INFORMATICA, 2025, Vol. 36, No. 2, 315–335
© 2025 *Vilnius University*
DOI: https://doi.org/10.15388/25-INFOR595

Scalable Authenticated Group Key Establishment in Quantum and Post-Quantum Networks

Maria Isabel GONZÁLEZ VASCO[1,*], Rainer STEINWANDT[2]

[1] *Departamento de Matemáticas, Universidad Carlos III de Madrid, Spain*
[2] *University of Alabama in Huntsville, USA*
e-mail: mariaisabel.gonzalez@uc3m.es, rs0141@uah.edu

Received: November 2024; accepted: May 2025

Abstract. Establishing secure keys over untrusted networks is one of the most fundamental cryptographic tasks. While two-party key establishment protocols are available for many scenarios, even offering resistance to potential adversaries equipped with quantum computing resources, the multi-party scenario is not as well understood. In particular, there is a need to find designs that can make the most of the technologies available to each party involved in a cooperative n-party key establishment.

We propose an authenticated key establishment protocol involving $n \geqslant 2$ parties, assuming that some—possibly all—network nodes have the potential to implement quantum key distribution (in pairs), while others only have access to standard technology. The protocol allows for the cooperative construction of a shared secret key from partial keys established by quantum and post-quantum solutions, which in turn can be implemented by different building blocks. We give a formal security analysis of our proposal using a hybrid security model simultaneously capturing quantum and classical actions and capabilities.

Key words: group key establishment, post quantum cryptography, quantum key distribution, security model.

1. Introduction

To cope with the potential of cryptanalytically-relevant quantum computers, two lines of research are actively pursued: *quantum cryptography* aims at leveraging results from quantum physics to establish security guarantees, and *post-quantum cryptography* makes use of purely classical constructions built from hardness assumptions that take into account quantum cryptanalysis.[1] In this paper, we look at one of the fundamental cryptographic tasks—establishing a common secret key among a set of $n \geqslant 2$ participants in an authenticated manner. This task is commonly referred to as *authenticated group key establishment* (GAKE). The common approach when aiming at *quantum-resistant* GAKE, is to either consider a solution relying entirely on quantum cryptography, specifically quantum key

*Corresponding author.
[1] Throughout, the word *classical* will always mean "not making use of quantum technologies".

distribution (QKD), or a pure post-quantum protocol, e.g. leveraging a post-quantum key encapsulation mechanism (KEM), which is a public-key design through which one party generates and transports securely a cryptographic key to a designated peer.

While large-scale QKD networks remain unavailable, non-trivial local QKD infrastructures have been implemented. An incomplete list of examples are testbeds in the Chicago area in the U.S. (see Wu *et al.*, 2021), in the Madrid area in Spain (see Cid *et al.*, 2021), and in the Berlin area in Germany (see Braun and Geitz, 2021). However, existing communication networks are not easy to combine/integrate with these quantum infrastructures, and the security guarantees for the complete network provided by establishing pairwise QKD keys are limited without other cryptographic solutions (most notably, for authentication, but also for refreshing keys or synchronizing multiple nodes). Toward this goal, connecting local QKD network infrastructures remains an active research topic (see Brauer *et al.*, 2024).

On the post-quantum side, there is a lot of recent work on the design, implementation, and verification of two-party key exchange protocols, mostly built from generic transformations applied to encryption schemes or KEMs. An overview of existing solutions is offered by Alagic *et al.* (2022), which summarizes the results of the third evaluation round of the NIST process toward standardizing post-quantum cryptographic solutions. In the group setting, GAKE proposals can already be found in the literature, for instance by Apon *et al.* (2019), Escribano Pablos *et al.* (2020), or Escribano Pablos *et al.* (2022), although these types of constructions have not been intensively analysed and little is known for instance, about implementation-dependent attacks.

From a pragmatic perspective, it is desirable to be able to leverage both existing quantum and post-quantum infrastructures, a scenario that is not well understood yet.

Our contribution. We present a construction and security analysis for a flexible quantum-safe GAKE that leverages available "local" solutions for (group) key establishment, e.g. post-quantum KEMs or QKD-based designs. Depending on the guarantees provided by the contributing local solutions, the resulting GAKE offers information-theoretic or computational guarantees. It is possible to use our design with classical primitives, QKD connections, or in a hybrid network.

We follow a well-established technique of defining a *compiler*, i.e. a metaprotocol using one or more GAKEs as embedded subroutines, which can be proven secure provided that a certain level of security is achieved by the basic building blocks involved. The flexibility of our design appears to be very useful with the current state-of-the art, as some quantum and post-quantum designs are still under cryptanalytic exploration.

