$,
    4. $\mathsf{h} \leftarrow \mathsf{H}(\sigma, \mathsf{c})$,
    5. $\mathsf{e} \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma; \mathsf{h})$,
    6. output the ciphertext $(\mathsf{e}, \mathsf{c}) \leftarrow \mathsf{Enc}^{\mathsf{fo}}(\mathsf{pk}, \mathsf{m})$.
- $\mathsf{Dec}^{\mathsf{fo}}(\mathsf{sk}, (\mathsf{e}, \mathsf{c}))$ is the decryption algorithm, which takes as input the private key sk and a ciphertext (e, c). The algorithm proceeds as follows:
    1. Parse (e, c) into e and c. If parsing fails, return the error symbol $\perp$.
    2. If $\mathsf{e} \notin \mathcal{C}^{\mathsf{asy}}$ or $\mathsf{c} \notin \mathcal{C}^{\mathsf{sym}}$, return $\perp$.
    3. $\hat{\sigma} \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{e})$,
    4. If $\hat{\sigma} \notin \mathcal{M}^{\mathsf{asy}}$, return $\perp$.
    5. $\hat{\mathsf{k}} \leftarrow \mathsf{G}(\hat{\sigma})$,
    6. $\hat{\mathsf{h}} \leftarrow \mathsf{H}(\hat{\sigma}, c)$,
    7. Verify that $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \hat{\sigma}; \hat{h})$. Otherwise, return $\perp$.

8. $y \leftarrow \mathsf{Dec}^{\mathsf{sy}}(\hat{\mathsf{k}}, c)$,

9. output the message $y \leftarrow \mathsf{Dec}^{\mathsf{fo}}(\mathsf{sk}, (\mathsf{e}, \mathsf{c}))$.

We now show that the scheme $\mathcal{B}_{\mathsf{fo}}$ satisfies the correctness condition. For a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{fo}}(1^n)$, a message $\mathsf{m} \in \mathcal{M}^{\mathsf{sym}}$, and a valid ciphertext (e, c) such that it can be parsed into $\mathsf{e} \in \mathcal{C}^{\mathsf{asy}}$ and $\mathsf{c} \in \mathcal{C}^{\mathsf{sym}}$, we have:

$$\mathsf{c} = \mathsf{Enc}^{\mathsf{sym}}(\mathsf{G}(\sigma), \mathsf{m}) \text{ and } \mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma; \mathsf{H}(\sigma, \mathsf{c})).$$

Decrypting $\sigma' \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{e})$, by the correctness of the scheme $\mathcal{B}_{\mathsf{asy}}$, we obtain:

$$\sigma' = \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma; \mathsf{H}(\sigma, \mathsf{c}))) = \sigma.$$

Consequently, since $\sigma' = \sigma$ implies $\mathsf{G}(\sigma') = \mathsf{G}(\sigma)$, decrypting $\mathsf{m}' \leftarrow \mathsf{Dec}^{\mathsf{sym}}(\mathsf{G}(\sigma), \mathsf{c})$ yields, by the correctness of the scheme $\mathcal{B}_{\mathsf{sym}}$:

$$\mathsf{m}' = \mathsf{Dec}^{\mathsf{sym}}(\mathsf{G}(\sigma), \mathsf{Enc}^{\mathsf{sym}}(\mathsf{G}(\sigma), \mathsf{m})) = \mathsf{m}.$$

Security Proof of the Fujisaki–Okamoto Scheme

Let $\mathcal{B}_{\mathsf{asy}} = (\mathsf{KGen}^{\mathsf{asy}}, \mathsf{Enc}^{\mathsf{asy}}, \mathsf{Dec}^{\mathsf{asy}})$ be an asymmetric encryption scheme defined over $(\mathcal{M}^{\mathsf{asy}}, \mathcal{C}^{\mathsf{asy}}, \mathcal{R}^{\mathsf{asy}})$ that is secure in the CPA model. Let $\mathcal{A}_{\mathsf{asy}}$ be a PPT adversary attacking $\mathcal{B}_{\mathsf{asy}}$ in this model, i.e.,

$$\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n) \leq \mathsf{negl}(n).$$

Assume further that $\mathcal{B}_{\mathsf{sym}} = (\mathsf{KGen}^{\mathsf{sym}}, \mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ is a symmetric encryption scheme defined over $(\mathcal{K}^{\mathsf{sym}}, \mathcal{M}^{\mathsf{sym}}, \mathcal{C}^{\mathsf{sym}})$ that satisfies semantic security. Hence the advantage of any adversary $\mathcal{A}_{\mathsf{sym}}$ is negligible:

$$\mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}}, \mathcal{B}_{\mathsf{sym}}}(n) \leq \mathsf{negl}(n).$$

Theorem 1. *Let* $\mathcal{E}^{\mathsf{fo}} = (\mathsf{KGen}^{\mathsf{fo}}, \mathsf{Enc}^{\mathsf{fo}}, \mathsf{Dec}^{\mathsf{fo}})$ *over* $(\mathcal{K}^{\mathsf{fo}}, \mathcal{M}^{\mathsf{fo}}, \mathcal{C}^{\mathsf{fo}})$ *be the hybrid scheme obtained by applying the Fujisaki–Okamoto transform to* $\mathcal{B}_{\mathsf{asy}}$ *and* $\mathcal{B}_{\mathsf{sym}}$. *Under the above assumptions, there exists an adversary* $\mathcal{A}_{\mathsf{fo}}$ *allowed to make* q_{H} *queries to the random oracles and* q_{Dec} *queries to the decryption oracle such that its advantage satisfies* $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}}, \mathcal{E}^{\mathsf{fo}}}(n) = 2 \cdot q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n) + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}}, \mathcal{B}_{\mathsf{sym}}}(n) + 2 \cdot q_{\mathsf{Dec}} \cdot 2^{-\gamma}$.

The original security proof of the Fujisaki-Okamoto hybrid scheme can be found in (Fujisaki & Okamoto, 1999). However, due to several modifications in this construction, we provide a complete proof here, as it differs from the original version.

Proof. The proof proceeds via a standard sequence-of-games argument: we define games $G_1, G_2, \ldots, G_l$, all over the same probability space, and move from one game to the next by changing only how queries to the oracles are answered.

Remarks.

- In each game, the challenger (in G_0) or simulator (in subsequent games) samples a bit $\mathsf{b} \in \{0,1\}$, and the adversary outputs a bit $\mathsf{b}' \in \{0,1\}$ at the end.

- Variables sampled by the simulator are marked with the superscript "*". The challenge returned by the simulator to the adversary (i.e., the output of $\mathsf{Enc}^{\mathsf{fo}}$) is denoted $(\mathsf{e}^*, \mathsf{c}^*)$, where

$$\mathsf{c}^* = \mathsf{Enc}^{\mathsf{sym}}(\mathsf{G}(\sigma^*), \mathsf{m}_{\mathsf{b}}) \quad \text{and} \quad \mathsf{e}^* = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*, \mathsf{H}(\sigma^*, \mathsf{c}^*)).$$

- S_i, for $0 \leq i \leq l$, is the event that in game G_i the equality $b' = b$ holds (i.e., the adversary wins).

- $\mathsf{ASK}_{\sigma_i^*}$, for $0 \leq i \leq l$, is the event in game G_i that either

  1. $\mathcal{A}_{\mathsf{fo}}$ makes a query σ^* to the random oracle G, *or*

  2. $\mathcal{A}_{\mathsf{fo}}$ makes a query (σ^*, c^*) to the random oracle H.

Game G_0

Game G_0 is the baseline CCA attack game. Its flow against $\mathcal{E}^{\mathsf{fo}}$ is:

1. The challenger generates a key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}^{\mathsf{fo}}.(1^n)$.
2. The adversary receives pk and access to the oracles G, H, and $\mathsf{Dec}^{\mathsf{fo}}$.
3. The adversary submits a pair of messages $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}^{\mathsf{fo}}$.
4. The challenger samples $\mathsf{b} \xleftarrow{\$} \{0,1\}$ and encrypts, producing the challenge $\mathsf{e}^* \parallel \mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{fo}}(\mathsf{pk}, \mathsf{m}_\mathsf{b})$.
5. The adversary receives $(\mathsf{e}^*, \mathsf{c}^*)$, regains access to G, H and $\mathsf{Dec}^{\mathsf{fo}}$, and finally outputs b'.

By definition of CCA security,

$$\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{E}^{\mathsf{fo}}}(n) = \left|\Pr\left[\mathsf{Exp}^{\mathrm{G}_0}_{\mathcal{E}^{\mathsf{fo}},\mathcal{A}_{\mathsf{fo}}}(1^n) = 1\right] - \frac{1}{2}\right| = \left|\Pr[\mathsf{S}_0] - \frac{1}{2}\right|.$$

Our goal is to upper bound $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{E}^{\mathsf{fo}}}(n)$. In subsequent steps we modify the event S_i and bound the differences $\Pr[\mathsf{S}_{i-1}] - \Pr[\mathsf{S}_i]$. Once $\Pr[\mathsf{S}_l]$ is determined exactly, we obtain the desired upper bound on $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{E}^{\mathsf{fo}}}(n)$.

Game G_1

Game G_1 modifies G_0 by changing the decryption oracle: the simulator does not use the secret key sk.

During the simulation, all queries to G and H, made by both the adversary and the simulator, are recorded together with their responses. We maintain the lists

$$Q_{\mathsf{G}} = [(\sigma_i, \mathsf{k}_i)],$$
$$Q_{\mathsf{H}} = [(\sigma_i, \mathsf{c}_i, \mathsf{H}_i)].$$

The flow of G_1 is as follows:

1. The simulator generates $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}^{\mathsf{fo}}(1^n)$.
2. The adversary receives pk, access to G and H, and to the *simulated* decryption oracle. Upon a decryption query (e, c), the simulator checks whether there is an entry $(\sigma, \mathsf{c}, \mathsf{h})$ in Q_{H} with $\sigma \in \mathcal{M}^{\mathsf{asy}}$ and $\mathsf{e} = \mathsf{Enc}^{\mathsf{fo}}(\mathsf{pk}, \sigma; \mathsf{h})$.
   - If not, it returns $\perp$.
   - Otherwise, it computes $\mathsf{k} = \mathsf{G}(\sigma)$ and $\mathsf{m}' \leftarrow \mathsf{Dec}^{\mathsf{sym}}(\mathsf{k}, \mathsf{c})$.

   If the pair (e, c) has been queried before, the same output as previously is returned.
3. The adversary submits $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}^{\mathsf{fo}}$.
4. The simulator samples $\mathsf{b} \xleftarrow{\$} \{0,1\}$ and forms the challenge $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{fo}}(\mathsf{pk}, \mathsf{m}_\mathsf{b})$.
5. The adversary receives c^* and regains access to G, H and the simulated decryption oracle, then outputs b'.

