mathematics

MDPI

Article

Compiled Constructions towards Post-Quantum Group Key Exchange: A Design from Kyber

José Ignacio Escribano Pablos [1,2], María Isabel González Vasco [1], Misael Enrique Marriaga [1,*] and Ángel Luis Pérez del Pozo [1]

[1] MACIMTE, U. Rey Juan Carlos, 28933 Móstoles, Spain; joseignacio.escribano.pablos.next@bbva.com (J.I.E.P.); mariaisabel.vasco@urjc.es (M.I.G.V.); angel.perez@urjc.es (Á.L.P.d.P.)

[2] BBVA Next Technologies, 28050 Madrid, Spain

* Correspondence: misael.marriaga@urjc.es

Received: 15 September 2020; Accepted: 13 October 2020; Published: 21 October 2020

Abstract: A group authenticated key exchange (GAKE) protocol allows a set of parties belonging to a certain designated group to agree upon a common secret key through an insecure communication network. In the last few years, many new cryptographic tools have been specifically designed to thwart attacks from adversaries which may have access to (different kinds of) quantum computation resources. However, few constructions for group key exchange have been put forward. Here, we propose a four-round GAKE which can be proven secure under widely accepted assumptions in the *Quantum Random Oracle Model*. Specifically, we integrate several primitives from the so-called Kyber suite of post-quantum tools in a (slightly modified) compiler from Abdalla et al. (TCC 2007). More precisely, taking as a starting point an IND-CPA encryption scheme from the Kyber portfolio, we derive, using results from Hövelmanns et al. (PKC 2020), a two-party key exchange protocol and an IND-CCA encryption scheme and prove them fit as building blocks for our compiled construction. The resulting GAKE protocol is secure under the Module-LWE assumption, and furthermore achieves authentication without the use of (expensive) post-quantum signatures.

Keywords: post-quantum cryptography; group authenticated key exchange; Module-LWE; Kyber

1. Introduction

The search for cryptographic primitives that will remain secure once quantum computing is a reality has been on going for over twenty years. Noticeably, in the last few years this search has gained greater attention from academia and industry, especially since the US National Institute of Standards and Technology (NIST) launched a competition towards standardizing quantum-resistant (also called *post-quantum*) public-key cryptographic algorithms in 2017. While this competition is focused on constructions for public key encryption, two party key establishment and digital signatures, research towards different post-quantum primitives has also been aroused as a side effect.

Group key establishment protocols (GKE) are fundamental cryptographic constructions. Indeed, for many real life applications of information technologies, the crucial starting point is establishing a "secure session", i.e., setting confidential communication channels among users. GKE protocols allow a group of $n \geq 2$ users, interacting through an insecure communication network, to establish a common known high entropy secret that can be used to secure their subsequent communication. Typically, once this secret has been agreed upon, tools from symmetric cryptography can be used to attain confidentiality,

and thus the communication network is understood as secure for confidential transmissions within the group of honest users. Using a GKE in this setting clearly outperforms the use of two party solutions, as establishing different session keys for every pair of participants (e.g., using a two party key exchange) would force each participant to store a large number of keys. Moreover, every message intended for the whole group should be encrypted multiple times ($n-1$) with different keys, while GKE can be used in a broadcast fashion (as messages are processed the same way for each group member). There might, however, be no way of assessing origin and integrity of messages. In this case, when authenticated channels are not available, protocols pursuing this goal—GAKE protocols—get way more involved and often need to rely on an external public key infrastructure to be able to authenticate legitimate group members, frequently adding a significant cost to the constructions.

Related work. Several group key exchange protocols which can be considered to resist quantum attacks have been proposed so far. Fujioka et al. [1] presented two one-round authenticated protocols, whose security is based on a certain algebraic-geometric problem related to the problem of finding a so-called *isogeny mapping* between two supersingular elliptic curves with the same number of points.

Other protocols use lattice problems as a base. For instance, Apon et al. [2] constructed a three-round unauthenticated protocol proven secure under the so-called *ring learning with errors* (RLWE) assumption. This scheme may be transformed into an authenticated one by using the Katz and Yung compiler [3]. However, the resulting protocol has one additional round of communication and each message that is sent must be signed, adding a significant computation and communication overhead if a post-quantum signature scheme is employed. Using the same problem as a base, Choi et al. [4] built on [3] and proposed three group protocols: the first is unauthenticated, the second adds authentication, and the third is, in addition, dynamic. The second one, STAG, is a three-round authenticated protocol in which each user computes two signatures.

Finally, we have *compilers* which produce a quantum-resistant group authenticated key exchange (GAKE) from simpler post-quantum primitives. Persichetti et al. [5] presented a three-round protocol constructed from a key encapsulation mechanism (KEM) and a signature scheme; each user needs to compute only one signature. González Vasco et al. [6] introduced a two-round password GAKE protocol derived from a KEM and a message authentication code (MAC). However, in this construction, security holds in the so-called *future-quantum* scenario, where the adversary is assumed to have access to quantum computation only after the protocol execution is completed.

Our contribution. In this work, we take the so-called `Kyber` family [7] of post-quantum cryptographic tools and use it as a base for a GAKE design. More precisely, our construction is a compiled system using Abdalla et al.'s [8] as design frame. From the results of Hövelmanns et al. [9], we assess that both a suitable commitment scheme and a secure two-party AKE can be obtained from the encryption scheme `Kyber.CPA`$'$ (this result was hinted, yet not explicitly proven by Hövelmanns et al. [9]). As far as we are aware, our instantiation is the first group authenticated key exchange protocol which provides post-quantum security guarantees based solely on the so-called Module-LWE assumption, doing without (often unaffordably expensive) post-quantum signatures.

Our GAKE: overview. The workflow of our construction is depicted in Figure 1. Our construction relies on Abdalla et al.'s compiler [8] that requires a two-party AKE and a commitment scheme and we need both building blocks to fulfill post-quantum security. To achieve post-quantum security, we apply the Kyber family and its derived tools (see green rectangle in Figure 1).

`Kyber` [7] is a KEM based on lattices whose security relies on Module-LWE assumption (Definition 1), claimed to be post-quantum secure. Our GAKE inherits the Module-LWE assumption. The two-party AKE and the commitment scheme are derived from the initial IND-CPA PKE in [7] named `Kyber.CPA`$'$.

The two-party AKE (named Kyber.2AKE) is the result of applying the transformation $\mathrm{FO}_{\mathrm{AKE}}$ [9] to `Kyber.CPA`′. Finally, a commitment scheme can be achieved from any IND-CCA PKE, as pointed out in [8]. In our construction, we turn `Kyber.CPA`′ into a KEM applying the $\mathrm{FO}^{\not\perp}_m$ transformation [9] obtaining Kyber$^{\not\perp}$, which is transformed into an IND-CCA PKE (Kyber.PKE) as a result of [10].

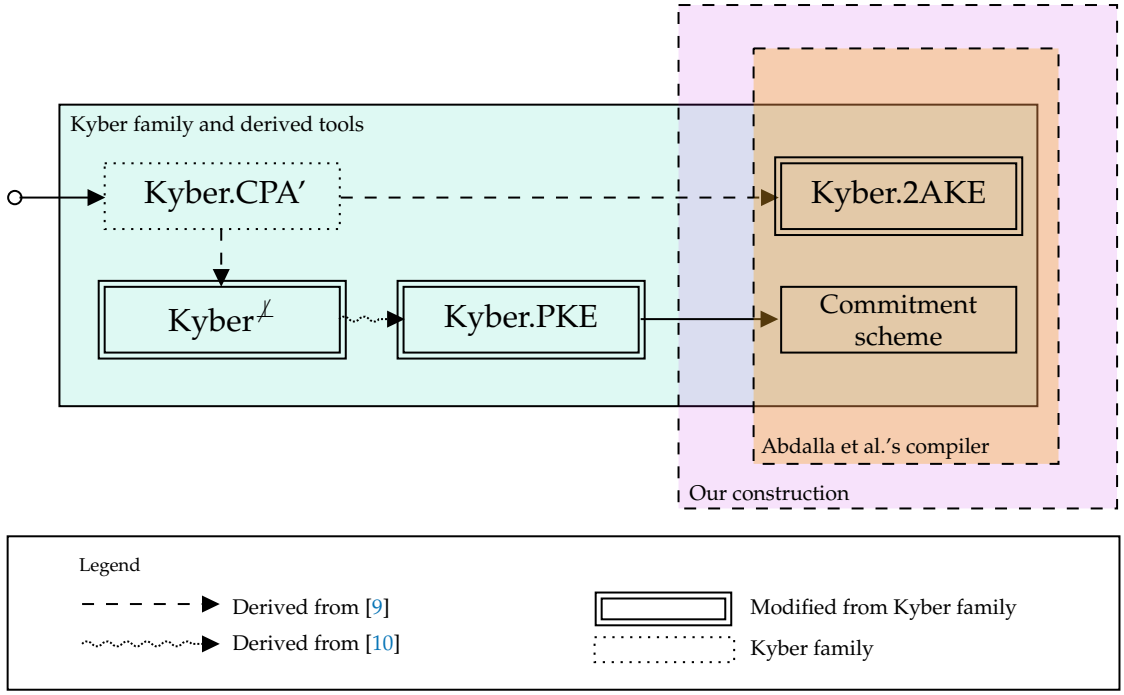

Figure 1. Workflow of our construction.

Comparison with other schemes. We present two tables that summarize some features of other GAKE schemes with quantum-resistance and compare them with our proposal.

Table 1 summarizes some parameters related to the performance of the schemes. The number of communication rounds is one of the most important parameters when dealing with GAKE protocols. In addition, for each scheme, we point out whether the use of post-quantum signatures is avoided, which is a nice feature, as this kind of signatures are usually expensive in terms of both computation and size. Note that the scheme in [2] does not use post-quantum signatures but is unauthenticated. Finally, we include the *total* number of messages sent throughout an execution involving n parties, pointing out whether messages are broadcasted or just sent point-to-point (PtP) (i.e., from one party to another).