To be able to capture a hybrid classical/quantum communication infrastructure, we build on a versatile security model proposed by Mosca *et al.* (2013), introducing some small adaptations. We believe this model to be of interest for exploring (group) key establishment over hybrid classical-quantum networks, independent of our specific GAKE construction.

Paper Roadmap. We begin this work with a brief overview of the related literature in Section 2, followed by an introduction to the security model we propose to evaluate our construction in Section 3. Section 4 provides a detailed description of our proposal, starting with an outline of the rationale behind the construction, and concluding with a formal security analysis in the final subsection. Given the various choices available for concrete implementation in our design, we discuss different options in Section 5. We conclude with a brief summary of the paper's main contributions and suggest potential directions for future research in the Conclusion section.

2. Related Work

In the literature, several works can be found exploring different ways of establishing secure keying material through the combination of quantum and classical technologies (see the summarizing Table 1 below).

A number of research contributions look at different ways to derive (two-party) cryptographic keys by *combining* keys established through different implementations of quantum, post-quantum, or traditional key exchange protocols (see, for instance, Dowling *et al.*, 2020; Bruckner *et al.*, 2023). This hybrid approach differs from ours, as it assumes that each party has access to *all* involved key sources, while in our scenario nodes have potentially different capabilities, e.g. we may have only few (or no) QKD connections available.

Also, in the last few years, several authors have explored the combination of classical and quantum resources in order to build secure networks, considering the QKD nodes to be the main source of keying material. For instance, in Viksna *et al.* (2023), Kozlovics *et al.* (2023), parties use QKD as a service to obtain secure keying material, which is accessed through (classical, post-quantum) TLS links. There, it is however assumed that there exist perfectly secure direct links between users and devices establishing QKD keys, which is a strong assumption. In Geitz *et al.* (2023), a high-level key management system is described through which all network nodes may access keying material (coming from a QKD, a post-quantum key exchange protocol, or a combination of those). Different protocols within these systems have been implemented in the OpenQKD testbed in Berlin. Similarly, in James *et al.* (2023), the authors explore different options for building a secure *Key Management System* in order to scale up from link-to-link quantum key generation to large key distribution networks.

Table 1
Comparison of (hybrid) quantum resistant key exchange constructions.

Protocol	Type	Contributing parties	Q-parties	PQ-parties
Dowling *et al.* (2020)	Hybrid AKE	2	2	2
Bruckner *et al.* (2023)	Hybrid AKE	2	2	2
Viksna *et al.* (2023)	QKD distribution/management	2	2	$n \leqslant 2$
Kozlovics *et al.* (2023)	QKD distribution/management	2	2	$n \leqslant 2$
Geitz *et al.* (2023)	QKD distribution/management	2	2	$n \leqslant 2$
James *et al.* (2023)	QKD distribution/management	2	2	$n \leqslant 2$
Our work	GAKE	$n \geqslant 2$	any j in $\{2, n\}$	any j in $\{2, n\}$

While all these approaches are in a way related to ours, their main goal is to establish *two-party* keys taking advantage of QKD installations and reinforcing them through PQC. Thus, they could in principle be implemented with “only” two actors involved in the key generation (through a QKD execution). We aim at a contributory scenario, i.e. a general protocol solution where $n \geqslant 2$ users leverage classical and/or quantum techniques in order to (jointly) generate keying material from both sources to establish a secure key for the whole group.

3. Security Model

In this section, we present a novel security model to enable a formal analysis of our proposal. While established models exist in the literature for group key establishment in the *classical* setting, we are not aware of any prior definition that formalizes the interaction between classical and quantum entities.

As a starting point, we base our work on a proposal by Mosca *et al.* (2013) and make some small changes to better capture some subtleties of special relevance to the group setting. For the special case of a two-party Authenticated Key Establishment (AKE), our model specializes to Mosca *et al.’s*. This “downward compatibility” enables us to compile a GAKE from existing popular two-party QKD (or classical key exchange) protocols that are already known to be secure in Mosca *et al.*’s framework.

Parties. We follow the formalization in Mosca *et al.* (2013) based on classical and quantum Turing machines, but make some tweaks to better align with classical group key exchange models (see, e.g. Bohli *et al.*, 2007). By $\overline{\mathcal{P}}$ we denote a finite set of potential protocol participants, each of them labelled by a public identifier. Each party can be seen as a dual entity consisting of an interactive Turning machine that communicates through a classical tape (e) with a coupled quantum Turing machine. Parties communicate through an input-output classical communication channel (c)—established between their “classical” parts. In turn, the quantum Turing machines from each party communicate through a quantum chanel (q). The classical Turing machine has also access to a randomness source (through a specific tape denoted r).