Define BAD as the event that the adversary submits a decryption query $(\mathsf{e}_{\mathsf{BAD}}, \mathsf{c}_{\mathsf{BAD}})$ for which there is no triple $(\sigma, \mathsf{c}_{\mathsf{BAD}}, \mathsf{h})$ in Q_{H} satisfying the simulation's consistency conditions. In G_0 this is not an issue as long as $(\mathsf{e}_{\mathsf{BAD}}, \mathsf{c}_{\mathsf{BAD}})$ is a valid ciphertext (since the real decryption algorithm is used), whereas in G_1 the response is $\perp$.

From the adversary's perspective, outside of BAD there is no difference between interacting with the real scheme in G_0 and the simulation in G_1. Hence,

$$S_0 \wedge \neg BAD = S_1 \wedge \neg BAD,$$
$$\Pr[S_0 \wedge \neg BAD] = \Pr[S_1 \wedge \neg BAD],$$
$$\Pr[S_0] - \Pr[S_1] \leq \Pr[BAD].$$

Assume $e \in \mathcal{C}^{asy}$, i.e., there exist $\sigma \in \mathcal{M}^{asy}$ and $r \in \mathcal{R}^{asy}$ such that $e = Enc^{asy}(pk, \sigma, r)$, and that the simulator's challenge is $e^* \| c^*$ with $\sigma^* = Dec^{asy}(sk, e^*)$. Then:

1. If $e = e^*$, necessarily $c \neq c^*$ because $(e^*, c^*) \neq (e, c)$ (the adversary may not submit the challenge for decryption),

   or

2. if $e \neq e^*$, then $c \neq c^*$ or $\sigma \neq \sigma^*$; otherwise $H(\sigma, c) = H(\sigma^*, c)$, hence $e = Enc^{asy}(pk, \sigma; H(\sigma, c)) = Enc^{asy}(pk, \sigma^*; H(\sigma^*, c^*)) = e^*$, a contradiction.

Therefore every decryption query satisfies

$$(\sigma, c) \neq (\sigma^*, c^*) \Leftrightarrow \sigma \neq \sigma^* \vee c \neq c^*.$$

Since H is modeled as a random oracle, the answer to a query (σ, c) is independent of the answer to (σ^*, c^*) for any $(\sigma, c) \neq (\sigma^*, c^*)$. The adversary attempts to form a valid ciphertext e by guessing a triple (σ, c, h) such that $e = Enc^{asy}(pk, \sigma; h)$, without first querying H on (σ, c). Under the random-oracle assumption, the value $h = H(\sigma, c)$ is uniformly random from the adversary's viewpoint. Hence,

$$\Pr[e = Enc^{asy}(pk, \sigma, h) \wedge h = H(\sigma, c)] \leq \Pr\left[e = Enc^{asy}(pk, \sigma, h) : h \xleftarrow{\$} \mathcal{R}^{asy}\right].$$

For any pk and any $\sigma \in \mathcal{M}^{asy}$ there are at least 2^{γ} possible ciphertexts of the form $e = Enc^{asy}(pk, \sigma, h)$ as h ranges uniformly. Thus,

$$\Pr\left[e = Enc^{asy}(pk, \sigma, h) : h \xleftarrow{\$} \mathcal{R}^{asy}\right] \leq \frac{q_{Dec}}{2^{\gamma}},$$
$$\Pr[BAD] \leq q_{Dec} \cdot 2^{-\gamma}.$$

Since $\Pr[S_0] - \Pr[S_1] \leq \Pr[BAD]$ and $\Pr[BAD] \leq q_{Dec} \cdot 2^{-\gamma}$, we conclude that

$$\Pr[S_0] - \Pr[S_1] \leq q_{Dec} \cdot 2^{-\gamma}.$$

Game G_2

In game G_2 we modify game G_1 by replacing the values $G(\sigma^*)$ and $H(\sigma^*, c^*)$ with independently sampled elements, denoted a^* and r^*, respectively. This change is purely conceptual, since a^* and r^* have the same distributions as $G(\sigma^*)$ and $H(\sigma^*, c^*)$, while $(a^* \neq G(\sigma^*))$ and $(r^* \neq H(\sigma^*, c^*))$.

- The simulator sends the challenge (e^*, c^*) such that $c^* = Enc^{sym}(a^*, m_b)$, $e^* = Enc^{asy}(pk, \sigma^*; r^*))$.
- If the adversary queries the random oracle G at σ^*, the simulator returns a^*. For a query (σ^*, c^*) to H, it returns r^*.
- This modification can be viewed as replacing the oracles G and H with new oracles G' and H' that return exactly the same answers as the originals for all queries except at the inputs σ^* and (σ^*, c^*).
- The joint distribution of answers under G, H is identical to that under G', H', hence

$$\Pr[S_1] - \Pr[S_2] = 0.$$

Game G_3

In game G_3 we modify game G_2 by reusing the same challenge that the simulator generated for the adversary, namely

$$(e^*, c^*) \text{ such that } c^* = Enc^{sym}(a^*, m_b), \quad e^* = Enc^{asy}(pk, \sigma^*; r^*)),$$

where $\mathsf{r}^* \neq \mathsf{H}(\sigma^*, \mathsf{c}^*)$ and $\mathsf{a}^* \neq \mathsf{G}(\sigma^*)$. However, the modified oracles G', H' are replaced back by the original oracles G, H. Thus, their answers to the queries σ^* and (σ^*, c^*) are no longer $\mathsf{a}^*, \mathsf{r}^*$ but rather $\mathsf{G}(\sigma^*)$ and $\mathsf{H}(\sigma^*, \mathsf{c}^*)$.

Note that from the adversary's perspective, the simulations in G_2 and G_3 are indistinguishable, except when the adversary queries the oracles at σ^* or (σ^*, c^*). Hence, a difference can arise only if the events $\mathsf{ASK}_{\sigma_2^*}$ or $\mathsf{ASK}_{\sigma_3^*}$ occur. We therefore have

$$\mathsf{S}_2 \wedge \neg \mathsf{ASK}_{\sigma_2^*} = \mathsf{S}_3 \wedge \neg \mathsf{ASK}_{\sigma_3^*},$$
$$\mathsf{ASK}_{\sigma_2^*} = \mathsf{ASK}_{\sigma_3^*}.$$

Consequently,

$$\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3] \leq \Pr\left[\mathsf{ASK}_{\sigma_3^*}\right].$$

Game G_4

In game G_4 we modify game G_3 by changing the way random oracle responses are generated. Instead of selecting the random functions G and H in advance, their outputs are determined lazily, at the time of each query.

- Q_{G} is the list of G queries and responses. Initially empty, upon a query σ, a value $\mathsf{a} \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$ is sampled, returned as the response, and recorded in Q_{G} as the pair (σ, a).
- Q_{H} is the list of H queries and responses. Initially empty, upon a query (σ, c), a value $\mathsf{h} \xleftarrow{\$} \mathcal{R}^{\mathsf{asy}}$ is sampled, returned as the response, and recorded in Q_{H} as the triple $(\sigma, \mathsf{c}, \mathsf{h})$.
- If the adversary repeats a query that already appears in the list, the previously stored value is returned.

Since this change is purely conceptual, we have

$$\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3] = 0,$$
$$\Pr\left[\mathsf{ASK}_{\sigma_4^*}\right] - \Pr\left[\mathsf{ASK}_{\sigma_3^*}\right] = 0.$$

Bounding the Advantage of Adversary $\mathcal{A}_{fo}$

We obtain

$$\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_4] \leq q_{\mathsf{Dec}} \cdot 2^{-\gamma} + \Pr\left[\mathsf{ASK}_{\sigma_4^*}\right].$$

We now show that $\Pr\left[\mathsf{ASK}_{\sigma_4^*}\right] \leq q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n)$. The argument proceeds via a reduction: given an efficient adversary $\mathcal{A}_{\mathsf{fo}}$ attacking the CCA security of $\mathcal{E}^{\mathsf{fo}}$ with advantage $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}}, \mathcal{B}_{\mathsf{fo}}}(n)$, we can construct an efficient adversary $\mathcal{A}_{\mathsf{asy}}$ that attacks the CPA security of $\mathcal{B}_{\mathsf{asy}}$.

Since for asymmetric encryption schemes semantic security implies CPA security, and semantic security also implies resistance to message recovery attacks with the same advantage, we may assume that $\mathcal{A}_{\mathsf{asy}}$ attacks the CPA security of $\mathcal{B}_{\mathsf{asy}}$ in the message-recovery game.

Adversary $\mathcal{A}_{\mathsf{asy}}$ simulates the interaction of $\mathcal{A}_{\mathsf{fo}}$ in game G_4, including the random oracles and the decryption oracle. The game proceeds as follows:

1. The challenger $\mathfrak{S}$ in the message-recovery game computes $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}^{\mathsf{asy}}(1^n)$, samples $\sigma^* \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$, sets $\mathsf{e}^* \xleftarrow{\$} \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*)$, and sends e^* to $\mathcal{A}_{\mathsf{asy}}$.
2. $\mathcal{A}_{\mathsf{asy}}$ forwards the public key pk to $\mathcal{A}_{\mathsf{fo}}$, initiating the simulation of game G_4.
3. $\mathcal{A}_{\mathsf{fo}}$ submits two messages $(\mathsf{m}_0, \mathsf{m}_1)$.
4. $\mathcal{A}_{\mathsf{asy}}$ constructs the challenge for $\mathcal{A}_{\mathsf{fo}}$ by sampling $\mathsf{b} \xleftarrow{\$} \{0,1\}$, choosing a one-time key $\mathsf{k}^* \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$, and computing $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{k}^*, \mathsf{m}_{\mathsf{b}})$. It then sends the challenge $(\mathsf{e}^*, \mathsf{c}^*)$ to $\mathcal{A}_{\mathsf{fo}}$.

5. When $\mathcal{A}_{\mathsf{fo}}$ outputs a bit b', adversary $\mathcal{A}_{\mathsf{asy}}$ selects uniformly at random one of the queries from $Q_{\mathsf{G}} \parallel Q_{\mathsf{H}}$:

$$q \xleftarrow{\$} [Q_{\mathsf{G}} \parallel Q_{\mathsf{H}}],$$

and sends σ' (extracted from q) as its answer to the challenger.

Thus, q is either of the form $(\sigma', \mathsf{G}(\sigma'))$ or $(\sigma', \mathsf{c}', \mathsf{H}(\sigma', \mathsf{c}'))$. The value σ' is returned to $\mathfrak{S}$. Since q was chosen uniformly at random, we have

$$\Pr\left[\sigma^* = \sigma' | \mathsf{ASK}_{\sigma_4^*}\right] = \frac{1}{q_{\mathsf{H}}},$$
$$\Pr[\sigma^* = \sigma'] = \frac{1}{q_{\mathsf{H}}} \cdot \Pr\left[\mathsf{ASK}_{\sigma_4^*}\right].$$

Because $\mathcal{B}_{\mathsf{asy}}$ is CPA-secure, it follows that

$$\Pr[\sigma^* = \sigma'] = \frac{1}{q_{\mathsf{H}}} \cdot \Pr\left[\mathsf{ASK}_{\sigma_4^*}\right] \leq \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n),$$

which yields the bound

$$\Pr\left[\mathsf{ASK}_{\sigma_4^*}\right] \leq q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n).$$

Therefore,

$$\begin{aligned}\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_4] &\leq q_{\mathsf{Dec}} \cdot 2^{-\gamma} + \Pr\left[\mathsf{ASK}_{\sigma_4^*}\right] \\ &= q_{\mathsf{Dec}} \cdot 2^{-\gamma} + q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n).\end{aligned}$$

To bound $\Pr[\mathsf{S}_0]$, it remains to estimate $\Pr[\mathsf{S}_4]$. We now show that

$$\Pr[\mathsf{S}_4] \leq \frac{1}{2} + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}}, \mathcal{B}_{\mathsf{sym}}}(n).$$

As before, we use adversary $\mathcal{A}_{\mathsf{fo}}$ against the CCA security of $\mathcal{E}^{\mathsf{fo}}$ to construct an adversary $\mathcal{A}_{\mathsf{sym}}$ against the semantic security of $\mathcal{B}_{\mathsf{sym}}$. By assumption, $\mathcal{A}_{\mathsf{sym}}$ has advantage at most $\mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}}, \mathcal{B}_{\mathsf{sym}}}(n)$.