Table 1. Some efficiency parameters for GKE/GAKE protocols claimed to be quantum-resistant.

Protocol	# Rounds	Avoids PQ-Sign.	# Broadcast Messages	# PtP Messages
n-UM [1]	1	Yes	n	0
BC n-DH [1]	1	Yes	n	0
Apon et al. [2]	3	Yes (but is unauth.)	$2n+1$	0
STAG [4]	3	No	$2n+1$	0
Pers. et al. [5]	3	No	n	$2n$
Gonz. et al. [6]	2	Yes	n	n^2-n
This work	4	Yes	$2n$	$2n$

Table 2 is focused on security issues. In the first column, we include the type of assumption (isogeny/lattice) the security of the scheme is based on. In the second column, we state in which of the idealized models the security claim and corresponding proof hold: either in the Random Oracle Model (ROM) or the Quantum Random Oracle Model (QROM). The latter is stronger than the former because, as discussed in Section 2, it assumes a more powerful adversary. Next, it is specified if quantum resistant features hold in a future quantum (FutQ) or post-quantum (PostQ) scenario. The latter, where it is assumed that the adversary has access to quantum computation during protocol executions, is preferable to the former, where key secrecy is only guaranteed against adversaries that cannot make quantum computations during protocol executions but have access to this option at some point in the future. Finally, we indicate if the key exchange is authenticated. Note that the compilers [5,6] have a special treatment, as the assumption type and the model depend on the underlying post-quantum tools used to implement them.

Table 2. Security of GKE/GAKE protocols claimed to be quantum-resistant.

Protocol	Assumption Type	Model	FutQ/PostQ	Authent.
n-UM [1]	Isogeny	QROM	PostQ	Yes
BC n-DH [1]	Isogeny	ROM	PostQ	Yes
Apon et al. [2]	Lattice	ROM	PostQ	No
STAG [4]	Lattice	ROM	PostQ	Yes
Pers. et al. [5]	Compiler	No RO added	PostQ	Yes
Gonz. et al. [6]	Compiler	No RO added	FutQ	Yes
This work	Lattice	QROM	PostQ	Yes

Seeing the two comparison tables, it seems clear that the only scheme outperforming our construction is n-UM [1], which is based on the isogeny paradigm. However, it is fair to say that lattice-based constructions such as ours seem somewhat more promising in this field, considering the recent outcome of the Third Round of the NIST competition for standardizing post-quantum tools. While several lattice-based constructions made it to this last round, no isogeny-based scheme is in the final (and only one proposal, SIKE [11], is considered as alternative for replacing finalists that may be discarded in the last phase).

Paper Roadmap. We start with a brief outline of the preliminaries in Section 2, where we introduce Abdalla et al.'s compiler from [8] and comment on the basics of post-quantum security. Further, we explain in Section 3 how to derive building blocks for our construction (AKE and a commitment scheme) from the `Kyber` family. In particular, we use the results from [9] to prove that we can obtain both a suitable commitment scheme and a secure two-party AKE from the encryption scheme `Kyber.CPA`′. Our compiled construction is then described and proven secure in Section 4, where we also make explicit the security model used. We conclude this contribution with a brief conclusion.

2. Preliminaries

2.1. Abdalla et al.'s Compiler

Here, we describe a compiler constructed by Abdalla et al. in [8], which enables the derivation of a group authenticated key establishment protocol `GAKE` from an arbitrary two-party key establishment `2AKE`. The compiler does not rely on further authentication techniques than those used in `2AKE`, nor on further idealization assumptions. Moreover, if `2AKE` requires r rounds of communication, then `GAKE` requires $r+2$ rounds.

Let $\mathcal{P}$ be the set of users that can participate in the protocol `GAKE`. This set $\mathcal{P}$ is assumed to be of polynomial size. The set $\mathcal{G} = \{U_0, U_1, \ldots, U_{n-1}\} \subset \mathcal{P}$ denotes the set of $n > 2$ participants that wish to

establish a common session key. Each protocol participant $U_i \in \mathcal{G}$, $i = 0, \ldots, n-1$, may be involved in distinct, possibly parallel, executions of GAKE.

Since 2AKE is an authenticated key establishment protocol, it is assumed that long-term secrets required for 2AKE have been established during a trusted authentication phase. One of the following three cases is assumed:

- Each user $U_i \in \mathcal{G}$ owns a pair (pk_i, sk_i) consisting of a public key pk_i and a secret key sk_i, and all needed public keys may be distributed to all protocol participants during the initialization phase.
- Each pair of users $U_i, U_j \in \mathcal{G}$, $i \neq j$, shares a high entropy symmetric key, or the complete set of participants $\mathcal{G}$ shares one common secret (different instances of a user may hold different long-term secrets).
- Each pair of users $U_i, U_j \in \mathcal{G}$, $i \neq j$, shares a low entropy password. In this case, we assume a publicly available dictionary $\mathcal{D} \subseteq \{0,1\}^*$, from which passwords are chosen uniformly at random.

The compiler uses the following cryptographic tools:

1. **A non-interactive non-malleable commitment scheme** C that is perfectly binding and achieves *non-malleability for multiple commitments*.
2. **A collision-resistant pseudorandom function family** $\mathtt{F} = \{F^\ell\}_{\ell \in \mathbb{N}}$ with $F^\ell = \{F^\ell_\eta\}_{\eta \in \{0,1\}^L}$ to be indexed by a set $\{0,1\}^L$ of polynomial size, and two publicly known values v_0 and v_1 such that no ppt adversary can find two different indices $\lambda \neq \mu \in \{0,1\}^L$ such that $F^\ell_\lambda(v_j) = F^\ell_\mu(v_j)$, $j = 0, 1$.
3. **A hash function** H selected from a family of universal hash functions that maps the concatenation of bitstrings from $\{0,1\}^{kn}$ and the set of participants $\mathcal{G}$ onto $\{0,1\}^L$, where n is the number of participants in $\mathcal{G}$ and $k \in \mathbb{N}$.

With these ingredients, the compiler proceeds as depicted in Figure 2. Our proposed GAKE protocol uses a simplified version of the aforementioned compiler and builds on a post-quantum 2AKE. We describe it in detail in Section 4.

2.2. Security in a Post-Quantum Setting

When proving a certain cryptographic construction secure, it is necessary to depict a precise security model making explicit claims and assumptions that can be formally proven and verified. This is, however, not always the case in the post-quantum scenario, as quantum adversaries are often modeled in a very different fashion. Most often, constructions are substantiated on computational assumptions that explicitly state that an adversary is assumed not to be able to efficiently complete a certain computational task (e.g., decoding a word with respect to a certain partially known code or solving certain approximation problems in lattices). However, the way this quantum adversary is assumed to interact with other system-related idealizations (e.g., the oracles modeling information leakage or misuse) is often disregarded, while it may play a central role in a security proof. A paradigmatic example of this situation is the case of hash functions, typically modeled as so-called *random oracles*.

Random oracles are classically used in cryptography to model *idealized* hash functions, which are deterministic algorithms that select, for each new query, an output chosen uniformly at random from a certain given range. It is assumed that all users and processes from a certain system are given access to the same random oracles, which means that, for security proofs, if the real cryptographic environment is simulated for an adversary, all random oracle queries must be consistently answered with values that are indistinguishable from random (uniform). In the quantum setting, queries to a random oracle can be done in superposition, which complicates significantly the translations of many classical proofs into this new scenario.

Round $1 \sim r$: For each $i = 0, \ldots, n-1$, execute 2AKE with U_i and $U_{[i+1]}$, where $[k] = k \mod n$ (i.e., all indices are to be taken in a cycle). Thus, each $U_i \in \mathcal{G}$ holds two pairwise keys $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ shared with $U_{[i+1]}$ and $U_{[i-1]}$, respectively. Note that, for $i = 0, \ldots, n-1$, $\overrightarrow{K}_i = \overleftarrow{K}_{[i+1]}$.

Round $r+1$:

- **Computation**: Each U_i computes $X_i = \overrightarrow{K}_i \oplus \overleftarrow{K}_i$, and chooses a random r_i to compute a commitment $C_i = C(i, X_i, r_i)$.
- **Broadcast**: Each U_i broadcasts $M_i^1 = (U_i, C_i)$.

Round $r+2$:

- **Broadcast**: Each U_i broadcasts $M_i^2 = (U_i, X_i, r_i)$.
- **Verification**: Each U_i checks that $X_0 \oplus X_1 \oplus \cdots \oplus X_{n-1} = 0$ and the correctness of the commitments. If any one of these checks fails, then U_i terminates the protocol execution without computing a session key.
- **Computation**: Each U_i computes the $n-1$ values

$$\overleftarrow{K}_{[i-j]} = \overleftarrow{K}_{[i-j+1]} \oplus X_{[i-j]}, \quad j = 1, 2, \ldots, n-1.$$

Then, U_i defines a master key

$$K = \left(\overleftarrow{K}_0, \overleftarrow{K}_1, \cdots, \overleftarrow{K}_{n-1}, \mathcal{G} \right),$$

and sets the session key $\mathtt{sk}_i = F^{\ell}_{\mathrm{H}(K)}(v_1)$ and the session identifier $\mathtt{sid}_i = F^{\ell}_{\mathrm{H}(K)}(v_0)$, where $\ell \in \mathbb{N}$ is the security parameter.

Figure 2. Abdalla et al.'s compiler.