All classical information held by a specific party is stored in its *memory*, consisting of a collection of value pairs of the form (x, X) where x is assumed to be private and X to be public—often referred to as a *label* of the secret value x.

Remark 1. Our compiler does not require potential protocol participants to have access to quantum computational capabilities. Similarly, our protocol allows for a scenario where only some (possibly no) potential participants have the capability to execute a QKD protocol.

Session. A *protocol* can be seen as a well-defined sequence of classical and quantum interactions that will produce a private output to all involved parties, which is either a shared secret or an error message. Specific executions of the protocol are referred to as *sessions*.

Memory pairs held by a party may either be *ephemeral*, i.e. linked to a specific session, or *static*, that is, used across multiple sessions.

Each protocol participant $P \in \overline{\mathcal{P}}$ may execute several sessions in parallel. We will refer to a session of P with the notation Π_P^ψ, where Ψ is a unique *protocol session identifier*.[2] Each such session may be taken for a process executed by P and has assigned several variables to it: state_P^ψ, sid_P^ψ, pid_P^ψ, and sk_P^ψ, These variables are used as follows.

state_P^ψ keeps the state information during the protocol execution, i.e. it stores all memory pairs defined above, and mantains two vectors $\mathbf{u}$, $\mathbf{v}$ defined as follows:

- $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1 \dots)$ is a vector of public values or labels, whose components are vectors themselves. E.g., a vector $\mathbf{v}_i$ can store the ordered public values contributed by an involved participant in the session;
- $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1 \dots)$ is a vector storing in each component public values linked to an involved participant, yet these values are linked to the authentication process (thus, $\mathbf{u}$ will be referred to as the *authentication vector*).

sid_P^ψ denotes a (non-secret) global session identifier that can serve as the name for the session key sk_P^ψ.

pid_P^ψ stores, as unordered sequence, the public identifiers of those participants that Π_P^ψ aims at establishing a key with—not including P itself. We will further denote by $\overline{\mathsf{pid}}_P^\psi$ the unordered sequence of identifiers involved in a concrete execution, namely, the set of identifiers contained in pid_P^ψ along with the identifier id_P.

sk_P^ψ stores the session key once it is accepted by the session Π_P^ψ. Before acceptance, it stores a distinguished $\perp$ value.

We assume that a session Π_P^ψ must accept the session key constructed at the end of the corresponding protocol session if no deviation from the protocol specification occurs, resulting in a non-$\perp$ value of sk_P^ψ. In the sequel, sessions and session variables may be written without sub- or superscripts, if the context renders them unnecessary. The state_P^ψ variable mentioned above is merely a convenient notation to refer to multiple locally stored values. Otherwise, our terminology follows by and large Mosca *et al.* (2013) with the following generalizations:

- In the model of Mosca *et al.*, each session has associated a pid value to store the identifier of the (one) peer with whom the session is executed. We allow the pid to be an unordered non-empty sequence of peer identifiers instead. The *authentication vector* $\mathbf{u}$ now captures the information used to identify all peers included in the pid of the session. Further, as noted above, we will denote by $\overline{\mathsf{pid}}$ the set of all identities involved in the execution (namely, those in pid and that from party P itself).
- The classical Turing machine's output in Mosca *et al.* (2013) after a sucessful procotol completion is a tuple $(\mathsf{sk}, \mathsf{pid}, \mathbf{v}, \mathbf{u})$. Here, we allow an *optional* global session identifier

[2] To better align with the use of the term *session identifier* in other classical group key exchange models, we slightly deviate from the terminology in Mosca *et al.* (2013) here and refer to Ψ as *protocol session identifier*; a *session identifier* will be a value sid obtained as output from a protocol.

sid to be included, i.e. the output may have the form $(\mathsf{sk}, \mathsf{pid}, \mathbf{v}, \mathbf{u}, \mathsf{sid})$, where the first component—sk—is private output and the vector $(\mathsf{pid}, \mathbf{v}, \mathbf{u}, \mathsf{sid})$ is public output. If sid is not explicitly specified, it defaults to $\mathsf{sid} = \mathbf{v}$.

We now rephrase (Mosca *et al.*, 2013, Definition 1) in terms of the session identifier:

Definition 1 (*Correctness*). A key exchange protocol is said to be *correct* if, when all protocol messages are relayed faithfully, without changes to content or ordering, the peer parties output the same session key sk and the same session identifier sid.