Adversary $\mathcal{A}_{\mathsf{sym}}$ simulates $\mathcal{A}_{\mathsf{fo}}$ as in game G_4, including the random oracles and the decryption oracle. The game proceeds as follows:

1. $\mathcal{A}_{\mathsf{sym}}$ generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{fo}}$ and sends pk to $\mathcal{A}_{\mathsf{fo}}$.
2. $\mathcal{A}_{\mathsf{fo}}$ submits $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}^{\mathsf{fo}}$. $\mathcal{A}_{\mathsf{sym}}$ forwards $(\mathsf{m}_0, \mathsf{m}_1)$ to its challenger $\mathfrak{S}$ in the semantic security game.
3. $\mathfrak{S}$ generates $\mathsf{k} \leftarrow \mathsf{KGen}^{\mathsf{sym}}$, samples $\mathsf{b} \xleftarrow{\$} \{0,1\}$, and computes $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_{\mathsf{b}})$, which is sent to $\mathcal{A}_{\mathsf{sym}}$.
4. $\mathcal{A}_{\mathsf{sym}}$ constructs the challenge for $\mathcal{A}_{\mathsf{fo}}$ as follows:
   - sample $\sigma^* \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$,
   - sample $\mathsf{h}^* \xleftarrow{\$} \mathcal{R}^{\mathsf{asy}}$,
   - compute $\mathsf{e}^* \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{h}^*)$,
   - form the challenge $(\mathsf{e}^*, \mathsf{c}^*)$ and send it to $\mathcal{A}_{\mathsf{fo}}$.
5. $\mathcal{A}_{\mathsf{fo}}$ returns a bit b' to $\mathcal{A}_{\mathsf{sym}}$, which is then forwarded to $\mathfrak{S}$.

Adversary $\mathcal{A}_{\mathsf{sym}}$ wins iff $\mathrm{b}' = \mathrm{b}$, which occurs precisely when $\mathcal{A}_{\mathsf{fo}}$ succeeds. Hence

$$\Pr[\mathrm{b}' = \mathrm{b}] = \frac{1}{2} + \mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) \leq \frac{1}{2} + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n).$$

From the perspective of $\mathcal{A}_{\mathsf{fo}}$, this experiment is identical to game G_4. Therefore

$$\Pr[\mathrm{S}_4] = \frac{1}{2} + \mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) \leq \frac{1}{2} + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n).$$

Recalling that

$$\Pr[\mathrm{S}_0] - \Pr[\mathrm{S}_4] \leq q_{\mathsf{Dec}} \cdot 2^{-\gamma} + q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n),$$

we obtain the upper bound

$$\Pr[\mathrm{S}_0] \leq q_{\mathsf{Dec}} \cdot 2^{-\gamma} + q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) + \frac{1}{2} + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n).$$

It follows that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) &= \left|\Pr[\mathrm{S}_0] - \frac{1}{2}\right| \\ &\leq q_{\mathsf{Dec}} \cdot 2^{-\gamma} + q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n).\end{aligned}$$

This completes the proof.

Finally, since

$$\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) \leq \mathsf{negl}(n) \quad \text{and} \quad \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n) \leq \mathsf{negl}(n),$$

we conclude that $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) \leq \mathsf{negl}(n)$. Thus the theorem is proven.

The Fujisaki–Okamoto Transform in the QROM

In the previous section we presented a complete security proof of the Fujisaki–Okamoto (FO) transform in the classical ROM. However, this proof cannot be directly transferred to the quantum setting. We now highlight the key obstacles that arise when considering a quantum adversary:

- In the ROM proof, we constructed a sequence of games. In game 1 (3.3.2), the decryption oracle was simulated by answering the adversary's queries not through the secret key, but by using lists of queries and answers to the random oracles involved in the scheme. For a quantum adversary, who may query the oracle in superposition over all possible inputs, constructing and maintaining such lists becomes problematic for the simulator.

- In game 2 (3.3.3), we replaced the values $\mathsf{G}(\sigma^*)$ and $\mathsf{H}(\sigma^*, \mathrm{c}^*)$ with random values a^* and r^* of the same distribution. In game 3 (3.3.4), we reintroduced $\mathsf{G}(\sigma^*)$ and $\mathsf{H}(\sigma^*, \mathrm{c}^*)$ as oracle responses, while still using a^* and r^* in the challenge ciphertext. From the adversary's perspective, the games are indistinguishable—except when the adversary queries the oracle at σ^* or (σ^*, c^*). In the classical model, the probability of this event was shown to be negligible. In the quantum model, however, the adversary can always include such queries within its superposition, making the argument substantially more difficult.

At present, no general technique is known for overcoming these issues in the FO transform. Following (Targhi & Unruh, 2015), we consider a modified variant of FO that introduces an additional hash function $\mathsf{H}' : \mathcal{M}^{\mathsf{asy}} \to \mathcal{M}^{\mathsf{asy}}$.

The encryption algorithm is modified as follows, by adding step 4:

1. $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$,

2. $\mathsf{c} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{G}(\sigma), \mathsf{m})$,
3. $\mathsf{e} \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma; \mathsf{H}(\sigma, \mathsf{c}))$,
4. $\mathsf{d} \leftarrow \mathsf{H}'(\sigma)$,
5. output ciphertext $(\mathsf{e}, \mathsf{c}, \mathsf{d}) \leftarrow \mathsf{Enc}^{\mathsf{hy}}(\mathsf{pk}, \mathsf{m})$.

The decryption algorithm is modified by adding step 5 to verify the value of H' on $\hat{\sigma}$:

1. $\hat{\sigma} \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{e})$,
2. if $\hat{\sigma} \notin \mathcal{M}^{\mathsf{asy}}$, return $\perp$,
3. $\hat{\mathsf{h}} \leftarrow \mathsf{H}(\hat{\sigma}, c)$,
4. if $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \hat{\sigma}; \hat{h})$, return $\perp$,
5. if $\mathsf{d} \neq \mathsf{H}'(\hat{\sigma})$, return $\perp$,
6. $\hat{\mathsf{k}} \leftarrow \mathsf{G}(\hat{\sigma})$,
7. $y \leftarrow \mathsf{Dec}^{\mathsf{sy}}(\hat{\mathsf{k}}, c)$,
8. output message $y \leftarrow \mathsf{Dec}^{\mathsf{hy}}(\mathsf{sk}, (\mathsf{e}, \mathsf{c}, \mathsf{d}))$.

This modified construction differs from the standard FO transform only by the introduction of the additional function H'. As a result, the ciphertext is extended by one component, and the algorithms are augmented with the computation of $\mathsf{H}'(\sigma)$ during encryption and its verification during decryption. All other components of the scheme remain unchanged. The assumptions on the underlying building blocks, namely $\mathcal{B}_{\mathsf{asy}}$ and $\mathcal{B}_{\mathsf{sym}}$, are also preserved.

Security Proof of the Fujisaki–Okamoto Scheme in the QROM

Let $\mathcal{B}_{\mathsf{asy}} = (\mathsf{KGen}^{\mathsf{asy}}, \mathsf{Enc}^{\mathsf{asy}}, \mathsf{Dec}^{\mathsf{asy}})$ be an asymmetric encryption scheme, defined over $(\mathcal{M}^{\mathsf{asy}}, \mathcal{C}^{\mathsf{asy}}, \mathcal{R}^{\mathsf{asy}})$, which is secure in the CPA model. Let $\mathcal{A}_{\mathsf{asy}}$ denote an adversary attacking scheme $\mathcal{B}_{\mathsf{asy}}$ in this model. This means that the following condition holds:

$$\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n) \leq \mathsf{negl}(n).$$

Furthermore, let $\mathcal{B}_{\mathsf{sym}} = (\mathsf{KGen}^{\mathsf{sym}}, \mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ be a symmetric encryption scheme, defined over $(\mathcal{K}^{\mathsf{sym}}, \mathcal{M}^{\mathsf{sym}}, \mathcal{C}^{\mathsf{sym}})$, which satisfies semantic security. Thus, the advantage of an adversary $\mathcal{A}_{\mathsf{sym}}$ is negligible:

$$\mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}}, \mathcal{B}_{\mathsf{sym}}}(n) \leq \mathsf{negl}(n).$$

Theorem 2. *Let $\mathcal{E}^{\mathsf{fo}} = (\mathsf{KGen}^{\mathsf{fo}}, \mathsf{Enc}^{\mathsf{fo}}, \mathsf{Dec}^{\mathsf{fo}})$ over $(\mathcal{K}^{\mathsf{fo}}, \mathcal{M}^{\mathsf{fo}}, \mathcal{C}^{\mathsf{fo}})$ be the hybrid scheme obtained via the Fujisaki–Okamoto transform applied to $\mathcal{B}_{\mathsf{asy}}$ and $\mathcal{B}_{\mathsf{sym}}$. Under the above assumptions, there exists a quantum adversary $\mathcal{A}_{\mathsf{fo}}$, making at most $q_{\mathsf{fo}} = q_{\mathsf{G}} + q_{\mathsf{H}} + q_{\mathsf{H}'} + q_{\mathsf{Dec}} + 1$ queries to the random oracles $\mathsf{G}, \mathsf{H}, \mathsf{H}'$, to the decryption oracle, and to the encryption oracle (the additional $+1$ accounts for the mandatory encryption query used to obtain the challenge), such that its advantage satisfies:*

$$\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}}, \mathcal{B}_{\mathsf{fo}}}(n) \leq O\left(\frac{{q_{\mathsf{fo}}}^{9/5}}{2^{\gamma/5}}\right) + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}}, \mathcal{B}_{\mathsf{sym}}}(n) + 2q_{\mathsf{G}\times\mathsf{H}'}\sqrt{\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n) + 2q_{H1}\sqrt{\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}}, \mathcal{B}_{\mathsf{asy}}}(n)} + q_{H0} \cdot 2^{-l/2+2}}.$$

Here $q_{\mathsf{G}\times\mathsf{H}'} = q_{\mathsf{G}} + q_{\mathsf{H}'} + 2q_{\mathsf{Dec}}$, q_{H0} denotes the number of queries made by $\mathcal{A}_{\mathsf{fo}}$ to H before receiving the challenge, and q_{H1} the number of queries made to H after receiving the challenge.