Following Hövelmanns et al. [9], in this work, we consider quantum adversaries that are given quantum access to the (offline) quantum random oracle involved in our design. More precisely, we need to make use of two basic properties of this so-called *quantum-accessible* random oracles:

- *Collision-freeness*. In [12], it is proven that the best quantum algorithm for finding a collision for a random function $H : \{0,1\}^n \mapsto \{0,1\}^n$ (i.e., a pair of distinct $x, x' \in \{0,1\}^n$ such that $H(x) = H(x')$) is $\widetilde{\mathcal{O}}(2^{\frac{n}{5}})$. (Notation $\widetilde{O}$ "wipes out" logarithmic factors in $\mathcal{O}$, namely, $f(n) \in \widetilde{\mathcal{O}}(h(n)) \iff \exists k \in \mathbb{N}$ s.t. $f(n) \in \mathcal{O}(h(n) log^k(h(n)))$). The analogous classical bound is $\mathcal{O}(2^{\frac{n}{2}})$. While being suboptimal in number of queries, this algorithm is the most efficient in terms of time complexity with small quantum memory. Thus, in the sequel, we may assume that (even for a quantum adversary) finding a collision pair for a quantum-accessible random oracle can only be done with negligible probability (this is used in Section 4.3.1).
- *Pseudorandomness*. Following again Hövelmanns et al. [9], we use the argument of Zhandry (see [13]) stating that no quantum algorithm, making at most q quantum queries to a quantum random oracle $\widehat{H}$ implementing a random function $\mathcal{H} : \{0,1\}^m \mapsto \{0,1\}^n$, can distinguish between $\widehat{H}$ and a random polynomial of degree $2q$ defined over the field $\mathbb{F}_{2^n}$. As a result, if the input to a quantum random oracle contains enough entropy, then the probability of distinguishing its output from a value chosen uniformly at random is negligible. In other words, when the input is unknown and chosen

uniformly at random, the fact that the random oracle can be queried in superposition is of no help in distinguishing the oracle's output from a randomly chosen element. This is used in the quantum random oracle proof from Section 4.3.

We strongly suggest the interested reader consult [14] for a comprehensive introduction to the quantum random oracle model.

3. Post-Quantum Primitives: `2AKE` and Commitment Scheme

In this section, we describe the post-quantum tools used in the construction of our `GAKE`, namely a two-party authenticated key exchange (AKE) and a commitment scheme. The relations between these tools are summarized in Figure 3.

In the first subsection, we describe `Kyber`'s public key encryption (PKE) scheme and state its security properties that are of importance to the construction of the primitives mentioned above.

In the second subsection, we detail how the `2AKE` is obtained from a generic construction proposed in [9], of two-message AKE provably secure in the quantum random oracle model (QROM) from PKE schemes that possess both *Disjoint Simulatability* (DS) (Definition 2) and IND-CPA security. In particular, we use a slight modification (called `Kyber.CPA`′) of the CPA-secure PKE scheme introduced in [7] as part of `Kyber`'s package submitted to NIST's post-quantum standardization effort. We describe the $\mathrm{FO}_{\mathrm{AKE}}$ transformation which turns a secure PKE into a secure AKE. This subsection ends by proving that `Kyber.CPA`′ is DS secure and, therefore, it is possible to construct an AKE secure in the QROM by applying the $\mathrm{FO}_{\mathrm{AKE}}$ to it.

The third subsection is devoted to the construction of the post-quantum commitment scheme mentioned in Section 2. It must be a non-interactive non-malleable commitment scheme that is perfectly binding and achieves non-malleability for multiple commitments. As pointed out in [8], this can be directly constructed from a public key encryption scheme which achieves the well-known IND-CCA security notion. To this end, we use another transformation described in [9], specifically $\mathrm{FO}_m^{\not\perp}$, which turns an IND-CPA and DS PKE into an IND-CCA KEM. Then, we recall that it is straightforward to obtain an IND-CCA PKE from an IND-CCA KEM. Putting everything together, we obtain the desired commitment scheme from `Kyber.CPA`′, the same primitive we use to construct the two-party AKE.

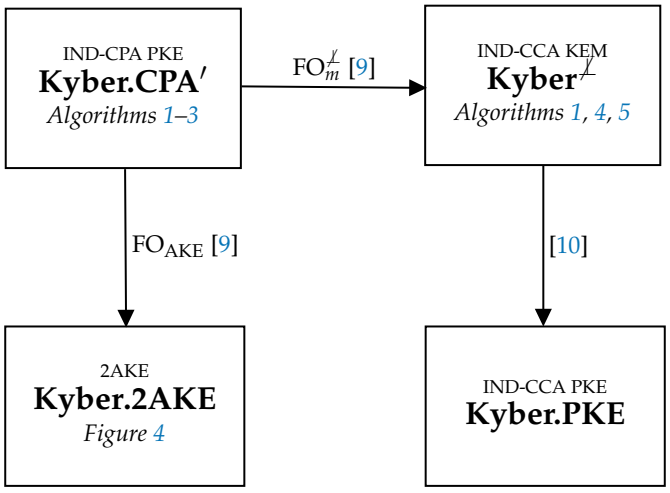

Figure 3. `Kyber` and derived tools.

3.1. Kyber's IND-CPA PKE

In this subsection, we describe the CPA-secure PKE scheme `Kyber.CPA` introduced in [7] as part of the *Cryptographic Suite for Algebraic Lattices (CRYSTALS)*, a package of cryptographic primitives submitted to NIST's post-quantum standardization effort. In fact, what we really describe and work with is a slightly modified version, also proposed in [7], called `Kyber.CPA`$'$.

First, we introduce some definitions needed to understand how the PKE has been constructed and summarize in Table 3 the notation used in the sequel. Then, we describe the key generation, encryption, and decryption algorithms used in `Kyber.CPA`$'$, as well as its CPA-security under the Module-LWE hardness assumption (Definition 1).

Denote by R the ring $\mathbb{Z}[X]/(X^n+1)$ and by R_q the ring $\mathbb{Z}_q[X]/(X^n+1)$, where $n = 2^{n'-1}$ such that X^n+1 is the $2^{n'}$th cyclotomic polynomial. As in [7], we fix the values for n, n' and q to 256, 9 and 7681.

For some positive integer η, define the centered binomial distribution B_η as follows ([7]):

$$\text{Sample } \{(a_i, b_i)\}_{i=1}^{\eta} \leftarrow (\{0,1\}^2)^\eta$$

$$\text{and output } \sum_{i=1}^{\eta}(a_i - b_i).$$

Table 3. Notation used for `Kyber.CPA`$'$.

Notation	Representation
Bold lower-case	Vectors with coefficients in R or R_q. All vector will be column vectors by default.
Regular font letter	Elements in R or R_q.
Bold upper-case	Matrices.
$s \leftarrow S$	If S is a set, s is chosen uniformly at random from S. If S is a distribution, s is chosen according to such distribution S.
$y \sim S := \mathtt{Sam}(x)$ where `Sam` is an eXtendable Output Function (XOF)	Value y that is distributed according to distribution S (or uniformly over a set S). This is a deterministic procedure.
$v \leftarrow \beta_\eta$, $\mathbf{v} \leftarrow \beta_\eta^k$	$v \in R$ is generated from a distribution where each of its coefficients are generated from B_η. A k-dimensional vector of polynomials $\mathbf{v} \in R^k$ can be generated according to the distribution β_η^k.
$\lceil \cdot \rfloor$	$\lceil \cdot \rfloor$ is the rounding function i.e., $\lceil x \rfloor = \left\lfloor x + \frac{1}{2} \right\rfloor$ where $x \in \mathbb{Q}$ and $\lfloor \cdot \rfloor$ is the floor function.
$r' = r \bmod^{\pm} \alpha$	For an even (respectively, odd) integer α, $r' = r \bmod^{\pm} \alpha$ is the unique element r' in the range $-\frac{\alpha}{2} < r \leq \frac{\alpha}{2}$ (respectively, $-\frac{\alpha-1}{2} < r \leq \frac{\alpha+1}{2}$) such that $r' = r \mod \alpha$.

The security assumption underlying `Kyber.CPA`$'$ is based on the hardness of the Module-LWE problem, which generalizes the Learning with Errors (LWE) problem. Learning with errors (LWE) is the computational problem of inferring a linear n-ary function f over a finite ring from given (slightly incorrect) samples $y_i = f(\mathbf{x}_i)$. Recall that Ring Learning with Errors (RLWE) is the variant of LWE specialized to polynomial rings over finite fields. Informally, Module-LWE can be seen as the result of

replacing single ring elements in the RLWE problem with module elements over the same ring (thus, RLWE can be seen as Module-LWE with module rank 1).

Definition 1 (Module-LWE assumption [7]). *The Module-LWE problem consists in distinguishing uniform samples* $(\mathbf{a}_i, b_i) \leftarrow R_q^k \times R_q$ *from samples* $(\mathbf{a}_i, \mathbf{a}_i^T\mathbf{s} + e_i) \in R_q^k \times R_q$ *where* $\mathbf{a}_i \leftarrow R_q^k$ *is uniform,* $\mathbf{s} \leftarrow \beta_\eta^k$ *common to all samples, and* $e_i \leftarrow \beta_n$ *is fresh for every sample. The advantage of an adversary* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{mlwe}_{m,k,\eta}(\mathcal{A}) = \left| \Pr\left[b' = 1 : \begin{array}{l} \mathbf{A} \leftarrow R_q^{m\times k}; \\ (\mathbf{s}, \mathbf{e}) \leftarrow \beta_\eta^k \times \beta_\eta^m; \\ \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}; \\ b' = \mathcal{A}(\mathbf{A}, \mathbf{b}) \end{array} \right] - \Pr\left[b' = 1 : \begin{array}{l} \mathbf{A} \leftarrow R_q^{m\times k}; \\ \mathbf{b} \leftarrow R_q^m; \\ b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b}) \end{array} \right] \right|.$$

The Module-LWE assumption states that the above advantage is negligible for any given adversary $\mathcal{A}$.