For 2-party key exchange protocols as discussed in Mosca *et al.* (2013), which do not define an explicit session identifier, this specializes to correctness as used in Mosca *et al.* (2013). However, for our purposes, the ability to specify a separate session identifier will be convenient: we want to use existing, e.g. 2-party, protocols as building blocks of a protocol involving a larger number of participants. Here, the separation of $\mathbf{v}$ from the session identifier frees us from the problem of having to make "local" $\mathbf{v}$-values of a 2-party protocol available to other protocol participants, just to ensure correctness of the overall protocol.

Communication. Following Mosca *et al.* (2013), the classical Turing machine will have two incoming-outgoing classical communication channels (e and c). Communication through these channels is modelled through oracle queries:[3]

- $\mathsf{SendC}(params, \mathsf{pid})$: this query is received by a party $P \in \overline{\mathsf{pid}}$ through the c-channel and directs P to begin a new protocol execution, assigning to it a chosen internal protocol session identifier ψ and setting up $\mathsf{pid}_P^{\psi} = \mathsf{pid}$.
- $\mathsf{SendC}(\psi, m)$: the party receiving this query will receive the classical message m over the c-channel.
- $\mathsf{Q2C}(m)$: this models internal communication from the quantum to the classical Turing machine of a party; the classical Turing machine receives the message m.
- $\mathsf{SendQ}(\rho)$: the party receiving this query will receive the quantum message ρ over the q-channel, activating its quantum Turing machine, which will return any outgoing quantum message over the q-channel and any classical message over the e-channel (towards the classical Turing machine).
- $\mathsf{C2Q}(m)$: this models internal communication from the classical to the quantum Turing machine of a party; the latter receives the classical message m and may output an outgoing quantum message on the q-channel or a classical message on the e-channel.

For further details on these oracle queries, we refer to Mosca *et al.* (2013).

Adversarial model. To a large extent, the communication network is controlled by the adversary (following the typical modelling in classical group key exchange). We assume arbitrary point-to-point connections among users to be available, involving for each party

[3] In Mosca *et al.* (2013), oracle queries are called *activations*, following the Turing-machine modelling terminology.

the aforementioned classical (c) and quantum (q) channels. With respect to these channels, the network is non-private and fully asynchronous: The adversary may delay, eavesdrop, insert, and delete messages at will, only limited—with regard to the q-channel—by the laws of quantum mechanics. However, the adversary has no control over the communication between the classical and quantum subcomponents of a party, which takes place over the e-channel; neither will he have access to any information concerning the randomness obtained through the r-channel. In addition to the SendC and SendQ oracles, the adversary may also issue the following two types of queries to parties:

- $\mathsf{RevealNext}$; this query allows the adversary to get the public tag X of a freshly generated memory entry (x, X) by the addressed party.
- $\mathsf{Partner}(X)$; when addressed to a certain party, if the pair (x, X) exists in its memory, it forwards to the adversary the secret value x. $\mathsf{Partner}(\psi)$ returns the secret key for the corresponding session, if it exists (thus modelling so-called session-key reveals).

Non-internal oracle calls (namely, SendC, SendQ, $\mathsf{RevealNext}$, $\mathsf{Partner}$) should, in order to avoid ambiguity, include (P, ψ) as part of the input to clarify which party and which corresponding session are being addressed.

Security. As customary, a special Test oracle is introduced in order to model a real attack and be able to define security:

- $\mathsf{Test}(P, \psi)$, when called by the adversary, will return $\perp$ if a corresponding session key sk_P^{ψ} has not been established. Otherwise, the oracle selects uniformly at random a bit $b \overset{\$}{\leftarrow} \{0, 1\}$ and will output sk_P^{ψ} if $b = 1$ and a randomly selected bitstring (of the same length as sk_P^{ψ}) if $b = 0$. The adversary may call this oracle only once.

Indeed, the security model must rule out trivial attacks, i.e. those for which the adversary will know the established session key from its interaction with the involved parties. This is done in Mosca *et al.* (2013) by introducing the following definition of *freshness*:

Definition 2 (Mosca *et al.*, 2013, *2*). A session ψ of a party P is *fresh* provided that, if $\mathsf{out} = (\mathsf{pid}, \mathbf{v}, \mathbf{u}, \mathsf{sid})$ is the corresponding public output vector, the following hold:

- for every component $\mathbf{v}_i$ of $\mathbf{v}$ there is at least one public label X in $\mathbf{v}_i$ such that the adversary never queried $\mathsf{Partner}(X)$, and
- no query of the form $\mathsf{Partner}(\psi')$ has been made by the adversary on a session ψ' with the same output vector out,[4] and
- at the time of session completion, for every component $\mathbf{u}_i$ from $\mathbf{u}$, there was at least one public label X in $\mathbf{u}_i$, such that the adversary did not query $\mathsf{Partner}(X)$.