Proof. The analysis proceeds by stating a few technical assumptions and fixing notation to be used throughout:

- As in the classical random oracle proof, we construct a sequence of games and derive upper bounds on the differences between the adversary's winning probabilities.
- We denote by S_i (for $0 \leq i \leq l$) the event that the adversary succeeds in game G_i, consistent with the classical approach.
- Random variables sampled by the simulator are annotated with index *, following the convention used earlier.

Game G_0

Analogously to the classical proof, Game G_0 represents the baseline version of a CCA attack on the scheme $\mathsf{Enc}^{\mathsf{fo}}$ by the adversary $\mathcal{A}_{\mathsf{fo}}$, who has quantum access to the oracles $\mathsf{H}, \mathsf{G}, \mathsf{H}'$ and classical access to the decryption oracle:

1. The challenger generates a key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}^{\mathsf{fo}}(1^n)$.
2. The adversary receives the public key pk and quantum access to the oracles $\mathsf{G}, \mathsf{H}, \mathsf{H}'$, as well as classical access to $\mathsf{Dec}^{\mathsf{fo}}$.
3. The adversary submits two messages: $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}^{\mathsf{fo}}$.
4. The challenger samples a bit $\mathsf{b} \xleftarrow{\$} \{0,1\}$ and encrypts the message, generating the challenge ciphertext: $(\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*) \leftarrow \mathsf{Enc}^{\mathsf{fo}}(\mathsf{pk}, \mathsf{m}_\mathsf{b})$.
5. The adversary receives $(\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*)$ along with renewed access to $\mathsf{G}, \mathsf{H}, \mathsf{H}'$ and $\mathsf{Dec}^{\mathsf{fo}}$. Finally, it outputs a bit b'.

We say that the adversary wins if $\mathsf{b} = \mathsf{b}'$ (event S_0). By definition, we have:

$$\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) = \left|\Pr[\mathsf{S}_0] - \frac{1}{2}\right|.$$

In the subsequent steps, we will gradually modify the game G_i and establish upper bounds on the differences between the adversary's success probabilities, i.e., $\Pr[\mathsf{S}_{i-1}] - \Pr[\mathsf{S}_i]$. Once the final probability $\Pr[\mathsf{S}_l]$ is determined, we will obtain an upper bound on $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n)$ and demonstrate that it is negligible.

Game G_1

In Game G_1, we introduce a modification with respect to G_0, consisting of replacing the decryption algorithm $\mathsf{Dec}^{\mathsf{fo}}$ with Dec^*, which for each query of the form $(\mathsf{e}, \mathsf{c}, \mathsf{d})$ performs the following steps:

1. If $\mathsf{e} = \mathsf{e}^*$ it returns $\perp$.
2. If $\hat{\sigma} \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{e}) \notin \mathcal{M}^{\mathsf{asy}}$, it sends a query $\mathsf{H}'(\sigma^* \oplus 1)$ and returns $\perp$.
3. If $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \hat{\sigma}; \mathsf{H}(\hat{\sigma}, \mathsf{c}))$, it sends a query $\mathsf{H}'(\sigma^* \oplus 1)$ and returns $\perp$.
4. If $\mathsf{d} \neq \mathsf{H}'(\hat{\sigma})$ it returns $\perp$.
5. Otherwise, it outputs $\mathsf{Dec}^{\mathsf{sy}}(\mathsf{G}(\hat{\sigma}), c)$.

The most important change from the adversary's perspective is step 1: the new algorithm always returns the error symbol when the asymmetric part of the ciphertext matches that of the challenge. Additionally, in steps 2 and 3, before returning an error, the algorithm queries the oracle H' with input $\sigma^* \oplus 1$. The reason for this operation will become clear later in the proof, during the analysis of the differences between Games G_4 and G_5.

Our goal is to show that the difference $|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1]|$ is negligible. To this end, let us analyze situations in which the adversary could distinguish the two games by issuing a decryption query $(\mathsf{e}, \mathsf{c}, \mathsf{d})$ to the oracle $\mathsf{Dec}^{\mathsf{fo}}$ in Game G_0 and to the oracle Dec^* in Game G_1.

Observe that if $\mathsf{e} \neq \mathsf{e}^*$, both decryption oracles behave identically. Thus, the only cases worth considering are those where $\mathsf{e} = \mathsf{e}^*$ (since Dec^* always returns $\perp$ in that case):

- $\mathsf{c} = \mathsf{c}^*, \mathsf{d} = \mathsf{d}^*$ - such a query cannot occur, since the adversary is not allowed to request decryption of the challenge ciphertext.
- $\mathsf{c} = \mathsf{c}^*, \mathsf{d} \neq \mathsf{d}^*$ - here $\mathsf{Dec}^{\mathsf{fo}}$ also returns $\perp$.
- $\mathsf{c} \neq \mathsf{c}^*, \mathsf{d} = \mathsf{d}^*$ - this implies that $\mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{H}(\sigma^*, \mathsf{c})) = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{H}(\sigma^*, \mathsf{c}^*))$, in which case the algorithm $\mathsf{Dec}^{\mathsf{fo}}$ would output the decryption result $\mathsf{Dec}^{\mathsf{fo}}(\mathsf{sk}, (\mathsf{e}, \mathsf{c}, \mathsf{d}))$.

To bound the probability of such a situation, we use the lemma derived in :

Lemma 3. *Let* $f: \{0,1\}^{n_1} \to \{0,1\}^{n_2}$ *be a function with min-entropy* k*, and let* $\mathsf{H}: \{0,1\}^* \to \{0,1\}^{n_1}$ *be a random oracle. Then any quantum algorithm* $\mathcal{A}$ *making* q *queries to* H *will encounter a collision with probability at most* $O\left(\frac{q^{9/5}}{2^{k/5}}\right)$.

Since in the scheme $\mathsf{Enc}^{\mathsf{asy}}$ encrypting any message may yield at least 2^{γ} possible ciphertexts (i.e., the encryption algorithm has min-entropy γ), and the event $\mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{H}(\sigma^*, \mathsf{c})) = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{H}(\sigma^*, \mathsf{c}^*))$ can be treated as a collision, we obtain:

$$|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1]| \leq O\left(\frac{{q_{\mathsf{fo}}}^{9/5}}{2^{\gamma/5}}\right) \leq \mathsf{negl}(n).$$

Games G_2 *and* G_3

In Game G_2, we modify Game G_1 by replacing the values $\mathsf{G}(\sigma^*)$ and $\mathsf{H}'(\sigma^*)$ with random values $\mathsf{a}^* \in \mathcal{K}^{\mathsf{sym}}$ and $\mathsf{d}^* \in \mathcal{M}^{\mathsf{asy}}$, respectively. The interaction of the adversary $\mathcal{A}_{\mathsf{fo}}$ with the simulator proceeds analogously to Game G_1, with the following differences:

- the simulator uses a^* instead of $\mathsf{G}(\sigma^*)$ as the symmetric encryption key, and
- the value d^* as the ciphertext component $(\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*)$ instead of $\mathsf{H}'(\sigma^*)$.

The randomness used during asymmetric encryption ($\mathsf{H}(\sigma^*, \mathsf{c}^*)$) remains unchanged.

Using the assumption of semantic security of $\mathsf{Enc}^{\mathsf{sym}}$, we can show that

$$\Pr[\mathsf{S}_2] = \frac{1}{2} + \mathsf{negl}(n)$$

(see proof in).

Let us now bound the difference $|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_2]|$. In the classical random oracle model, we defined the event BAD and analyzed probability differences via the estimate of $\Pr[\mathsf{BAD}]$. In the quantum random oracle model, however, this approach cannot be applied directly. Instead, we rely on the O2H Lemma ([O2H]).

Assume $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ is a quantum adversary with access to the random oracle $\mathsf{G} \times \mathsf{H}'$ (where $\mathsf{G} \times \mathsf{H}'(\sigma) = (\mathsf{G}(\sigma), \mathsf{H}'(\sigma))$). Upon receiving input $(\sigma^*, \mathsf{a}^*, \mathsf{d}^*)$, it performs the following steps:

- selects a function $\mathsf{H}: \mathcal{M}^{\mathsf{asy}} \times \mathcal{C}^{\mathsf{sym}} \to \mathcal{R}^{\mathsf{asy}}$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{asy}}(1^n)$, samples a bit $\mathsf{b} \xleftarrow{\$} \{0,1\}$, and gives the public key pk to $\mathcal{A}_{\mathsf{fo}}$,
- $\mathcal{A}_{\mathsf{fo}}$ issues queries to the oracles $\mathsf{H}, \mathsf{G}, \mathsf{H}', \mathsf{Dec}^*$ via interaction with $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$, and eventually submits a challenge request $(\mathsf{m}_0, \mathsf{m}_1)$,
- $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ computes $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{a}^*, \mathsf{m}_\mathsf{b})$, $\mathsf{e}^* \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*, \mathsf{H}(\sigma^*, \mathsf{c}^*))$, and sends the challenge $(\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*)$ to $\mathcal{A}_{\mathsf{fo}}$,
- $\mathcal{A}_{\mathsf{fo}}$ issues further queries to the oracles $\mathsf{H}, \mathsf{G}, \mathsf{H}', \mathsf{Dec}^*$ and finally outputs a bit b',

- $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ outputs whether $\mathsf{b} = \mathsf{b}'$.

We can estimate that the maximum number of queries $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ makes to random oracles (on behalf of $\mathcal{A}_{\mathsf{fo}}$) is $q_{\mathsf{G}\times\mathsf{H}'} = q_{\mathsf{G}} + q_{\mathsf{H}'} + 2q_{\mathsf{Dec}}$ (since the algorithm Dec^* makes two oracle calls to $\mathsf{G}\times\mathsf{H}'$).

Now, define algorithm $\mathcal{B}$, which - upon receiving σ^* - operates according to the following Game G_3:

- samples $i \xleftarrow{\$} \{1, \dots, q_{\mathsf{G}\times\mathsf{H}'}\}$, $\mathsf{a}^* \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$, and $\mathsf{d}^* \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$,
- sends $(\sigma^*, \mathsf{a}^*, \mathsf{d}^*)$ to $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$,
- before $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ makes its i-th query to the oracle $\mathsf{G}\times\mathsf{H}'$, $\mathcal{B}$ measures the query argument and obtains $\hat{\sigma}$ (if $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ makes fewer than i queries, $\mathcal{B}$ returns $\perp$),
- $\mathcal{B}$ wins in G_3 if $\hat{\sigma} = \sigma^*$, in which case it outputs 1.

Table 1 summarizes the behavior of $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ and $\mathcal{B}$ in Game G_3.