The authors of [7] defined a function $\mathtt{Compress}_q(x, d)$ that takes an element $x \in \mathbb{Z}_q$ and outputs an integer in $\{0, 1, \ldots, 2^d - 1\}$, where $d < \lceil \log_2(q) \rceil$. Furthermore, a function $\mathtt{Decompress}_q$ is defined such that

$$x' = \mathtt{Decompress}_q(\mathtt{Compress}_q(x, d), d)$$

is an element close to x. More specifically,

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

The functions satisfying these requirements are defined in [7] as:

$$\begin{array}{rcl} \mathtt{Compress}_q(x, d) & = & \lceil (2^d/q) \cdot x \rfloor \mod 2^d, \\ \mathtt{Decompress}_q(x, d) & = & \lceil (q/2^d) \cdot x \rfloor. \end{array}$$

Kyber's PKE scheme $\mathtt{Kyber.CPA}' = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ is parameterized by the positive integers k, d_u, and d_v. The value of these parameters vary for different security levels. Moreover, $\mathcal{M} = \{0,1\}^n$ is the message space and ciphertexts are of the form $(\mathbf{u}, v) \in \{0,1\}^{n\,k\,d_u} \times \{0,1\}^{n\,d_v}$. The definition of the key generation, encryption, and decryption of $\mathtt{Kyber.CPA}'$ is given in Algorithms 1–3 as defined in [7]. Unlike $\mathtt{Kyber.CPA}'$, the unmodified PKE scheme Kyber.CPA compresses $\mathbf{t}$ on Line 4 of Algorithm 1 and, therefore, must decompress $\mathbf{t}$ in Algorithm 2.

$\mathtt{Kyber.CPA}'$ was shown to be IND-CPA secure under the Module-LWE hardness assumption in [7]. This result is stated in the following theorem.

Theorem 1 ([7]). *For any adversary* $\mathcal{A}$ *against the CPA security of* $\mathtt{Kyber.CPA}'$*, let define the advantage*

$$\mathsf{Adv}^{cpa}_{\mathtt{Kyber.CPA}'}(\mathcal{A}) = \left| \Pr\left[b = b' : \begin{array}{l} (pk,\, sk) \leftarrow \mathtt{KeyGen}(); \\ (m_0,\, m_1,\, s) \leftarrow \mathcal{A}(pk); \\ b \leftarrow \{0,1\};\ c^* \leftarrow \mathtt{Enc}(pk,\, m_b); \\ b' \leftarrow \mathcal{A}(s,\, c^*) \end{array} \right] - \frac{1}{2} \right|.$$

Then, there exists an adversary $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{cpa}_{\mathtt{Kyber.CPA}'}(\mathcal{A}) \leq 2\,\mathsf{Adv}^{mlwe}_{k+1,k,\eta}(\mathcal{B}).$$

Finally, it is worth pointing out that neither `Kyber.CPA` nor $\texttt{Kyber.CPA}'$ provides perfect correctness. This is discussed in [7], where a value for δ, the probability of decryption error, is obtained for `Kyber.CPA`. This is easily adapted to $\texttt{Kyber.CPA}'$; the details can be found in Appendix A.

Algorithm 1: $\texttt{Kyber.CPA}'\texttt{.KeyGen()}$

$\rho, \sigma \leftarrow \{0,1\}^n$
$\mathbf{A} \sim R_q^{k \times k} := \texttt{Sam}(\rho)$
$(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \texttt{Sam}(\sigma)$
$\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$
return $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

Algorithm 2: $\texttt{Kyber.CPA}'\texttt{.Enc}(pk = (\mathbf{t}, \rho), m \in \mathcal{M}))$

$r \leftarrow \{0,1\}^n$
$\mathbf{A} \sim R_q^{k \times k} := \texttt{Sam}(\rho)$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$
$\mathbf{u} := \texttt{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
$v := \texttt{Compress}_q(\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
return $c := (\mathbf{u}, v)$

Algorithm 3: $\texttt{Kyber.CPA}'\texttt{.Dec}(sk = \mathbf{s}, c = (\mathbf{u}, v))$

$\mathbf{u} := \texttt{Decompress}_q(\mathbf{u}, d_u)$
$v := \texttt{Decompress}_q(v, d_v)$
return $\texttt{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

3.2. The FO_{AKE} Transformation: From PKE to AKE

FO_{AKE} is a generic construction proposed in [9], which transforms an IND-CPA secure PKE scheme into an AKE protocol, provably secure in the QROM. The construction admits that the PKE scheme has non-perfect correctness, which makes it suitable for the $\texttt{Kyber.CPA}'$ scheme we have previously introduced. Another nice feature is that it avoids the use of (usually expensive) quantum-secure signature schemes. FO_{AKE} can be seen as an extension of the Fujisaki–Okamoto transform (which turns IND-CPA encryption schemes into IND-CCA ones) for the AKE setting.

The resulting AKE after applying the FO_{AKE} transformation is quite efficient in terms of communication. In [9], it is called a two-message protocol, meaning that it is a two-round AKE protocol where one party sends a message in the first round while the other party answers with another message in the second round. As an interesting additional contribution, the authors of [9] defined a security model and two security notions for two-message AKEs: key indistinguishability agains active attacks (IND-AA) and the weaker notion of indistinguishability against active attacks without state reveal in the test session (IND-StAA). We are interested in the second one, as the security of the AKE obtained by using the FO_{AKE} transformation is proved in [9] under this slightly weaker model. Nevertheless, this is enough for our purposes because, as discussed in Section 5 of [15] (the extended version of [9]), IND-StAA implies security in the sense required in the compiler from [8].

A high level description of the IND-StAA model, as formulated in [9], is the following. It states that the session key remains indistinguishable from a random one even if:

1. The attacker knows either the long-term secret key or the secret state information (but not both) of both parties involved in the test session, as long as it did not modify the message received by the test session.
2. If the attacker modified the message received by the test session, as long as it obtained neither the long-term secret key of the test session's peer **nor the test session's state**.

The authors of the $\mathrm{FO}_{\mathrm{AKE}}$ transformation proved its IND-StAA security in the QROM as long as the PKE is IND-CPA, and it is possible to efficiently fake ciphertexts that are indistinguishable from proper encryptions, while the probability that the sampling algorithm hits a proper encryption is small. This last notion is called Disjoint Simulatability (DS) of ciphertexts, and is defined in [9] as follows.

Definition 2 (DS). *Let $\mathtt{PKE} = (\mathtt{KG}, \mathtt{Enc}, \mathtt{Dec})$ be a PKE scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, coming with an additional ppt algorithm $\overline{\mathtt{Enc}}$. For quantum adversaries $\mathcal{A}$, we define the advantage against $\mathtt{PKE}$'s disjoint simulatability as*

$$\mathsf{Adv}^{DS}_{PKE,\overline{Enc}}(\mathcal{A}) = \left| \Pr\left[\begin{array}{l} pk \leftarrow KG, \\ m \leftarrow \mathcal{M}, \\ c \leftarrow Enc(pk, m) \end{array} : 1 \leftarrow \mathcal{A}(pk, c) \right] - \Pr\left[\begin{array}{l} pk \leftarrow KG, \\ c \leftarrow \overline{Enc} \end{array} : 1 \leftarrow \mathcal{A}(pk, c) \right] \right|.$$

When there is no chance of confusion, we drop $\overline{\mathtt{Enc}}$ from the advantage's subscript for convenience. We call $\mathtt{PKE}$ ε_{dis}-disjoint if for all $pk \in supp(\mathtt{KG})$,

$$Pr[c \leftarrow \overline{Enc} : c \in Enc(pk, \mathcal{M}; \mathcal{R})] \leq \varepsilon_{dis},$$

where $\mathcal{R} = \mathcal{R}(pk)$ is a finite randomness space defined by pk.

The authors of the $\mathrm{FO}_{\mathrm{AKE}}$ transformation suggested that many lattice-based schemes fulfill DS in a natural way as follows: fake encryptions could be sampled uniformly random. DS would follow from the LWE assumption, and since LWE samples are relatively sparse, uniform sampling should be disjoint.

The following theorem establishes that the DS security of $\mathtt{Kyber.CPA}'$ equipped with an additional algorithm $\overline{\mathtt{Enc}}$ reduces to its Module-LWE security.

Theorem 2 (DS security of $\mathtt{Kyber.CPA}'$). *Let η, k, d_u, and d_v be positive integer parameters for $\mathtt{Kyber.CPA}'$. If $\mathtt{Kyber.CPA}'$ is equipped with a ppt algorithm $\overline{\mathtt{Enc}}$ which samples a uniform ciphertext when given a public key, then, for any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{DS}_{Kyber.CPA'}(\mathcal{A}) \leq 2\,\mathsf{Adv}^{mlwe}_{k+1,k,\eta}(\mathcal{B}).$$

Furthermore, $\mathtt{Kyber.CPA}'$ is ε_{dis}-disjoint with

$$\varepsilon_{dis} = \frac{1}{2^{n\,(d_u\,k + d_v - 2)}}.$$

Proof. Let $\mathcal{A}$ be an adversary attacking the DS security of $\mathtt{Kyber.CPA}'$. We obtain a bound for $\mathsf{Adv}^{\mathsf{DS}}_{\mathtt{Kyber.CPA}'}(\mathcal{A})$ following the sequence of games in the proof of Theorem 2 in [7].

First, the value $\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$ which is used in $\mathtt{KeyGen}$ is substituted by a uniform random value. It follows from the Module-LWE security of $\mathtt{Kyber.CPA}'$ that the value $\mathbf{t}$ and the uniform random value are indistinguishable from each other. Next, the values $\mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ and $\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m$ used in the generation of the challenge ciphertext are simultaneously substituted with uniform random values. Again, it follows

from the Module-LWE security of $\texttt{Kyber.CPA}'$ that $\mathbf{A}^T\mathbf{r}+\mathbf{e}_1$ and $\mathbf{t}^T\mathbf{r}+e_2+\lceil\frac{q}{2}\rfloor\cdot m$ are indistinguishable from the random values. As in [7], we deduce that there exists an adversary $\mathcal{B}$ with the same running time as that of $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{DS}}_{\texttt{Kyber.CPA}'}(\mathcal{A}) \leq 2\,\mathsf{Adv}^{\mathsf{mlwe}}_{k+1,k,\eta}(\mathcal{B})$.