Note that *all* authentication-related private values may be revealed after protocol completion (the above definition only ensures there will be at least one private value the adversary ignores at all times, which is linked to the vector $\mathbf{v}$). This definition of freshness

[4]In particular, the query $\mathsf{Partner}(\Psi)$ violates freshness.

is geared towards including forward security in the model; leakage of authentication keys should not endanger previously established session keys.

Now, the definition of security must establish a corresponding bound on the adversary's *advantage* when querying the Test oracle. Let $\lambda \in \mathbb{N}$ be a fixed security parameter. The advantage $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ of an adversary $\mathcal{A}$ in attacking the given protocol is a function in the security parameter λ, defined as

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

Here, $\mathsf{Succ}_{\mathcal{A}}$ is the probability that the adversary queries Test on a fresh instance and guesses correctly the bit b used by the Test oracle.

Definition 3. We say that an authenticated group key establishment protocol P is *secure* if for every adversary $\mathcal{A}$ the following inequality holds for some negligible function negl:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) \leqslant \mathrm{negl}(\lambda). \tag{1}$$

In the above definition, following Mosca *et al.* (2013), the classical running time of adversaries is assumed to be bounded, and so are the quantum runtime and the quantum memory. We consider here only the case of a polynomially bounded adversary, which is common in classical group key establishment models. However, in connection with quantum key distribution it makes sense to consider so-called *long-term security*, introduced by Müller-Quade and Unruh (2010), capturing the feature of a protocol that it will remain secure even if all hardness assumptions made no longer hold after the execution has finished. Following Mosca *et al.* (2013), we capture this as follows:

Definition 4. A protocol is *long-term* secure if, for any unbounded quantum Turing machine $\mathcal{M}$ acting on a classical and quantum transcript produced by a bounded adversary $\mathcal{A}$, the advantage of $\mathcal{M}$ in guessing the bit b chosen by the Test oracle is negligible in the security parameter.

3.1. *Adding Integrity to a Secure Key Exchange*

To analyse the protocol design proposed below, it will be helpful to adopt a notion of *integrity* as introduced by Bohli *et al.* (2007) in the context of classical GAKE. Formally, note that the given definition of freshness allows the adversary to reveal keys from sessions which have different output vectors; as a result, if two completed sessions holding different session identifiers result in the same session key, the adversary trivially wins the Test challenge by construction. On top of this, integrity guarantees correctness and usability in adversarial environments (for users know who they share the session key with and how to address it).

Definition 5 (*Integrity*). We say that a correct (group) key exchange protocol fulfills *integrity*, if, with overwhelming probability, for all sessions of honest parties that have completed with the same session identifier in their output ($\neq \perp$), the session keys and parties associated with the session key (i.e. the partner identifier combined with the identifier of the party itself, denoted by $\overline{\mathsf{pid}}$) are identical.

In the quantum random oracle model, the following simple observation enables us to augment a given secure AKE so that it fulfills integrity, in case it does not come with this guarantee already. Note that this approach also preserves long-term security, which is particularly desirable when working with QKD-based solutions.

Lemma 1. *Let* GAKE *be a correct and secure key establishment protocol, and write* $(\mathsf{sk}, \mathsf{pid}, \mathbf{u}, \mathbf{v}, \mathsf{sid})$ *for the output of a successfully completed session for party* P. *Assume that* sk *is a bitstring of length* $\geqslant 9\lambda$, *where* λ *is the security parameter, and write* $\mathsf{sk}[:\lambda]$ *for the* λ *left-most bits of* sk, *and* $\mathsf{sk}[\lambda:]$ *for the remaining bits of* sk. *With a random oracle* $H : \{0,1\}^* \longrightarrow \{0,1\}^\lambda$, *the following protocol* GAKE' *is correct, secure and fulfills integrity.* GAKE' *is identical to* GAKE, *but in case of a successful session completion outputs*

$$\left(\mathsf{sk}[:\lambda], \mathsf{pid}, \mathbf{u}, \mathbf{v}, H\left(\overline{\mathsf{pid}}, \mathsf{sk}[:\lambda], \mathit{sid}, \mathsf{sk}[\lambda:]\right)\right).$$