Notice that for the probabilities defined in the O2H Lemma, we have:

$$P_{\mathcal{A}}^1 = \Pr[1 \leftarrow G_1] = \Pr[\mathsf{S}_1],$$
$$P_{\mathcal{A}}^2 = \Pr[1 \leftarrow G_2] = \Pr[\mathsf{S}_2],$$

with respect to adversary $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$, and:

$$P_{\mathcal{B}} = \Pr[1 \leftarrow G_3] = \Pr[\mathsf{S}_3],$$

with respect to adversary $\mathcal{B}$. Based on the analysis and the O2H Lemma, we obtain:

$$|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_2]| \leq 2q_{\mathsf{G}\times\mathsf{H}'}\sqrt{\Pr[\mathsf{S}_3]}.$$

It remains to estimate $\Pr[\mathsf{S}_3]$, which we address in the next game.

Table 1: Operation of algorithms $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ and $\mathcal{B}$ in Game G_3

$\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}(\sigma^*, \mathsf{a}^*, \mathsf{d}^*)$	$\mathcal{B}(\sigma^*)$:
1. $\mathsf{H}: \mathcal{M}^{\mathsf{asy}} \times \mathcal{C}^{\mathsf{sym}} \to \mathcal{R}^{\mathsf{asy}}$ $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{asy}}(n)$ $\mathsf{b} \xleftarrow{\$} \{0,1\}$	1. $i \xleftarrow{\$} \{1, \dots, q_1\}$, $\mathsf{a}^* \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$ $\mathsf{d}^* \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$
2. $(\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}_{\mathsf{fo}}{}^{\mathsf{H},\mathsf{G},\mathsf{H}',\mathsf{Dec}^*}(\mathsf{pk})$.	2. sends $(\sigma^*, \mathsf{a}^*, \mathsf{d}^*)$ to $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$
3. $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{a}^*, \mathsf{m}_\mathsf{b})$, $\mathsf{e}^* \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*, \mathsf{H}(\sigma^*, \mathsf{c}^*))$	3. before $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ makes its i-th query to $\mathsf{G}\times\mathsf{H}'$, $\mathcal{B}$ measures the query argument obtaining $\hat{\sigma}$
4. $\mathsf{b}' \leftarrow \mathcal{A}_{\mathsf{fo}}{}^{\mathsf{H},\mathsf{G},\mathsf{H}',\mathsf{Dec}^*}((\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*))$	4. if $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ makes fewer than i queries, $\mathcal{B}$ returns $\perp$
5. outputs whether $\mathsf{b} = \mathsf{b}'$	5. outputs whether $\hat{\sigma} = \sigma^*$

Game G_4

In game G_4, we modify game G_3 by replacing the random oracle H' with a $2(q_{\mathsf{H}'} + q_{\mathsf{Dec}})$-wise independent function (as defined in [wiseindependent]). This function is constructed as a random polynomial of degree $2(q_{\mathsf{H}'} + q_{\mathsf{Dec}}) - 1$ with coefficients in the field $GF(\mathcal{M}^{\mathsf{asy}})$. Since a $2(q_{\mathsf{H}'} + q_{\mathsf{Dec}})$-wise independent function is

indistinguishable from a truly random oracle from the perspective of a quantum algorithm making at most $q_{\mathsf{H}'} + q_{\mathsf{Dec}}$ queries, we obtain:

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_4].$$

Game G_5

In game G_5, we once again modify the decryption algorithm Dec^*, replacing it with the algorithm Dec^{**}, which:

- Has full knowledge of the polynomial used to simulate the oracle H'. In what follows, we will continue to use the notation H', keeping in mind that in the context of game G_5 it refers to the polynomial rather than an ideal random oracle.
- Upon receiving a query of the form $(\mathsf{e}, \mathsf{c}, \mathsf{d})$, performs the following steps:
    1. If $\mathsf{e} = \mathsf{e}^*$, return $\perp$.
    2. Construct the set Ω consisting of all roots of the polynomial $\mathsf{H}' - \mathsf{d}$.
    3. If Ω contains $\hat{\sigma} \neq \sigma^*$ such that $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{sk}, \hat{\sigma}; \mathsf{H}(\hat{\sigma}, \mathsf{c}))$:
        - Compute $\mathsf{H}'_{\mathsf{poly}}(\hat{\sigma})$.
        - Compute $\hat{\mathsf{a}} \leftarrow \mathsf{G}(\hat{\sigma})$.
        - Return $\mathsf{Dec}^{\mathsf{sym}}(\hat{\mathsf{a}}, \mathsf{c})$.
    4. If $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{H}(\sigma^*, \mathsf{c}))$, compute H' for a random value $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}} \backslash \{\sigma^*\}$ and return $\perp$.
    5. If $\mathsf{H}'(\sigma^*) = \mathsf{d}$:
        - Compute $\hat{\mathsf{a}} \leftarrow \mathsf{G}(\sigma^*)$.
        - Return $\mathsf{Dec}^{\mathsf{sym}}(\hat{\mathsf{a}}, \mathsf{c})$.
    6. Otherwise, return $\perp$.

Since game G_5 differs from game G_4 only in the decryption algorithm, it suffices to show that the algorithms Dec^* and Dec^{**} behave identically with respect to any possible query. For clarity, Table 2 compares the steps of algorithms Dec^* and Dec^{**}.

Table 2: Comparison of the behavior of algorithms $\boldsymbol{Dec}^*$ and $\boldsymbol{Dec}^{**}$

$\mathsf{Dec}^*(\mathsf{e}, \mathsf{c}, \mathsf{d})$:	$\mathsf{Dec}^{**}(\mathsf{e}, \mathsf{c}, \mathsf{d})$:
1. If $\mathsf{e} = \mathsf{e}^*$ return $\perp$ 2. If $\hat{\sigma} \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk}, \mathsf{e}) \notin \mathcal{M}^{\mathsf{asy}}$, query $\mathsf{H}'(\sigma^* \oplus 1)$ and return $\perp$ 3. If $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \hat{\sigma}; \mathsf{H}(\hat{\sigma}, \mathsf{c}))$, query $\mathsf{H}'(\sigma^* \oplus 1)$ and return $\perp$ 4. If $\mathsf{d} \neq \mathsf{H}'(\hat{\sigma})$, return $\perp$ 5. Return $\mathsf{Dec}^{\mathsf{sym}}(\mathsf{G}(\hat{\sigma}), \mathsf{c})$	1. If $\mathsf{e} = \mathsf{e}^*$, return $\perp$. 2. Construct the set Ω consisting of all roots of the polynomial $\mathsf{H}' - \mathsf{d}$. 3. If Ω contains $\hat{\sigma} \neq \sigma^*$ such that $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{sk}, \hat{\sigma}; \mathsf{H}(\hat{\sigma}, \mathsf{c}))$: ○ Query $\mathsf{H}'(\hat{\sigma})$ ○ Compute $\hat{\mathsf{a}} \leftarrow \mathsf{G}(\hat{\sigma})$ ○ Return $\mathsf{Dec}^{\mathsf{sym}}(\hat{\mathsf{a}}, \mathsf{c})$ 4. If $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{H}(\sigma^*, \mathsf{c}))$, compute H' for a random value $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}} \backslash \{\sigma^*\}$ and return $\perp$

$\mathsf{Dec}^*(\mathsf{e},\mathsf{c},\mathsf{d})$:	$\mathsf{Dec}^{**}(\mathsf{e},\mathsf{c},\mathsf{d})$:
	5. If $\mathsf{H}'(\sigma^*) = \mathsf{d}$: ○ Compute $\hat{\mathsf{a}} \leftarrow \mathsf{G}(\sigma^*)$ ○ Return $\mathsf{Dec}^{\mathsf{sym}}(\hat{\mathsf{a}},\mathsf{c})$ 6. Return $\perp$

Suppose that the adversary submits a query to the decryption oracle of the form $(\mathsf{c},\mathsf{e},\mathsf{d})$ and let $\hat{\sigma} \leftarrow \mathsf{Dec}^{\mathsf{asy}}(\mathsf{sk},\mathsf{e})$. We consider the following cases:

1. $\hat{\sigma} = \perp$:
   Both algorithms issue a query $\mathsf{H}'(\sigma)$ for some $\sigma \neq \sigma^*$ and return $\perp$.

2. $\hat{\sigma} \neq \perp, \hat{\sigma} \neq \sigma^*, \mathsf{H}'(\hat{\sigma}) \neq \mathsf{d}$:
   Two cases arise:
   - $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\hat{\sigma};\mathsf{H}(\hat{\sigma},\mathsf{c}))$: then Dec^* queries $\mathsf{H}'(\sigma^* \oplus 1)$ and returns $\perp$, while Dec^{**} computes H' for a random value $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}} \backslash \{\sigma^*\}$ and also returns $\perp$.
   - $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\hat{\sigma};\mathsf{H}(\hat{\sigma},\mathsf{c}))$: then Dec^* returns $\perp$ since $\mathsf{H}'(\hat{\sigma}) \neq \mathsf{d}$, and Dec^{**} also returns $\perp$ because $\hat{\sigma} \notin \Omega$, which likewise follows from $\mathsf{H}'(\hat{\sigma}) \neq \mathsf{d}$.

3. $\hat{\sigma} \neq \perp, \hat{\sigma} \neq \sigma^*, \mathsf{H}'(\hat{\sigma}) = \mathsf{d}$:
   Two cases arise:
   - $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\hat{\sigma};\mathsf{H}(\hat{\sigma},\mathsf{c}))$: then Dec^* queries $\mathsf{H}'(\sigma^* \oplus 1)$ and returns $\perp$, while Dec^{**} computes H' for a random value $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}} \backslash \{\sigma^*\}$ and also returns $\perp$.
   - $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\hat{\sigma};\mathsf{H}(\hat{\sigma},\mathsf{c}))$: then Dec^* queries $\mathsf{H}'(\hat{\sigma})$ and returns $\mathsf{Dec}^{\mathsf{sym}}(\mathsf{G}(\hat{\sigma}),\mathsf{c})$, and Dec^{**} does the same.

4. $\hat{\sigma} = \sigma^*, \mathsf{H}'(\hat{\sigma}) \neq \mathsf{d}$:
   Three cases arise:
   - If $\mathsf{e}^* = \mathsf{e}$: then both algorithms return $\perp$.
   - If $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\sigma^*;\mathsf{H}(\sigma^*,\mathsf{c}))$: then Dec^* queries $\mathsf{H}'(\sigma^* \oplus 1)$ and returns $\perp$, while Dec^{**} computes H' for a random value $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}} \backslash \{\sigma^*\}$ and returns $\perp$.
   - $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\sigma^*;\mathsf{H}(\sigma^*,\mathsf{c}))$: then Dec^* returns $\perp$ since $\mathsf{H}'(\hat{\sigma}) \neq \mathsf{d}$, and Dec^{**} also returns $\perp$ because $\hat{\sigma} \notin \Omega$, which likewise follows from $\mathsf{H}'(\hat{\sigma}) \neq \mathsf{d}$.

5. $\hat{\sigma} = \sigma^*, \mathsf{H}'(\hat{\sigma}) = \mathsf{d}$:
   Three cases arise:
   - If $\mathsf{e}^* = \mathsf{e}$: then both algorithms return $\perp$.
   - If $\mathsf{e} \neq \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\sigma^*;\mathsf{H}(\sigma^*,\mathsf{c}))$: then Dec^* queries $\mathsf{H}'(\sigma^* \oplus 1)$ and returns $\perp$, while Dec^{**} computes H' for a random value $\sigma \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}} \backslash \{\sigma^*\}$ and returns $\perp$.
   - $\mathsf{e} = \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk},\sigma^*;\mathsf{H}(\sigma^*,\mathsf{c}))$: then both algorithms query $\mathsf{H}'(\sigma^*)$ and return $\mathsf{Dec}^{\mathsf{sym}}(\mathsf{G}(\sigma^*),\mathsf{c})$.