To prove the $\varepsilon_{\mathrm{dis}}$-disjointness of $\texttt{Kyber.CPA}'$ with $\varepsilon_{\mathrm{dis}} = 2^{n(2-d_u k-d_v)}$, we recall that $\mathcal{M}=\{0,1\}^n$, $\mathcal{C}=\{0,1\}^{nkd_u}\times\{0,1\}^{nd_v}$, and $\mathcal{R}=\{0,1\}^n$ are the message, ciphertext and random spaces, respectively. Since $|\texttt{Enc}(pk,\,\mathcal{M};\,\mathcal{R})| \leq |\mathcal{M}|\,|\mathcal{R}| = 2^{2n}$, we obtain

$$\begin{aligned}\Pr[c \leftarrow \overline{\texttt{Enc}} \,:\, c \in \texttt{Enc}(pk,\,\mathcal{M};\,\mathcal{R})] &\leq \max_{(pk,sk)\in\texttt{KeyGen}(\mathcal{R})} \frac{|\texttt{Enc}(pk,\,\mathcal{M};\,\mathcal{R})|}{|\mathcal{C}|}\\ &\leq \frac{2^{2n}}{2^{n(d_u k+d_v)}}\\ &= \frac{1}{2^{n(d_u k+d_v-2)}},\end{aligned}$$

which is the desired result. □

Now that Theorems 1 and 2 guarantee that $\texttt{Kyber.CPA}'$ satisfies the hypotheses of Theorem 3 in [9], we can use it to produce a two-party AKE which fulfills IND-StAA security in the QROM. The resulting scheme, which we denote by $\texttt{Kyber.2AKE}$, is depicted in Figure 4. Here, G and H are random oracles and H'_R, H'_{L1}, H'_{L2}, and H'_{L3} are internal random oracles that cannot be accessed directly and could be implemented with a pseudorandom function. Note that this is not the same two-party AKE proposed in [7]. For reference, we include the precise statement of Theorem 3 [9] in Appendix B.

3.3. The Commitment Scheme

In this section, we describe how to obtain an IND-CCA PKE from an IND-CPA PKE. This process can be achieved in two steps:

1. Apply the $\mathrm{FO}^{\not\perp}_m$ transformation [9] that converts an IND-CPA PKE into a IND-CCA KEM.
2. Apply the transformation proposed in [10] to achieve an IND-CCA PKE from an IND-CCA KEM.

To achieve an IND-CCA secure KEM from $\texttt{Kyber.CPA}'$, we apply the $\mathrm{FO}^{\not\perp}_m$ transformation. This is analogous to the $\mathrm{FO}_{\mathrm{AKE}}$ transformation that transforms a PKE scheme that is both IND-CPA and DS secure into a CCA-secure KEM. As shown in [9], unlike similar transformations, $\mathrm{FO}^{\not\perp}_m$ is robust against correctness errors and its security reduction is tighter than the one that results from applying other known transformations. In cases where the PKE is not already DS, this requirement can be waived with negligible loss of efficiency. In the case of $\texttt{Kyber.CPA}'$, there is no loss of efficiency since it is IND-CPA secure and, as shown in Theorem 2, it is DS secure as well. The Algorithms 1, 4, and 5 show the KEM $\texttt{Kyber}^{\not\perp} = (\texttt{Kyber.CPA}'\texttt{.KeyGen}, \texttt{Encaps}, \texttt{Decaps})$ that results from applying the transformation $\mathrm{FO}^{\not\perp}_m$ to $\texttt{Kyber.CPA}'$. Here, G and H are random oracles and H_r is an internal random oracle that cannot be accessed directly and could be implemented with a pseudorandom function. For reference, we include the precise statement of Theorem 2 [9] in Appendix C.

$\mathbf{P_i}$		$\mathbf{P_j}$
Static auth. keys $(pk_i, sk_i) \leftarrow \texttt{Kyber.CPA}'\texttt{.KeyGen}()$		Static auth. keys $(pk_j, sk_j) \leftarrow \texttt{Kyber.CPA}'\texttt{.KeyGen}()$
$m_j \leftarrow \mathcal{M}$ $c_j := \texttt{Kyber.CPA}'\texttt{.Enc}(pk_j, m_j; \mathsf{G}(m_j))$ $(\tilde{sk}, \tilde{pk}) \leftarrow \texttt{Kyber.CPA}'\texttt{.KeyGen}()$ $M := (\tilde{pk}, c_j)$ $st := (\tilde{sk}, m_j, M)$		
	$\xrightarrow{M}$	

		$(\tilde{pk}, c_j) := M$
		$m_i, \tilde{m} \leftarrow \mathcal{M}$
		$c_i := \texttt{Kyber.CPA}'\texttt{.Enc}(pk_i, m_i; \mathsf{G}(m_i))$
		$\tilde{c} := \texttt{Kyber.CPA}'\texttt{.Enc}(\tilde{pk}, \tilde{m}; \mathsf{G}(\tilde{m}))$
		$M' := (c_i, \tilde{c})$
		$m'_j := \texttt{Kyber.CPA}'\texttt{.Dec}(sk_j, c_j)$
		if $m'_j = \perp$
		or $c_j \neq \texttt{Kyber.CPA}'\texttt{.Enc}(pk_j, m'_j; \mathsf{G}(m'_j))$
		  $\mathsf{k}' := \mathsf{H}'_R(m_i, c_j, \tilde{m}, i, j, M, M')$
		else
		  $\mathsf{k}' := \mathsf{H}(m_i, m'_j, \tilde{m}, i, j, M, M')$

	$\xleftarrow{M'}$	

$(c_i, \tilde{c}) := M'$
$m'_i := \texttt{Kyber.CPA}'\texttt{.Dec}(sk_i, c_i)$
$\tilde{m}'_i := \texttt{Kyber.CPA}'\texttt{.Dec}(\tilde{sk}, \tilde{c})$
if $m'_i = \perp$
or $c_i \neq \texttt{Kyber.CPA}'\texttt{.Enc}(pk_i, m'_i; \mathsf{G}(m'_i))$
  if $\tilde{m}' = \perp$
  $\mathsf{k} := \mathsf{H}'_{L1}(c_i, m_j, \tilde{c}, i, j, M, M')$
  else
  $\mathsf{k} := \mathsf{H}'_{L2}(c_i, m_j, \tilde{m}', i, j, M, M')$
else if $\tilde{m}' = \perp$
  $\mathsf{k} := \mathsf{H}'_{L3}(m'_i, m_j, \tilde{c}, i, j, M, M')$
else
  $\mathsf{k} := \mathsf{H}(m'_i, m_j, \tilde{m}', i, j, M, M')$		

Figure 4. `Kyber.2AKE`.

Algorithm 4: $\texttt{Kyber}^{\not\perp}.\texttt{Encaps}(pk)$

$m \xleftarrow{\$} \mathcal{M}$
$c := \texttt{Kyber.CPA}'.\texttt{Enc}(pk, m; \mathsf{G}(m))$
$\mathsf{k} := \mathsf{H}(\mathrm{m})$
return (k, c)

Algorithm 5: $\texttt{Kyber}^{\not\perp}.\texttt{Decaps}(sk, c)$

$m' := \texttt{Kyber.CPA}'.\texttt{Dec}(sk, c)$
if $m' = \perp$ ***or*** $\texttt{Kyber.CPA}'.\texttt{Enc}(pk, m'; \mathsf{G}(m')) \neq c$ **then**
  return $\mathsf{k} := \mathsf{H}_r(c)$
else
  return $\mathsf{k} := \mathsf{H}(m')$

Finally, an IND-CCA PKE is obtained after applying the transformation introduced in [10] to $\texttt{Kyber}^{\not\perp}$ with a secure one-time symmetric key encapsulation (SKE or DEM). We call this scheme Kyber.PKE. The security of this transformation follows from Theorem 5 in [10]. As pointed out in [8], a commitment scheme with the required security properties can be obtained in a straightforward way from the IND-CCA PKE.

4. Our Post-Quantum Group Key Exchange

In this section, we present our compiled construction of GAKE. Informally, let us recall the setting we are considering. Our participants are honest entities which can be modeled as probabilistic polynomial time Turing machines (thus, have no access to quantum computing resources). These participants can only exchange messages through an insecure network, which is fully under adversarial control (adversaries may insert, delay, suppress or forward messages at will). Moreover, the adversarial computing capabilities are superior to those of participants, as we assume adversaries can preform quantum polynomial time computations and have quantum access to any hash function (modeled as a random oracle) involved. With this in mind, the goal pursued by our protocol is to guarantee that, whenever a participant has computed a session key through the network, this key is indistinguishable from a random value for those outside the intended group of participants involved in that concrete execution. Note that (as is standard in GKE proposals) we cannot expect to prove that the protocol will always terminate when executed by honest parties, we rather pursue formal assurance that, whenever the protocol indeed produces an output key for a participant, this key is secure for subsequent use.

Now, to make the text fully self-contained, we start by describing the main notations and formalism used in the sequel.

4.1. Security Model

Our security model is inherited from Abdalla et al. [8], which in turn builds upon the seminal work of Bellare et al. [16]. However, ours is a less generic scenario; while in [8] all the proofs are in the common reference string model, our proofs are in the (quantum) random oracle model. More precisely, we assume that all public keys and parameters needed for implementing Kyber.2AKE and Kyber.PKE are publicly known (and certified), as well as the description of all involved hash functions, which are idealized as random oracles. Further, we will assume that the long term keys needed for authentication in Kyber.2AKE are generated and distributed to all potential protocol participants in a trusted initialization

phase. As customary, we use variables to detail the information stored by users with respect to each protocol execution, and oracles to model adversarial action.

4.1.1. Protocol Instances

Each protocol participant $U_i \in \mathcal{U}$ ($i \in \mathbb{N}$) may execute a polynomial number of protocol *instances* in parallel. A single instance $\Pi_i^{s_i}$ can be interpreted as a process executed by protocol participant U_i. Throughout, the notation $\Pi_i^{s_i}$ is used to refer to instance s_i of protocol participant $U_i \in \mathcal{U}$. To each instance, we assign seven variables:

$\mathsf{used}_i^{s_i}$ indicates whether this instance is or has been used for a protocol run. The $\mathsf{used}_i^{s_i}$ flag can only be set through a protocol message received by the instance due to a call to the Execute- or to the Send-oracle (see below).

$\mathsf{state}_i^{s_i}$ keeps the state information needed during the protocol execution as well as the long term keys needed for authentication.

$\mathsf{term}_i^{s_i}$ shows if the execution has terminated.