Namely, it sets the output secret key as $\mathsf{sk}[:\lambda]$, *and the corresponding session identifier as*

$$H\left(\overline{\mathsf{pid}}, \mathsf{sk}[:\lambda], \mathit{sid}, \mathsf{sk}[\lambda:]\right).$$

Moreover, if GAKE *is long-term secure, then* GAKE' *is long-term secure.*

Proof. The *correctness* of GAKE' follows immediately from the correctness of GAKE. To see that GAKE' is *secure*, note that $H(\overline{\mathsf{pid}}, \mathsf{sk}[:\lambda], \mathsf{sid}, \mathsf{sk}[\lambda:])$ implements a statistically hiding commitment on $(\overline{\mathsf{pid}}, \mathsf{sk}[:\lambda], \mathsf{sk}[\lambda:], \mathsf{sid})$ in the quantum oracle model—this follows from (Unruh, 2022, Lemma 17), which attributes the result to (Pass, 2004, Lemma 9). So any successful adversary on GAKE' could immediately be turned into a successful adversary on GAKE. And as the hiding property of this commitment scheme does not rely on a computational assumption, *long-term security* is preserved.

Finally, *integrity* follows from a result by Zhandry (Zhandry, 2015, Theorem 3.1) with an argument by Unruh (Unruh, 2022, Footnote 13), establishing the quantum random oracle H as collison-resistant. $\square$

4. The Proposed Construction – A Compiler

In this section, we give a detailed description of our proposal. We start by sketching the main idea behind our design and then give the details of the protocol steps. We then state and prove the main result (Theorem 1) stating the security of our construction.

Let $\mathcal{P} \subseteq \overline{\mathcal{P}}$ be the set of parties that seek to establish a common key in a concrete execution, and let

$$\mathcal{P} = \mathcal{P}^{(0)} \uplus \cdots \uplus \mathcal{P}^{(n-1)}$$

be a partition of $\mathcal{P}$ into non-empty subsets or "clusters"—an important special case is the partition where each $\mathcal{P}^{(i)}$ contains only a single party. For each $\mathcal{P}^{(i)}$, we fix a *cluster leader* $\hat{P}^{(i)} \in \mathcal{P}^{(i)}$, and assume that

- for each $i = 0, \dots, n-1$, a secure $\mathsf{GAKE}^{(i)}$ solution among the parties from cluster i is available, which we denote by $\mathsf{GAKE}^{(i)}$. We write $(\mathbf{c}\mathsf{sk}_P^{(i)}, \mathbf{c}\mathsf{pid}_P^{(i)}, \mathbf{cv}_P^{(i)}, \mathbf{cu}_P^{(i)}, \mathbf{c}\mathsf{sid}_P^{(i)})$ for the output vector of a party P of an execution of $\mathsf{GAKE}^{(i)}$;
- for $i = 0, \dots n-1$, a secure 2-$\mathsf{AKE}^{(i)}$ protocol can be executed among the leaders $\hat{P}^{(i)} \in \mathcal{P}^{(i)}$ and $\hat{P}^{((i+1) \bmod n)} \in \mathcal{P}^{((i+1) \bmod n)}$. We write $(\mathsf{sk}_{\hat{P}}^{(i)}, \overline{\mathsf{pid}}_{\hat{P}}^{(i)}, \mathbf{v}_{\hat{P}}^{(i)}, \mathbf{u}_{\hat{P}}^{(i)}, \mathsf{sid}_{\hat{P}}^{(i)})$ for the corresponding output vector of a cluster leader $\hat{P}$.

4.1. *High-Level Description*

We start with a high-level description of our compiler and fill in the details in Fig. 4 in the next section. First, we note that for the "trivial" case when each cluster consists of a single party, this protocol follows a "standard" Burmester-Desmedt rationale, through which two-party keys established in pairs in a ring configuration are combined in order to derive a group key. The general case can be broken down into three steps:

Step A. During a first phase, all parties within each cluster $\mathcal{P}^{(i)}$ execute $\mathsf{GAKE}^{(i)}$ to set up a shared "cluster key" $\mathbf{c}\mathsf{sk}^{(i)}$ among all parties in that cluster. This value will play the role of an *ephemeral masking value* for this execution. Also, possibly in parallel to the $\mathsf{GAKE}^{(i)}$ execution, each leader $\hat{P}^{(i)}$ executes two sessions in parallel of the two-party key establishment, 2-$\mathsf{AKE}^{(i)}$ and 2-$\mathsf{AKE}^{((i-1) \bmod n)}$, in order to establish two keys, shared with its "counterclockwise neighbour" $\hat{P}^{((i+1) \bmod n)}$ and "clockwise neighbour" $\hat{P}^{((i-1) \bmod n)}$ respectively. We denote these two keys by $\mathsf{sk}_{\hat{P}^{(i)}}^{\circlearrowleft}$ and $\mathsf{sk}_{\hat{P}^{(i)}}^{\circlearrowright}$, respectively. From this point on, cluster leaders will act on behalf of all participants in the cluster and will be the only ones to send/receive messages to/from outside the cluster. Figure 1 illustrates an example setup for Step A.