Based on the above analysis, we can write:

$$\Pr[\mathsf{S}_4] = \Pr[\mathsf{S}_5].$$

Games G_6 and G_7

The value $\mathsf{H}(\sigma^*, \mathsf{c}^*)$ has so far served as the randomness seed in the asymmetric encryption algorithm. We now replace it with actual randomness from $\mathcal{R}^{\mathsf{asy}}$. Table 3 presents the operation of algorithms $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ and $\mathcal{B}$ in game G_6:

Table 3: Operation of algorithms $\mathcal{A}^{G\times H'}$ and $\mathcal{B}$ in game G_6

$\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}(\sigma^*, \mathsf{a}^*, \mathsf{d}^*)$	$\mathcal{B}(\sigma^*)$:
1. $\mathsf{H}: \mathcal{M}^{\mathsf{asy}} \times \mathcal{C}^{\mathsf{sym}} \to \mathcal{R}^{\mathsf{asy}}$ $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}^{\mathsf{asy}}(n)$ $\mathsf{b} \xleftarrow{\$} \{0,1\}$	1. $i \xleftarrow{\$} \{1, \dots, q_1\}$ $\mathsf{a}^* \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$ $\mathsf{d}^* \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$
2. $(\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}_{\mathsf{fo}}{}^{(\mathsf{H},\mathsf{G},\mathsf{H}',\mathsf{Dec}^{**})}(\mathsf{pk})$.	2. send $(\sigma^*, \mathsf{a}^*, \mathsf{d}^*)$ to $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$
3. $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{a}^*, \mathsf{m}_\mathsf{b})$ $\mathsf{e}^* \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*)$	3. before the i-th query of $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ to the oracle $\mathsf{G}\times\mathsf{H}'$, $\mathcal{B}$ performs a measurement of the query argument and obtains $\hat{\sigma}$
4. $\mathsf{b}' \leftarrow \mathcal{A}_{\mathsf{fo}}{}^{\mathsf{H},\mathsf{G},\mathsf{H}',\mathsf{Dec}^{**}}(\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*)$	4. if $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$ makes fewer than i queries, $\mathcal{B}$ returns $\perp$
5. return the result of the comparison $\mathsf{b} = \mathsf{b}'$	5. return the result of the comparison: $\hat{\sigma} = \sigma^*$

To estimate the upper bound of the difference $|\Pr[\mathsf{S}_5] - \Pr[\mathsf{S}_6]|$, we use Lemma O2HA (2).

Assume that adversary $\mathcal{A}_{\mathsf{fo}}$ makes q_{bf} queries to the oracle $\mathsf{G}\times\mathsf{H}'$ before submitting the encryption query.

We create two adversaries with access to the random oracle H: $\mathcal{A}_{\mathsf{H0}}$ and $\mathcal{A}_{\mathsf{H1}}$ by "splitting" adversary $\mathcal{A}^{\mathsf{G}\times\mathsf{H}'}$. $\mathcal{A}_{\mathsf{H0}}$ samples $i \xleftarrow{\$} \{1, \dots, q_{\mathsf{G}\times\mathsf{H}'}\}$ and performs the following:

- $\mathsf{a}^* \xleftarrow{\$} \mathcal{K}^{\mathsf{sym}}$, $\mathsf{d}^* \xleftarrow{\$} \mathcal{M}^{\mathsf{asy}}$,
- $(\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}_{\mathsf{fo}}{}^{(\mathsf{H},\mathsf{G},\mathsf{H}',\mathsf{Dec}^{**})}(\mathsf{pk})$,
- $\mathsf{c}^* \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathsf{a}^*, \mathsf{m}_\mathsf{b})$.

Before the i-th query to the oracle $\mathsf{G}\times\mathsf{H}'$, $\mathcal{A}_{\mathsf{H0}}$ outputs c^*.

Adversary $\mathcal{A}_{\mathsf{H1}}$ has access to the parameters and final state of $\mathcal{A}_{\mathsf{H0}}$. Upon receiving (σ^*, h^*) as input, it performs the following:

- If $i > q_{bf}$ (i.e., $\mathcal{A}_{\mathsf{fo}}$ has already submitted the encryption query):
  - $\mathsf{e}^* \leftarrow \mathsf{Enc}^{\mathsf{asy}}(\mathsf{pk}, \sigma^*; \mathsf{h}^*)$,
  - $\mathsf{b}' \leftarrow \mathcal{A}_{\mathsf{fo}}{}^{\mathsf{H},\mathsf{G},\mathsf{H}',\mathsf{Dec}^{**}}(\mathsf{e}^*, \mathsf{c}^*, \mathsf{d}^*)$.
- It measures the argument of the i-th query to $\mathsf{G}\times\mathsf{H}'$, obtaining $\hat{\sigma}$.
- Returns the result of the comparison $\hat{\sigma} = \sigma^*$.

Assuming that $\mathcal{A}_{\mathsf{H0}}$ makes q_{H0} queries to oracle H and $\mathcal{A}_{\mathsf{H1}}$ makes q_{H1} queries to oracle H, we define algorithm $\mathcal{B}$, which—after receiving σ^*—operates according to the following game G_7:

- Sample $j \xleftarrow{\$} \{1, \dots, q_{H1}\}$, $\mathrm{h} \xleftarrow{\$} \mathcal{R}^{\mathsf{asy}}$.
- Send (σ^*, h^*) to $\mathcal{A}_{\mathsf{H1}}$.
- Before $\mathcal{A}_{\mathsf{H1}}$ issues its j-th query to oracle H, $\mathcal{B}$ measures the query argument and obtains $\hat{\sigma}$ (if $\mathcal{A}_{\mathsf{H1}}$ makes fewer than j queries, $\mathcal{B}$ returns $\perp$).
- $\mathcal{B}$ wins in G_7 if $\hat{\sigma} = \sigma^*$, in which case it returns 1 as the outcome of the game.

Now note that for the probabilities defined in Lemma O2HA we have:

$$P_{\mathcal{A}}^1 = \Pr[1 \leftarrow G_5] = \Pr[\mathrm{S}_5],$$
$$P_{\mathcal{A}}^2 = \Pr[1 \leftarrow G_6] = \Pr[\mathrm{S}_6],$$

with respect to adversary $\mathcal{A}_{\mathsf{H1}}$, and:

$$P_{\mathcal{B}} = \Pr[1 \leftarrow G_7] = \Pr[\mathrm{S}_7],$$

with respect to adversary $\mathcal{B}$. Based on the analysis and Lemma O2HA, we obtain:

$$|\Pr[\mathrm{S}_5] - \Pr[\mathrm{S}_6]| \leq 2q_{H1}\sqrt{\Pr[\mathrm{S}_7]} + q_{H0} \cdot 2^{-l/2+2}.$$

where l is the bit-length of elements from $\mathcal{M}^{\mathsf{asy}}$.

Using the assumption of CPA-security of $\mathcal{B}_{\mathsf{asy}}$, we can show that

$$\Pr[\mathrm{S}_6] \leq \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) \text{ and } \Pr[\mathrm{S}_7] \leq \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n)$$

(proof in (Targhi & Unruh, 2015)).

Bounding the Adversary's Advantage

Based on the established bounds, we obtain

$$\Pr[\mathrm{S}_0] \leq O\left(\frac{{q_{\mathsf{fo}}}^{9/5}}{2^{\gamma/5}}\right) + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n) + 2q_{\mathsf{G}\times\mathsf{H}'}\sqrt{\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) + 2q_{H1}\sqrt{\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n)} + q_{H0} \cdot 2^{-l/2+2}}.$$

Recalling that

$$\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) = \left|\Pr[\mathrm{S}_0] - \frac{1}{2}\right|,$$

we can upper-bound $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n)$ as

$$\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) \leq O\left(\frac{{q_{\mathsf{fo}}}^{9/5}}{2^{\gamma/5}}\right) + \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n) + 2q_{\mathsf{G}\times\mathsf{H}'}\sqrt{\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) + 2q_{H1}\sqrt{\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n)} + q_{H0} \cdot 2^{-l/2+2}}.$$

This proves the claimed bound. Since

$$\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}_{\mathsf{asy}},\mathcal{B}_{\mathsf{asy}}}(n) \leq \mathsf{negl}(n) \quad \text{and} \quad \mathsf{Adv}^{\mathrm{SEM}}_{\mathcal{A}_{\mathsf{sym}},\mathcal{B}_{\mathsf{sym}}}(n) \leq \mathsf{negl}(n),$$

it follows that $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}_{\mathsf{fo}},\mathcal{B}_{\mathsf{fo}}}(n) \leq \mathsf{negl}(n)$. This completes the proof of the theorem.

Evaluating the Fujisaki-Okamoto Transform with CRYSTALS-Kyber

CRYSTALS-Kyber PKE

Before presenting the individual algorithms that define the CRYSTALS-KYBER scheme, it is essential to first introduce the underlying definitions and assumptions on which the construction relies.

- All computations in the scheme are carried out over integers modulo q and polynomials from the ring $R_{\mathsf{q}} = \mathbb{Z}_{\mathsf{q}}[x]/(x^n + 1)$.

- For further considerations, we define the notion of symmetric modular reduction. For an integer n, its symmetric representation modulo q is denoted as

$$\mathsf{n}^* = \mathsf{n} \bmod^{\mathsf{s}} \mathsf{q},$$

where

$$\mathsf{n}^* \in \begin{cases} \left(-\frac{\mathsf{q}}{2}, \frac{\mathsf{q}}{2}\right] & \text{if } \mathsf{q} \text{ is even,} \\ \left(-\frac{\mathsf{q}-1}{2}, \frac{\mathsf{q}-1}{2}\right] & \text{if } \mathsf{q} \text{ is odd.} \end{cases}$$

When the symmetric representation is not required, we apply the standard modular reduction: $\mathsf{n}^* = \mathsf{n} \bmod^{\mathsf{s}} \mathsf{q}$, where $\mathsf{n}^* \in [0, \mathsf{q})$.

- The notation $\lceil x \rfloor$ denotes the operation of rounding the rational number x to the nearest integer.

- A key part of the scheme is the generation of polynomials with *small* coefficients, which are used in constructing the secret key and the error vectors. For this, we define a centered binomial distribution B_η, where η is a positive integer. Sampling from B_η proceeds as follows:

1. $\{a_i, b_i\} \xleftarrow{\$} \{0,1\}^2 \quad \text{for} \quad i \in \{0, \dots, \eta - 1\},$
2. return $\sum_{i=0}^{\eta-1} (a_i - b_i).$

Sampling a polynomial v with coefficients from B_η is denoted $v \leftarrow \beta_\eta$. This distribution, similar to a discrete Gaussian centered at zero, enables the efficient construction of small error vectors.