$\mathsf{sid}_i^{s_i}$ denotes a public session identifier that can serve as identifier for the session key $\mathsf{sk}_i^{s_i}$. Note that, even though we do not construct session identifiers as session transcripts, the adversary is allowed to learn all session identifiers.

$\mathsf{pid}_i^{s_i}$ stores the set of identities of those users that $\Pi_i^{s_i}$ aims at establishing a key with—including U_i himself.

$\mathsf{acc}_i^{s_i}$ indicates if the protocol instance was successful, i. e., the user accepted the session key.

$\mathsf{sk}_i^{s_i}$ stores the session key once it is accepted by $\Pi_i^{s_i}$. Before acceptance, it stores a distinguished NULL value.

We do not make explicit the initialization and evolution of all variables mentioned above, yet omissions are straightforward to understand from the context.

4.1.2. Communication Network

We assume arbitrary point-to-point connections among users to be available. The network is non-private and fully asynchronous: The adversary may delay, eavesdrop, insert, and delete messages at will.

4.1.3. Adversarial Capabilities

Following Hövelmanns et al. [15], we consider adversaries that can preform (quantum) polynomial time computations, and have classical access to all (online) oracles listed below. Furthermore, as explained in Section 2.2, our adversaries are given quantum access to any (offline) random oracles involved.

The capabilities of an adversary $\mathcal{A}$ are made explicit through access to *oracles* allowing $\mathcal{A}$ to communicate with protocol instances run by the users:

$\mathsf{Send}(U_i, s_i, M)$ This sends message M to the instance $\Pi_i^{s_i}$ and returns the reply generated by this instance. If $\mathcal{A}$ queries this oracle with an unused instance $\Pi_i^{s_i}$ and $M \subseteq \mathcal{P}$ a set of identities of principals, the $\mathsf{used}_i^{s_i}$-flag is set, $\mathsf{pid}_i^{s_i}$ initialized with $\mathsf{pid}_i^{s_i} := \{U_i\} \cup M$, and the initial protocol message of $\Pi_i^{s_i}$ is returned.

$\mathsf{Execute}(\{\Pi_{u_1}^{s_{u_1}}, \ldots, \Pi_{u_\mu}^{s_{u_\mu}}\})$ This executes a complete protocol run among the specified unused instances of the respective users. The adversary obtains a transcript of all messages sent over the network. A query to the Executeoracle is supposed to reflect a passive eavesdropping.

$\mathsf{Reveal}(U_i, s_i)$ This yields the value stored in $\mathsf{sk}_i^{s_i}$.

$\mathsf{Test}(U_i, s_i)$ Let b be a bit chosen uniformly at random. Provided that the session key is defined (i. e., $\mathsf{acc}_i^{s_i} = \mathsf{true}$ and $\mathsf{sk}_i^{s_i} \neq \mathrm{NULL}$) and instance $\Pi_i^{s_i}$ is fresh (see the definition of freshness below), $\mathcal{A}$ can execute this oracle query at any time when being activated. Then, the session key $\mathsf{sk}_i^{s_i}$ is returned if $b = 0$ and a uniformly chosen random session key is returned if $b = 1$. In this model, an arbitrary number of Testqueries is allowed for the adversary $\mathcal{A}$, but, once the Test oracle has returned a value for an instance $\Pi_i^{s_i}$, it will return the same value for all instances partnered with $\Pi_i^{s_i}$ (see the definition of partnering below).
$\mathsf{Corrupt}(U_i)$ This returns all long-term secrets of user U_i—in our case, the private keys used for authentication in `Kyber.2AKE`.

4.1.4. Correctness, Integrity and Secrecy

To define our correctness and security goals, we introduce *partnering* to express which instances are associated in a common protocol session.

Partnering. We refer to instances $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ as being *partnered* if $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$, $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.

An instance $\Pi_i^{s_i}$ is assumed to accept the session key constructed at the end of the corresponding protocol run if no deviation from the protocol specification has occurred. Moreover, without adversarial interference, all users involved in a certain session should come up with the same session key.

Definition 3. *We call a group key establishment protocol* P correct, *if in the presence of a passive adversary $\mathcal{A}$—i. e., $\mathcal{A}$ must neither use the* Send *nor the* Corrupt*oracle—the following holds: for all i, j with both $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$, we have $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j} \neq \mathrm{NULL}$ and $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$.*

Some sort of correctness should also be guaranteed even if adversaries actively participate in a concrete executions: the notion of *integrity*, introduced in [17], captures this idea.

Definition 4. *We say that a correct group key establishment protocol fulfills integrity if, with overwhelming probability, all instances of honest principals that have accepted with the same session identifier $\mathtt{sid}_j^{s_j}$ hold identical session keys $\mathtt{sk}_j^{s_j}$ and associated this key with the same principals $\mathtt{pid}_j^{s_j}$.*

Next, for detailing the security definition, we have to specify under which conditions a Test-query may be executed.

Definition 5. *A* Test*-query should only be allowed to those instances holding a key that is not for trivial reasons known to the adversary. To this aim, an instance $\Pi_i^{s_i}$ is called* fresh *if none of the following holds:*

- *For some $U_j \in \mathsf{pid}_i^{s_i}$, a query $\mathsf{Corrupt}(U_j)$ was executed before a query of the form $\mathsf{Send}(U_k, s_k, M)$ has taken place, for some message (or set of identities) M and some $U_k \in \mathsf{pid}_i^{s_i}$.*
- *The adversary earlier queried $\mathsf{Reveal}(U_j, s_j)$ with $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ being partnered.*

The idea of this definition is that revealing a session key from an instance $\Pi_i^{s_i}$ trivially yields the session key of all instances partnered with $\Pi_i^{s_i}$, and hence this kind of "attack" will be excluded in the security definition.

For a secure group key establishment protocol, we have to impose a corresponding bound on the adversary's *advantage*: The advantage $\mathsf{Adv}_{\mathcal{A}}(\ell)$ of a ppt adversary $\mathcal{A}$ in attacking protocol P is a function in the security parameter ℓ, defined as

$$\mathsf{Adv}_{\mathcal{A}} := |2 \cdot \mathsf{Succ} - 1|.$$

Here, Succ is the probability that the adversary queries Test only on fresh instances and guesses correctly the bit b used by the Test oracle (without violating the freshness of those instances queried with Test) :

Definition 6. *We say that an authenticated group key establishment protocol* P *is* secure *if for every ppt adversary* $\mathcal{A}$ *the following inequality holds for some negligible function* negl*:*

$$\mathsf{Adv}_{\mathcal{A}}(\ell) \leq \mathrm{negl}(\ell), \tag{1}$$

4.2. Our Construction

We aim at a full description of a GAKE protocol that can be proven secure against quantum adversaries, building on a post-quantum 2AKE and using the compiler described in Section 2.1. Our proposal is depicted in Figure 5. Note that in our compiled design we take as starting point a slightly modified version of the compiler from [8], in two ways:

- We simplify the session key and session identifier computation using two hash functions to extract them from the shared master key K. Indeed, as the 2AKE we use as building block is proven secure in the (quantum) random oracle model, it no longer makes sense to use the (somewhat complicated) key extraction procedure defined in [8] to dodge idealized hash functions. Thus, we forgo Tools 1 and 2 mentioned in Section 2.1 and use two hash functions $\hat{\mathtt{H}}$ and $\hat{\mathtt{F}}$ instead. Thus, at the final **Computation** phase, each user U_i will set the session key as $\mathtt{sk}_i = \hat{\mathtt{H}}(K)$ and the corresponding session identifier as $\mathtt{sid}_i = \hat{\mathtt{F}}(K)$, where K is the master key shared by everyone involved in the execution.
- Further, we make an additional requirement on the compiled 2AKE, needed for the security proof. Indeed, as pointed out by Nam in [18], an extra condition on the two party protocol used as a base must be imposed in Theorem 1 of [8]. Indeed, the underlying 2AKE should fulfill *integrity* in order to thwart a simple replay attack (in the proof of Theorem 1 of [8], it is actually assumed that integrity is fulfilled—see the argument related to Game 1). We thus slightly tune up the two-party 2AKE to make sure integrity is achieved.

4.3. Security Arguments and Proofs

To prove that our compiled version is secure, we build upon the security of our underlying tools. More precisely, we use the following results:

(i) Kyber.2AKE, as depicted in Figure 4, is secure in the sense of IND-StAA and can be modified to also attain integrity as in Definition 4. Note that, as explained in Section 5 of [15], IND-StAA implies security in the sense required in the original compiler from [8].
(ii) The encryption scheme Kyber.PKE yields a non-interactive commitment scheme that is both non-malleable for multiple commitments and perfectly binding. This comes straightforward as a result of this scheme being IND-CCA (see Section 3.3 and [15]).

4.3.1. A Variant of Kyber.2AKE Attaining Integrity

Informally, it is easy to modify in a standard way the construction Kyber.2AKE to attain integrity. The main idea is to add a second random oracle F which, at the point of key derivation, will be applied to the same input as H in order to derive a session identifier. Then, it is trivial to state that integrity of this modified Kyber.2AKE construction is attained both in the ROM and in the QROM, due to the collision resistance of the involved random oracles (see Section 2.2). Indeed, suppose that $\mathtt{k} = \mathtt{k}'$. Since H and F are random oracles, their collision resistance guarantees that, with overwhelming probability, both participants have the same partner identifiers and, therefore, use the same session key k. This argument is valid both in

the classical and quantum-accessible random oracle model (see Section 2.2). In the sequel, we assume this modification is in place and thus `Kyber.2AKE` attains integrity.

Round $1 \sim 2$**:** For each $i = 0, \ldots, n-1$, execute `Kyber.2AKE` with U_i and $U_{[i+1]}$, where $[k] = k \bmod n$ (i.e., all indices are to be taken in a cycle).
At completion, each $U_i \in \mathcal{G}$ holds two pairwise keys $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ shared with $U_{[i+1]}$ and $U_{[i-1]}$, respectively. Note that, for $i = 0, \ldots, n-1$, $\overrightarrow{K}_i = \overleftarrow{K}_{[i+1]}$.