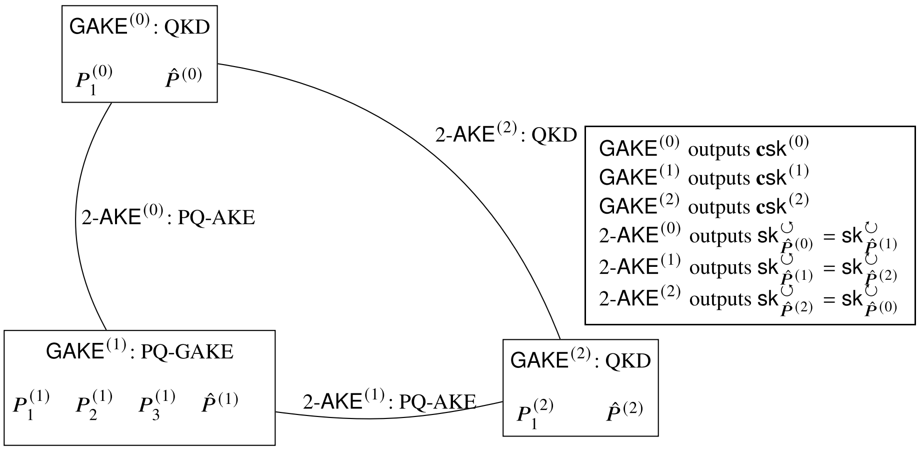
GAKE$^{(0)}$: QKD
$P_1^{(0)}$
$\hat{P}^{(0)}$
2-AKE$^{(2)}$: QKD
2-AKE$^{(0)}$: PQ-AKE
GAKE$^{(0)}$ outputs $\mathbf{c}\mathsf{sk}^{(0)}$
GAKE$^{(1)}$ outputs $\mathbf{c}\mathsf{sk}^{(1)}$
GAKE$^{(2)}$ outputs $\mathbf{c}\mathsf{sk}^{(2)}$
2-AKE$^{(0)}$ outputs $\mathsf{sk}_{\hat{P}^{(0)}}^{\circlearrowleft} = \mathsf{sk}_{\hat{P}^{(1)}}^{\circlearrowright}$
2-AKE$^{(1)}$ outputs $\mathsf{sk}_{\hat{P}^{(1)}}^{\circlearrowleft} = \mathsf{sk}_{\hat{P}^{(2)}}^{\circlearrowright}$
2-AKE$^{(2)}$ outputs $\mathsf{sk}_{\hat{P}^{(2)}}^{\circlearrowleft} = \mathsf{sk}_{\hat{P}^{(0)}}^{\circlearrowright}$
GAKE$^{(1)}$: PQ-GAKE
$P_1^{(1)}$ $P_2^{(1)}$ $P_3^{(1)}$ $\hat{P}^{(1)}$
2-AKE$^{(1)}$: PQ-AKE
GAKE$^{(2)}$: QKD
$P_1^{(2)}$
$\hat{P}^{(2)}$

Fig. 1. Step A in a 3-cluster configuration with two clusters of size two and one cluster of size four, invoking protocols based on post-quantum (PQ) cryptography and quantum key distribution (QKD).