- The Kyber scheme employs compression and decompression functions primarily to reduce the size of transmitted parameters. Additionally, compression is essential in recovering the message during decryption.

The compression function $\mathsf{Compr}_{\mathsf{q}}(x, \mathsf{d})$, where $x \in \mathbb{Z}_{\mathsf{q}}$ and $\mathsf{d} < \lceil \log_2 \mathsf{q} \rceil$, maps x to an integer in $\{0, \dots, 2^{\mathsf{d}} - 1\}$. The decompression function partially reverses this process, possibly introducing a bounded error:

$$x' = \mathsf{Decompr}_{\mathsf{q}}(\mathsf{Compr}_{\mathsf{q}}(x, \mathsf{d}), \mathsf{d}),$$

such that

$$|x' - x| \bmod^{\mathsf{s}} \mathsf{q} \leq \left\lceil \frac{\mathsf{q}}{2^{\mathsf{d}+1}} \right\rfloor.$$

The functions are defined as

$$\mathsf{Compr}_{\mathsf{q}}(x, \mathsf{d}) = \left\lceil \left(\frac{2^{\mathsf{d}}}{\mathsf{q}}\right) \cdot x \right\rfloor \bmod 2^{\mathsf{d}},$$
$$\mathsf{Decompr}_{\mathsf{q}}(x, \mathsf{d}) = \left\lceil \left(\frac{\mathsf{q}}{2^{\mathsf{d}}}\right) \cdot x \right\rfloor.$$

When applied to polynomials or vectors of polynomials, these functions are evaluated coefficient-wise.

- The notation $\mathsf{Sample}()$ denotes a deterministic function which, given a binary seed x, outputs y of arbitrary length according to a specified distribution $\mathcal{S}$, written as $y \leftarrow \mathcal{S} := \mathsf{Sample}(x)$.

We now describe the public-key encryption scheme $\mathsf{KyberCPA} = (\mathsf{KGen}^{\mathsf{KCPA}}, \mathsf{Enc}^{\mathsf{KCPA}}, \mathsf{Dec}^{\mathsf{KCPA}})$. The parameters are: k, $\mathsf{d_u}$, $\mathsf{d_t}$, $\mathsf{d_v}$, η, and the security parameter $n = 256$. The message space is $\mathcal{M} = \{0,1\}^n$. A message $\mathsf{m} \leftarrow \mathcal{M}$ is interpreted as a polynomial of degree $n-1$ with coefficients in $\{0,1\}$.

- $\mathsf{KGen}^{\mathsf{KCPA}}(1^n)$ – key generation algorithm, producing a pair $(\mathsf{pk}, \mathsf{sk})$:
    1. Sample two seeds $\rho, \sigma \leftarrow \{0,1\}^n$,
    2. Generate a matrix $A^{k \times k} \in R_{\mathsf{q}}$ from seed ρ: $A^{k \times k} \leftarrow R_{\mathsf{q}} := \mathsf{Sample}(\rho)$,
    3. Generate two vectors of k polynomials from β_η: $(\mathsf{s}, \mathsf{e}) \leftarrow \beta_\eta^k \times \beta_\eta^k := \mathsf{Sample}(\sigma)$,
    4. Compute $\mathsf{t} = A \cdot \mathsf{s} + \mathsf{e}$, then compress with parameter $\mathsf{d_t}$: $\mathsf{t} = \mathsf{Compr}_{\mathsf{q}}(A \cdot \mathsf{s} + \mathsf{e}, \mathsf{d_t})$,
    5. Output keys $(\mathsf{pk}, \mathsf{sk}) = ((\mathsf{t}, \rho), \mathsf{s})$.
- $\mathsf{Enc}^{\mathsf{KCPA}}(\mathsf{pk}, \mathsf{m} \in \mathcal{M})$ – encryption algorithm, taking public key $\mathsf{pk} = (\mathsf{t}, \rho)$ and message $\mathsf{m} \in \mathcal{M}$, outputs ciphertext $\mathsf{c} = (u, \mathsf{v})$:
    1. Sample randomness $\mathsf{rand} \leftarrow \{0,1\}^n$,
    2. Decompress t: $\mathsf{t}' = \mathsf{Decompr}_q(\mathsf{t}, \mathsf{d_t})$,
    3. Reconstruct matrix $A \leftarrow R_{\mathsf{q}} := \mathsf{Sample}(\rho)$,
    4. Generate vectors $\mathsf{r}, \mathsf{e}_1 \in \beta_\eta^k$ and polynomial $\mathsf{e}_2 \leftarrow \beta_\eta$ from rand: $(\mathsf{r}, \mathsf{e}_1, \mathsf{e}_2) \leftarrow \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$
    5. Compute $\mathsf{u} = \mathsf{Compr}_{\mathsf{q}}(A^T \cdot \mathsf{r} + \mathsf{e}_1, \mathsf{d_u})$,
    6. Compute $\mathsf{v} = \mathsf{Compr}_{\mathsf{q}}(\mathsf{q}, \mathsf{d_v})$,
    7. Output ciphertext $\mathsf{c} = (\mathsf{u}, \mathsf{v})$.
- $\mathsf{Dec}^{\mathsf{KCPA}}(\mathsf{sk}, \mathsf{c})$ – decryption algorithm, taking secret key $\mathsf{sk} = \mathsf{s}$ and ciphertext $\mathsf{c} = (\mathsf{u}, \mathsf{v})$, outputs recovered message m':
    1. Decompress u: $\mathsf{u}' = \mathsf{Decompr}_q(\mathsf{u}, \mathsf{d_u})$,
    2. Decompress v: $\mathsf{v}' = \mathsf{Decompr}_q(\mathsf{v}, \mathsf{d_v})$,
    3. Compute $\mathsf{m}' = \mathsf{Compr}_{\mathsf{q}}(\mathsf{v} - \mathsf{s}^T \cdot \mathsf{u}, 1)$,
    4. Output m'.

The authors of (Joppe, et al., 2017) prove that the above scheme is secure against chosen-plaintext attacks. However, as noted earlier, such a security level is insufficient for most practical applications. Modern cryptographic protocols require security against chosen-ciphertext attacks, which also applies to the Kyber scheme.

Key Encapsulation Mechanism

The CRYSTALS-KYBER scheme, in its CCA-secure variant, is designed in the form of a **Key Encapsulation Mechanism** (KEM). A KEM scheme consists of three algorithms $\mathcal{B}_{\mathsf{KEM}} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$:

1. The key generation algorithm $\mathsf{KGen}(1^n)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
2. The encapsulation algorithm $\mathsf{Encaps}(\mathsf{pk})$ returns a pair (K, c), where K denotes the established shared key and c its encapsulation.

3. The decapsulation algorithm $\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ returns the shared key K or a designated error symbol.

KEM schemes are not intended for direct encryption of messages. Instead, they enable the secure establishment of a shared key, which can then be used in other cryptographic schemes (e.g., symmetric encryption). CRYSTALS-KYBER has been approved as the standardized key encapsulation mechanism for post-quantum cryptography.

CRYSTALS-Kyber KEM

To construct a scheme secure against CCA attacks in the presence of quantum adversaries, the designers of KYBER employed the Fujisaki–Okamoto transformation in the variant introduced in (Targhi & Unruh, 2015) (see also Section 4, where this transformation and its security proof are discussed), together with the key encapsulation mechanism construction described in (Hofheinz, et al., 2017).

The key encapsulation scheme $\mathsf{KyberCCA} = (\mathsf{KGen}^{\mathsf{KCCA}}, \mathsf{Encaps}^{\mathsf{KCCA}}, \mathsf{Decaps}^{\mathsf{KCCA}})$ employs hash functions $\mathsf{G}: \{0,1\}^{2n} \to \{0,1\}^{2n}$, $\mathsf{H}: \{0,1\}^{n} \to \{0,1\}^{n}$, and $\mathsf{H}': \{0,1\}^{2n} \to \{0,1\}^{2n}$ (the latter used for deriving the shared key). It operates as follows:

- $\mathsf{KGen}^{\mathsf{KCCA}}(1^n)$ key generation algorithm producing a key pair $(\mathsf{pk}, \mathsf{sk})$:
    1. sample $\mathsf{z} \leftarrow \{0,1\}^n$,
    2. invoke the key generation algorithm $\mathsf{KGen}^{\mathsf{KCPA}}(1^n)$ to produce $(\mathsf{pk}^{\mathsf{KCPA}}, \mathsf{sk}^{\mathsf{KCPA}})$,
    3. output the key pair $(\mathsf{pk}, \mathsf{sk})$, where $\mathsf{pk} = \mathsf{pk}^{\mathsf{KCPA}}$ and $\mathsf{sk} = (\mathsf{sk}^{\mathsf{KCPA}}, \mathsf{z})$.
- $\mathsf{Encaps}^{\mathsf{KCCA}}(\mathsf{pk})$ — encapsulation algorithm taking as input a public key $\mathsf{pk} = (\mathsf{t}, \rho)$ and outputting a ciphertext $\mathsf{c} = (u, \mathsf{v})$ together with a key K:
    1. sample $\mathsf{m} \leftarrow \{0,1\}^n$,
    2. compute $(\widehat{\mathsf{K}}, \mathsf{r}) \leftarrow \mathsf{G}(\mathsf{H}(\mathsf{pk}), \mathsf{m})$,
    3. invoke the encryption algorithm $\mathsf{Enc}^{\mathsf{KCPA}}$ deterministically with randomness r : $\mathsf{c} \leftarrow \mathsf{Enc}^{\mathsf{KCPA}}(\mathsf{pk}, \mathsf{m}; \mathsf{r})$,
    4. compute $\mathsf{K} \leftarrow \mathsf{H}'(\widehat{\mathsf{K}}, \mathsf{H}(\mathsf{c}))$,
    5. output the ciphertext c and key K.
- $\mathsf{Decaps}^{\mathsf{KCCA}}(\mathsf{sk},\mathsf{pk},\mathsf{c})$ - decapsulation algorithm taking as input a secret key $\mathsf{sk} = (\mathsf{s}, \mathsf{z})$, a public key $\mathsf{pk} = (\mathsf{t}, \rho)$, and a ciphertext $\mathsf{c} = (\mathsf{u}, \mathsf{v})$, returning the key K:
    1. use the decryption algorithm $\mathsf{Dec}^{\mathsf{KCPA}}$ to recover m: $\mathsf{m}' = \mathsf{Dec}^{\mathsf{KCPA}}(\mathsf{sk}, \mathsf{c})$,
    2. compute $(\widehat{\mathsf{K}}', \mathsf{r}') \leftarrow \mathsf{G}(\mathsf{H}(\mathsf{pk}), \mathsf{m}')$,
    3. invoke the encryption algorithm $\mathsf{Enc}^{\mathsf{KCPA}}$ deterministically with randomness r' : $\mathsf{c}' \leftarrow \mathsf{Enc}^{\mathsf{KCPA}}(\mathsf{pk}, \mathsf{m}'; \mathsf{r}')$,
    4. if $\mathsf{c} = \mathsf{c}'$, output $\mathsf{K} \leftarrow \mathsf{H}'(\widehat{\mathsf{K}}', \mathsf{H}(\mathsf{c}))$,
    5. if $\mathsf{c} \neq \mathsf{c}'$, output $\mathsf{K} \leftarrow \mathsf{H}'(\mathsf{z}, \mathsf{H}(\mathsf{c}))$.