Round 3:

- **Computation**: Each U_i computes $X_i = \overrightarrow{K}_i \oplus \overleftarrow{K}_i$, and chooses a random r_i to derive a commitment $C_i = \texttt{Kyber.PKE}(i, X_i; r_i)$,
- **Broadcast**: Each U_i broadcasts $M_i^1 = (U_i, C_i)$.

Round 4:

- **Broadcast**: Each U_i broadcasts $M_i^2 = (U_i, X_i, r_i)$.
- **Verification**: Each U_i checks that $X_0 \oplus X_1 \oplus \cdots \oplus X_{n-1} = 0$ and the correctness of the commitments.
  If any one of these checks fails, U_i terminates the protocol execution setting $\texttt{acc}_i := \texttt{FALSE}$, otherwise it sets $\texttt{acc}_i := \texttt{TRUE}$ and $\texttt{pid}_i := \mathcal{G}$.
- **Computation**: Each U_i computes the $n-1$ values

$$\overleftarrow{K}_{[i-j]} = \overleftarrow{K}_{[i-j+1]} \oplus X_{[i-j]}, \quad j = 1, 2, \ldots, n-1.$$

  Then, U_i defines a master key

$$K = \left(\overleftarrow{K}_0, \overleftarrow{K}_1, \ldots, \overleftarrow{K}_{n-1}, \mathcal{G} \right),$$

  and sets the session key and session identifier as

$$\texttt{sk}_i = \hat{\mathrm{H}}(K)$$

  and

$$\texttt{sid}_i := \hat{\mathrm{F}}(K)$$

  where $\hat{\mathrm{H}} : \{0,1\}^* \to \{0,1\}^\ell$ and $\hat{\mathrm{F}} : \{0,1\}^* \to \{0,1\}^\ell$ are hash functions and $\ell \in \mathbb{N}$ is the security parameter.

Figure 5. Proposed Post-Quantum group-key establishment.

4.3.2. Security of Our Proposed Group Protocol

Theorem 3. *In the random oracle model, the protocol presented in Figure 5 is a correct and secure authenticated group key establishment protocol fulfilling integrity, in the sense of Definitions 3, 6, and 4.*

Proof. This proof is a (somewhat) straightforward adaptation of the security proof of Theorem 1 of [8], which we use as a main tool in our construction.

Correctness. In an honest execution of the protocol, it is easy to verify that all participants in the protocol will terminate by accepting and computing the same session identifier and session key.

Integrity. Owing to the collision-resistance of the random oracle $\hat{\mathrm{F}}$ all oracles that accept with identical session identifiers also hold with overwhelming probability the same master key K and pid (which can be read from K) will therefore also derive the same session key $\hat{\mathrm{H}}(K)$.

Key secrecy. The proof of key secrecy will proceed in a sequence of games, starting with the real attack against the key secrecy of the group key exchange protocol and ending in a game in which the adversary's advantage is 0, and for which we can bound the difference in the adversary's advantage between any two consecutive games. Following standard notation, we denote by $\mathsf{Adv}(\mathcal{A}, G_i)$ the advantage of the adversary $\mathcal{A}$ in Game i. Furthermore, for clarity, we classify the Send queries into three categories, depending on the stage of the protocol to which the query is associated, starting with Send-0 and ending with Send-2. Send-t denotes the Send query associated with round t for $t = 0, 1, 2$.

The first three games from this proof are exactly the same as those in the proof of Theorem 1 of [8]. We only summarize the reduction and refer the interested reader to the original paper for a detailed description.

Game 0. This first game corresponds to a real attack, in which all the parameters, such as the public parameters in the common reference string and the long-term secrets associated with each user, are chosen as in the actual scheme. By definition, $\mathsf{Adv}(\mathcal{A}, G_0) = \mathsf{Adv}(\mathcal{A})$.

Game 1. In this game, for $i = 1, \ldots, n$, we modify the simulation of the Send and Execute oracles so that, whenever an instance $\Pi_i^{s_i}$ is still considered fresh at the end of Round 2, the keys $\overleftarrow{K}_i$ and $\overrightarrow{K}_i$ that it shares with instances $\Pi_{i-1}^{s_{i-1}}$ and $\Pi_{i+1}^{s_{i+1}}$ are replaced with random values from the appropriate set.

It is easy to see that the distance between this game and the previous one is bounded by the probability that the adversary breaks the security of any of the underlying 2-AKE protocols. As a result, it holds

$$\left|\mathsf{Adv}(\mathcal{A}, G_1) - \mathsf{Adv}(\mathcal{A}, G_0)\right| \leq 2 \cdot \mathsf{Adv}_{\mathsf{2\text{-}AKE}}(\ell, 2 \cdot q_{\mathrm{send}}),$$

where q_{send} represents the number of *different* protocol instances in Send queries.

Game 2. In this game, we change the simulation of the Send oracle so that a *fresh* instance $\Pi_i^{s_i}$ does not accept in Round 4 whenever one commitment C_j for $j \neq i$ it receives in Round 3 was generated by the simulator but not generated by the respective instance $\Pi_j^{s_j}$, $j \neq i$ in the same session.

The adversary $\mathcal{A}$ can detect the difference to Game G_1 if $\mathcal{A}$ replayed a commitment that should have led to acceptance in Round 4 in that game. Because the committed value X_i is a random value independent of previous messages, the probability for this is negligible.

$$\left|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)\right| \leq \mathrm{negl}(\ell)$$

Game 3. This game reproduces the modification also for *adversary-generated* commitments: The simulation of the Send oracle changes so that a *fresh* instance $\Pi_i^{s_i}$ does not accept in Round 4 whenever one commitment C_j for $j \neq i$ it receives in Round 3 was *adversary-generated*. The adversary's advantage diverges only negligibly from the previous game:

$$\left|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_2)\right| \leq \mathrm{negl}(\ell)$$

Game 4. Now, the simulations of the Execute and Send oracles are modified at the point of computing the session key. The simulator keeps a list of strings $(K_1, \ldots, K_n, \mathcal{G})$. Once an instance receives the last Send-2 query, the simulator computes $K_1, \ldots, K_n$ and checks if for the corresponding string $(K_1, \ldots, K_n, \mathcal{G})$ a master key was already issued. If this is the case, it assigns the corresponding master key to the instance. If no such entry exists in the list, the simulator chooses a session key $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ uniformly at random. Note that, even if the messages from Round 4 are sent out, the master key is still containing sufficient entropy so that the random oracle output $\hat{\mathrm{H}}$ is indistinguishable from a random $\mathsf{sk}_i^{s_i}$ with negligible probability only. As a result,

$$\left|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)\right| \leq \mathrm{negl}(\ell).$$

Now, clearly, in Game G_4, all session keys are chosen uniformly at random and the adversary has no advantage.

$$\mathsf{Adv}(\mathcal{A}, G_4) = 0.$$

□

Theorem 4. *In the quantum random oracle model, the protocol presented in Figure 5 is a correct and secure authenticated group key establishment protocol fulfilling integrity, in the sense of Definitions 3, 6, and 4.*

Proof. (sketch) The proof follows the exact reasoning of Theorem 3; we only need to stress that the argument from Game 4 is still valid when considering quantum-accessible random oracles. Indeed, in this last game, the simulations of the Execute and Send oracles are modified at the point of computing the session key. The simulator keeps a list of strings $(K_1, \ldots, K_n, \mathcal{G})$, and, upon receiving the last Send-2 query, it computes the values $K_1, \ldots, K_n$ and checks if a corresponding master key has already been issued previously. If this is the case, this master key will be assigned to the instance. Otherwise, the simulator chooses a session key $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ uniformly at random. At this point, all two party keys $K_1, \ldots, K_n$ are chosen uniformly at random and are unknown to the adversary. The adversary can only notice this last change if it has already queried the very same key string to the quantum random oracle $\hat{\mathrm{H}}$. This event will happen with negligible probability. As a result, the output $\hat{\mathrm{H}}$ is indistinguishable from a random $\mathsf{sk}_i^{s_i}$ with overwhelming probability. Thus, we have

$$\left|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)\right| \leq \mathrm{negl}(\ell).$$

Now, clearly, in Game G_4, all session keys are chosen uniformly at random and the adversary has no advantage.

$$\mathsf{Adv}(\mathcal{A}, G_4) = 0.$$

□

5. Conclusions

We present in this paper a post-quantum GAKE using Abdalla et al.'s compiler from [8] as design frame. We choose the `Kyber` suite [7] as main building block, not only because it is a good design fit for our compiled strategy, but also considering its promising security properties (as `Kyber` is one of the four remaining finalists for public key encryption in the Third Round of the NIST competition). More precisely, we evidence that a secure `2AKE` as needed for our compiled construction can be derived using the $\mathrm{FO}_{\mathrm{AKE}}$ transformation proposed in [9], by proving the encryption scheme `Kyber.CPA`$'$ to be DS secure.

Our four-round instantiation can as a result be proven to provide post-quantum security guarantees under the Module-LWE assumption in the quantum random oracle model.

Author Contributions: All authors contributed equally to this work, in terms of Conceptualization, Methodology, Formal Analysis, Investigation, Writing—Original Draft Preparation and Review and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NATO Science for Peace and Security Programme, grant number G5448 and by MINECO under Grants MTM2016-77213-R and PID2019-109379RB-I00.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Non-Perfect Correctness of Kyber.CPA′

As defined in [19], a PKE is said to be $(1-\delta)$-correct if

$$\mathbf{E}\left[\max_{m\in\mathcal{M}} \Pr\left[\mathtt{Dec}(sk, \mathtt{Enc}(pk, m)) = m\right]\right] > 1-\delta,$$

where the expectation is taken over $(pk, sk) \leftarrow \mathtt{KeyGen}()$ and the probability is taken over the random space of Enc.

Following the proof of Theorem 1 in [7], it is not hard to prove the following theorem, which provides the value δ when dealing with Kyber.CPA′.

Theorem A1. *Let k be a positive integer parameter. Let $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1, e_2$ be random variables that have the same distribution as in Algorithms 1 and 2. In addition, let $\mathbf{c}_u \leftarrow \psi_{d_u}^k$, $\mathbf{c}_v \leftarrow \psi_{d_v}^k$ be distributed according to the distribution ψ defined as follows:*

Let ψ_d^k be the following distribution over R:

1. *Choose uniformly-random $\mathbf{y} \leftarrow R^k$*
2. ***return*** $(y - \mathtt{Descompress}_q(\mathtt{Compress}_q(\mathbf{y}, d), d)) \bmod^{\pm} q$

Denote

$$\delta = \Pr\left[\|\mathbf{e}^T\mathbf{r} + e_2 + \mathbf{c}_v - \mathbf{s}^T\mathbf{e}_1 - \mathbf{s}^T\mathbf{c}_u\|_\infty \geq \lceil q/4 \rfloor\right],$$

where, for $w = w_0 + w_1 X + \cdots + w_{n-1} X^{n-1} \in R$:

$$||w||_\infty = \max_i |w_i \bmod^{\pm} q|,$$

and, similarly, for $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$:

$$||\mathbf{w}||_\infty = \max_i ||w_i||_\infty.$$

Then, the modified scheme Kyber.CPA′ is $(1-\delta)$-correct.

Appendix B. Transformation from IND-CPA PKE to Secure 2AKE

We reproduce here the result given in [9] about the IND-StAA security of the $\mathrm{FO}_{\mathrm{AKE}}$ transformation. The following theorem states that the IND-StAA security of $\mathtt{AKE} = \mathrm{FO}_{\mathrm{AKE}}(\mathtt{PKE}, G, H)$, where PKE is a PKE scheme and G, H are random oracles, reduces to the DS and IND-CPA security of PKE. Note that some references to oracles appear in the statement; for details about these oracles and the formal definition of IND-StAA security, see [9].

Theorem A2 ([9])**.** *Assume PKE=(KG,Enc,Dec) to be $(1-\delta)$-correct, and to come with a sampling algorithm $\overline{\mathtt{Enc}}$ such that it is ε-disjoint. Let N be the number of parties, and suppose that any attacker is granted access to an oracle REVEAL which reveals the respective session's key (if already defined). Then, for any IND-StAA adversary $\mathcal{B}$ that*

establishes S sessions and issues at most q_R (classical) queries to REVEAL, at most q_G (quantum) queries to random oracle G, and at most q_H (quantum) queries to random oracle H, there exist adversaries $\mathcal{A}_{DS}$ and $\mathcal{A}_{CPA}$ against PKE such that

$$\begin{aligned}
&\mathsf{Adv}^{IND\text{-}StAA}_{AKE}(\mathcal{B}) \leq 2\,S\,(S+3\,N)\,\mathsf{Adv}^{DS}_{PKE}(\mathcal{A}_{DS}) \\
&+4\,S\,(S+3\,N)\sqrt{(q_G+2\,q_H+3\,S)\,\mathsf{Adv}^{cpa}_{PKE}(\mathcal{A}_{CPA})+\frac{4\,(q_G+2\,q_H+3\,S)^2}{|\mathcal{M}|}} \\
&+32\,(S+3\,N)\,(q_G+2\,q_H+3\,S)^2\,(1-\delta)+4\,S\,(S+N)\,\varepsilon_{dis} \\
&+S^2\,(N+1)\,\mu(KG)\,\mu(Enc)+2\,S^2+\mu(KG),
\end{aligned}$$

and the running times of $\mathcal{A}_{DS}$ and $\mathcal{A}_{CPA}$ is about that of $\mathcal{B}$. Here,

$$\mu(KG)=Pr[(pk,sk)\leftarrow KG,\ (pk',sk')\leftarrow KG:\ pk=pk']$$

and

$$\mu(Enc)=Pr[(pk,sk)\leftarrow KG,\ m,m'\leftarrow\mathcal{M},\ c\leftarrow Enc(pk,m),\ c'\leftarrow Enc(pk,m'):\ c=c'].$$

Appendix C. Transformation from IND-CPA PKE to IND-CCA KEM

We reproduce here the result given in [9] about the IND-CCA security of the $\mathrm{FO}^{\not\perp}_m$ transformation. The following theorem states that the IND-CCA security of $\mathrm{FO}^{\not\perp}_m=\mathrm{FO}^{\not\perp}(\mathtt{PKE},G,H)$, where PKE is a PKE scheme and G,H are random oracles, reduces to the DS and IND-CPA security of PKE. Note that some references to oracles appear in the statement; for details about these oracles (see [9]).

Theorem A3 ([9]). *Assume PKE=(KG,Enc,Dec) to be $(1-\delta)$-correct, and to come with a sampling algorithm $\overline{Enc}$ such that it is ε_{dis}-disjoint. Suppose that any attacker is granted access to an oracle DECAPS. Then, for any (quantum) IND-CCA adversary $\mathcal{A}$ issuing at most q_D (classical) queries to decapsulation oracle DECAPS, at most q_G quantum queries to random oracle G, and at most q_H quantum queries to random oracle H, there exist (quantum) adversaries $\mathcal{B}_{DS}$ and $\mathcal{A}_{CCA}$ against PKE such that*

$$\begin{aligned}
&\mathsf{Adv}^{IND\text{-}CCA}_{KEM}(\mathcal{A})\leq 8\cdot(2q_G+q_H+q_D+4)^2\cdot\delta+\mathsf{Adv}^{DS}_{PKE}(\mathcal{B}_{DS}) \\
&+2\sqrt{(q_G+q_H)\cdot\mathsf{Adv}^{IND\text{-}CPA}_{PKE}(\mathcal{B}_{IND\text{-}CCA})+\frac{4(q_G+q_H)^2}{|\mathcal{M}|}}+\varepsilon_{dis},
\end{aligned}$$

and the running times of $\mathcal{B}_{DS}$ and $\mathcal{B}_{IND\text{-}CPA}$ is about that of $\mathcal{A}$.

References

1. Fujioka, A.; Takashima, K.; Yoneyama, K. One-Round Authenticated Group Key Exchange from Isogenies. ProvSec. *Lect. Notes Comput. Sci.* **2019**, *11821*, 330–338.
2. Apon, D.; Dachman-Soled, D.; Gong, H.; Katz, J. Constant-Round Group Key Exchange from the Ring-LWE Assumption. PQCrypto. *Lect. Notes Comput. Sci.* **2019**, *11505*, 189–205.
3. Katz, J.; Yung, M. Scalable Protocols for Authenticated Group Key Exchange. In *Advances in Cryptology—CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2003, Proceedings*; Boneh, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2729, pp. 110–125. [CrossRef]
4. Choi, R.; Hong, D.; Kim, K. Constant-round Dynamic Group Key Exchange from RLWE Assumption. *IACR Cryptol. ePrint Arch.* **2020**, *2020*, 35.

5. Persichetti, E.; Steinwandt, R.; Corona, A.S. From Key Encapsulation to Authenticated Group Key Establishment—A Compiler for Post-Quantum Primitives †. *Entropy* **2019**, *21*, 1183. [CrossRef]
6. González Vasco, M.; Pérez del Pozo, A.; Steinwandt, R. Group Key Establishment in a Quantum-Future Scenario. *Informatica* **2020**, 1–18. [CrossRef]
7. Bos, J.W.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS—Kyber: A CCA-Secure Module-Lattice-Based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 353–367.
8. Abdalla, M.; Bohli, J.; Vasco, M.I.G.; Steinwandt, R. (Password) Authenticated Key Establishment: From 2-Party to Group. TCC. *Lect. Notes Comput. Sci.* **2007**, *4392*, 499–514.
9. Hövelmanns, K.; Kiltz, E.; Schäge, S.; Unruh, D. Generic Authenticated Key Exchange in the Quantum Random Oracle Model. In *Public-Key Cryptography—PKC 2020*; Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 389–422.
10. Cramer, R.; Shoup, V. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM J. Comput.* **2003**, *33*, 167–226. [CrossRef]
11. David Jao, E.A. Supersingular Isogeny Key Encapsulation. Submission to NIST Post-Quantum Project. 2017. Available online: https://sike.org/#nist-submission (accessed on 16 October 2020).
12. Chailloux, A.; Naya-Plasencia, M.; Schrottenloher, A. An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography. In *Advances in Cryptology—ASIACRYPT 2017—23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017*; Takagi, T., Peyrin, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10625, pp. 211–240. [CrossRef]
13. Zhandry, M. Secure Identity-Based Encryption in the Quantum Random Oracle Model. In *Advances in Cryptology—CRYPTO 2012—32nd Annual Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2012*; Safavi-Naini, R., Canetti, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012, Volume 7417, pp. 758–775. [CrossRef]
14. Boneh, D.; Dagdelen, Ö.; Fischlin, M.; Lehmann, A.; Schaffner, C.; Zhandry, M. Random Oracles in a Quantum World. In *Advances in Cryptology—ASIACRYPT 2011—17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, Korea, 4–8 December 2011*; Lee, D.H., Wang, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2011, Volume 7073, pp. 41–69. [CrossRef]
15. Hövelmanns, K.; Kiltz, E.; Schäge, S.; Unruh, D. Generic Authenticated Key Exchange in the Quantum Random Oracle Model. *IACR Cryptol. ePrint Arch.* **2018**, *2018*, 928.
16. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology—EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, 14–18 May 2000*; Preneel, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1807, pp. 139–155. [CrossRef]
17. Bohli, J.; Vasco, M.I.G.; Steinwandt, R. Secure group key establishment revisited. *Int. J. Inf. Sec.* **2007**, *6*, 243–254. [CrossRef]
18. Nam, J.; Paik, J.; Won, D. A security weakness in Abdalla et al.'s generic construction of a group key exchange protocol. *Inf. Sci.* **2011**, *181*, 234–238. [CrossRef]
19. Hofheinz, D.; Hövelmanns, K.; Kiltz, E. A Modular Analysis of the Fujisaki-Okamoto Transformation. TCC (1). *Lect. Notes Comput. Sci.* **2017**, *10677*, 341–371.