Implementation and Experimental Results

This section presents the design and results of the implementation of the KyberCPA and KyberCCA schemes. The objective of this implementation is not to deliver an optimized version of the CRYSTALS-KYBER scheme, but rather to evaluate the effect of applying the Fujisaki–Okamoto (FO) transformation within its construction. In this setting, the role of CRYSTALS-KYBER is that of a representative framework used to assess the transformation, rather than the principal object of study itself.

For this reason, the implementation was carried out in a straightforward manner, without the use of additional optimization techniques, in order to reproduce as faithfully as possible the mathematical structure and logical flow of the KyberCPA and KyberCCA schemes.

The entire implementation was developed in the SageMath environment, an open source platform for symbolic and numerical computations that integrates seamlessly with the Python programming language. This environment was chosen because of its extensive support for algebraic operations, which facilitates a transparent representation of computations on polynomials, matrices, finite fields, and modular structures. Moreover, its high level syntax and flexibility enabled a direct and faithful translation of the theoretical assumptions underlying the schemes into executable code.

It should be emphasized, however, that SageMath **is not optimized for low-level performance**. As an interpreted system running on top of Python, it introduces significant overhead, stemming from limited control over data representation and the absence of advanced optimization techniques, such as automatic avoidance of data copying, SIMD operations, or manual management of buffers and CPU cache.

For instance, in the conducted tests, the encapsulation time was around 1.29 seconds, whereas an analogous Python implementation (kyber-py), run on a modern processor (Intel i7-9750H), achieved times on the order of 2.9 milliseconds. This represents more than a 400-fold slowdown, resulting not from the algorithm itself, but from the characteristics of the environment.

Therefore, it must be stressed that the **performance results presented here are purely auxiliary** and serve only to support the analysis of the correctness of the encryption scheme construction with the applied FO transformation based on Kyber. All performance-related observations should be regarded as indicative only and not binding in the context of real-world, production-grade applications.

Implementation Details

The implementation was carried out in the SageMath environment, with the goal of faithfully reproducing the structure of KyberCPA and KyberCCA without applying additional optimizations. Polynomials are represented as arrays of coefficients, enabling precise control over arithmetic operations and a direct implementation of the NTT transformation.

Matrices and vectors of polynomials are generated from random seeds using SHAKE256 as an extendable output function, ensuring the required number of coefficients. Polynomials with small coefficients are sampled from the distribution B_η, following the scheme's specification.

Hash functions were instantiated as follows: G with SHA3-512, H with SHA3-256, and H' with SHAKE256. Compression and decompression procedures were implemented exactly as defined in the Kyber specification, applied coefficient-wise to polynomials and vectors.

The program accepts as input the modulus $q = 7681$, security parameter $n = 256$, dimension parameter $k = 4$, error parameter $\eta = 3$, compression parameters $\mathsf{d_u} = 11$, $\mathsf{d_v} = 3$, $\mathsf{d_t} = 11$, and primitive roots $\psi = 62$, $\omega = 3844$ used in NTT/INTT computations.

Based on these components, pseudocode was developed for the key generation, encryption, and decryption algorithms (or equivalently, encapsulation and decapsulation in the KEM variant).

Input: q, n, k, dt
Output: sk = s, pk = (t, rho)

```
1       rho ← {0,1}^n
2       sigma ← {0,1}^n
3       A ← Sample(rho, q, n, k)
4       s, e ← Sample(eta, q, n, k)
5       t ← INTT(NTT(A, s))+e
6       compress t
7       return (sk = s, pk = (t, rho))
```

Fig 1. Algorithm KyberCPA.KGen

Input: pk = (t, rho), q, n, k, eta, m, dt, du, dv, ran=None
Output: c = (u, v)

```
1       decompress t
2       if ran = None then
3            ran ← {0,1}^n
4       A ← Sample(rho, q, n, k)
5       r,e1,e2 ← Sample(ran, n, k, eta)
6       u ← INTT(NTT(A^T, r))+e1
7       compress u
8       v ← INTT(NTT(t^T, r)+e2 + round(q/2)*m
9       compress v
10      return c = (u, v)
```

Fig 2. KyberCPA.Enc

Input: sk = s, c = (u, v), q, n, du, dv
Output: m

```
1       decompress u
2       decompress v
3       m ← v-INTT(NTT(s^T, u))
4       compress m
5       return m
```

Fig 3. KyberCPA.Dec

Input: q, n, k, dt
Output: $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PKE}}, \mathsf{s}), \mathsf{pk} = \mathsf{pk}_{\mathsf{PKE}}$

```
1       generate (sk_PKE,pk_PKE) ← KyberCPA.KGen(q,n,k,dt)
2       sample z ← {0,1}^n
3       return (sk = (sk_PKE,s), pk = (pk_PKE))
```

Fig 4. KyberCCA.KGen

Input: pk = (t, rho), q, n, k, du, dv, dt, eta
Output: c = (u, v), K

```
1       m ← {0,1}^n
2       K1 || ran ← G(H(pk) || m)
3       c ← KyberCPA.Enc(pk, q, n, k, eta, m, dt, dt, dv, ran)
4       K ← H'(K1 || H(c))
5       return c, K
```

Fig 5. KyberCCA.Encaps

Input: sk = (z, s), pk = (t, rho), c = (u, v), q, n, k, du, dt, dv, eta
Output: K

```
m1 ← KyberCPA.Dec(sk, c, q, n, du, dv)
K11 || ran1 ← G(H(pk) || m1)
u1, v1 ← KyberCPA.Enc(pk,q,n,k,eta,m1,dt,du,dv,ran1)
if u1 = u then
     if v1 = v then
          K ← H'(K11 || H(c))
     if v1 != v then
          K ← H'(z || H(c))
if u1 1= u then
     K ← H'(z || H(c))
return K
```

Fig 6. KyberCCA.Decaps

Implementation Results

To evaluate the performance impact of applying the Fujisaki–Okamoto transform, 1000 independent measurements were conducted for each operation in KyberCPA (PKE) and KyberCCA (KEM). Tables 4–6 summarize the average execution times, standard deviations, and observed extrema for key generation, encryption/encapsulation, and decryption/decapsulation.

Table 4: Key generation times for KyberCPA (PKE) and KyberCCA (KEM).

Metric	PKE	KEM
Mean [s]	0.9007	0.9032
Std. deviation [s]	0.0191	0.0283
Maximum [s]	0.9715	1.1026
Minimum [s]	0.8755	0.8670

Table 5: Encryption (PKE) vs. encapsulation (KEM).

Metric	Encryption	Encapsulation
Mean [s]	1.2930	1.2957
Std. deviation [s]	0.0219	0.0199
Maximum [s]	1.6457	1.4523
Minimum [s]	1.2565	1.2589

Table 6: Decryption (PKE) vs. decapsulation (KEM).

Metric	Decryption	Decapsulation
Mean [s]	0.3226	1.6179
Std. deviation [s]	0.0077	0.0184
Maximum [s]	0.4074	1.7468
Minimum [s]	0.3110	1.5707

The results indicate that:

- Key generation times remain virtually identical in both schemes, as the FO transform does not alter this phase apart from adding a single random value to the secret key.
- Encapsulation in KyberCCA shows almost no overhead compared to encryption in KyberCPA, confirming that additional hash computations are negligible in practice.
- Decapsulation is the dominant cost: its average runtime is roughly the sum of encryption and decryption in KyberCPA, due to the need for an additional re-encryption step required by the FO transform.

Conclusions

In this work we analyzed the Fujisaki–Okamoto (FO) transform as a mechanism for upgrading public key encryption schemes to chosen ciphertext (CCA) security, with particular emphasis on the setting of quantum adversaries. On the theoretical side, we reviewed ROM and its quantum counterpart (QROM), highlighting the extent to which the security guarantees of the FO transform carry over to both frameworks. On the practical side, we implemented and evaluated two variants of the CRYSTALS–KYBER scheme: the CPA secure version and the CCA secure version obtained through the application of the FO transform.

Our performance evaluation shows that the FO transform introduces only negligible overhead in key generation and encapsulation, but imposes a significant cost in decapsulation. In particular, the average runtime of decapsulation in Kyber–FO is comparable to the combined cost of encryption and decryption in Kyber–CPA, primarily due to the additional encryption step required by the transform. These findings confirm that the FO transform remains a robust and conceptually simple method for strengthening the security of lattice based cryptography, though at the expense of increased computational effort in decapsulation.

As future work, it would be valuable to investigate modifications of CRYSTALS–KYBER that achieve full compatibility with the QROM framework. Such refinements hold promise for the development of quantum resilient schemes that combine rigorous theoretical guarantees with practical efficiency, and they will constitute the focus of our subsequent research.

References

- Ajtai, M., 1996. Generating hard instances of lattice problems (extended abstract). Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing.
- Boneh, D. et al., 2011. Random oracles in a quantum world. International conference on the theory and application of cryptology and information security.
- Boneh, D. & Shoup, V., 2023. A Graduate Course in Applied Cryptography.. s.l.:s.n.
- Fujisaki, E. & Okamoto, T., 1999. Secure Integration of Asymmetric and Symmetric Encryption Schemes. Springer Berlin Heidelberg.
- Guo, F., Susilo, W. & Mu, Y., 2018. Introduction to Security Reduction.. s.l.:s.n.

- Hofheinz, D., Hövelmanns, K. & Kiltz, E., 2017. A Modular Analysis of the Fujisaki-Okamoto Transformation. Cryptology ePrint Archive, Paper 2017/604.
- Joppe, B. et al., 2017. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Paper 2017/634.
- Katz, J. & Lindell, Y., 2015. Introduction to Modern Cryptography.. s.l.:s.n.
- Lippert, J., Blömer, J. & Domik, G., 2014. The Fujisaki-Okamoto Transformation.. s.l.:s.n.
- Lyubashevsky, V., Peikert, C. & Regev, O., 2010. On Ideal Lattices and Learning with Errors over Rings. Springer Berlin Heidelberg.
- National Institute of Standards and Technology, 2023. Post-Quantum Cryptography: Current State and Quantum Mitigation Strategies. s.l.:s.n.
- Regev, O., 2005. On lattices, learning with errors, random linear codes, and cryptography. Association for Computing Machinery.
- Satriawan, A., Mareta, R. & Lee, H., 2024. A Complete Beginner Guide to the Number Theoretic Transform (NTT). Cryptology ePrint Archive, Paper 2024/585.
- Targhi, E. E. & Unruh, D., 2015. Quantum Security of the Fujisaki-Okamoto and OAEP Transforms.. Cryptology ePrint Archive, Paper 2015/1210.
- Unruh, D., 2014. Quantum position verification in the random oracle model.. Cryptology ePrint Archive, Paper 2014/118.
- Zhandry, M., 2012. Secure Identity-Based Encryption in the Quantum Random Oracle Model.. Cryptology ePrint Archive, Paper 2012/076.