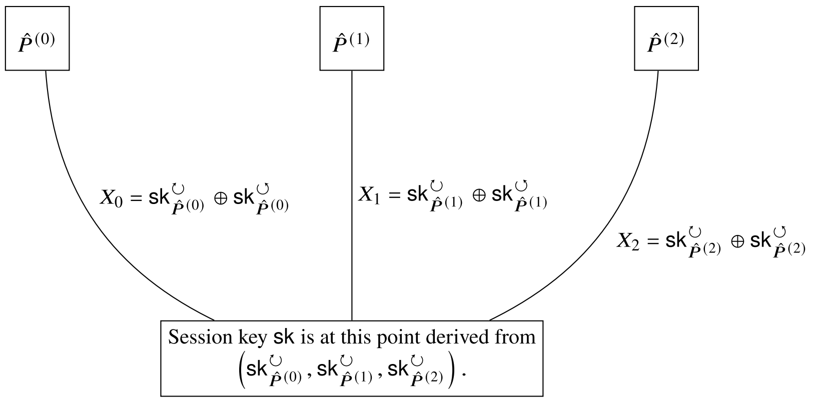
$\hat{P}^{(0)}$
$\hat{P}^{(1)}$
$\hat{P}^{(2)}$
$X_0 = \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(0)}} \oplus \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(0)}}$
$X_1 = \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(1)}} \oplus \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(1)}}$
$X_2 = \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(2)}} \oplus \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(2)}}$
Session key sk is at this point derived from $\left(\mathsf{sk}^{\circlearrowright}_{\hat{P}^{(0)}}, \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(1)}}, \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(2)}}\right)$.

Fig. 2. Steps B and C1 (only cluster leaders act) in our example 3-cluster configuration.

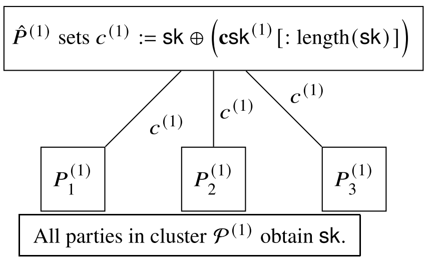
$\hat{P}^{(1)}$ sets $c^{(1)} := \mathsf{sk} \oplus \left(\mathsf{csk}^{(1)}[: \mathrm{length}(\mathsf{sk})]\right)$
$c^{(1)}$
$c^{(1)}$
$c^{(1)}$
$P_1^{(1)}$
$P_2^{(1)}$
$P_3^{(1)}$
All parties in cluster $\mathcal{P}^{(1)}$ obtain sk.

Fig. 3. Step C2: key transport inside cluster $\mathcal{P}^{(1)}$ (for $\mathcal{P}^{(0)}$ and $\mathcal{P}^{(2)}$ this step is analogous).

Step B. Now, each cluster leader $\hat{P}^{(i)}$ commits (using a public random oracle H) to the exclusive-or X_i of the two keys shared with neighbouring leaders. This commitment is contained in a first message (Step B1) M_i^B, while $\hat{P}^{(i)}$ sends in Step B2 the actual value X_i, together with the randomness needed to open the commitment (see Fig. 2).

Step C. In a final phase, each cluster leader recovers all two-party keys exchanged throughout the ring of leaders (namely, $\mathsf{sk}^{\circlearrowright}_{\hat{P}^{(0)}}, \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(1)}}, \ldots, \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(n-1)}}$) and derives a group key and a corresponding session identifier from them (see Fig. 3). Again, the public random oracle H is leveraged here. Finally, each cluster leader broadcasts within its cluster the sk, masked with the lower bits from the key $\mathsf{csk}^{(i)}$ and authenticated with the upper bits of this cluster key, using an unconditionally secure message authentication code (MAC) .

Remark 2. One could consider alternative approaches for the final key transport, e.g. using a computationally secure AEAD primitive (authenticated encryption with associated data). The protocol security would then, of course, be conditioned on the security of the specific AEAD primitive.

Remark 3. The primary communication bottleneck in our construction is the number of involved clusters, as this significantly influences both the number of exchanged messages and the local computations performed by cluster leaders. These exchanges are vulnerable to failures and depend on synchronization and network reliability, which may not always be guaranteed in certain application scenarios. Furthermore, leaders must maintain awareness of their *index* or position to correctly identify their clockwise and counterclockwise neighbuors. These constraints present major challenges when considering dynamic groups of participants, making our construction unsuitable for such scenarios.

4.2. *Detailing the Protocol Steps*

Let us look more closely at the individual steps of the proposed compiler. The individual steps are shown in Fig. 4, and we are left with specifying the vectors **v** and **u** output at the end of Step C2.

For the pertinent session of each protocol participant $P \in \mathcal{P}^{(i)}$ $(i = 0, \dots, n-1)$, we let

$$\mathbf{u} = \begin{cases} \mathbf{cu}_P^{(i)} || \mathbf{u}_P^{((i-1) \bmod n)} || \mathbf{u}_P^{(i)} || ((\ell(\mathsf{csk}^{(i)}[\text{length}(\mathsf{sk}) :]))), & \text{if } P \text{ is the cluster leader of } \mathcal{P}^{(i)}, \\ \mathbf{cu}_P^{(i)} || ((\ell(\mathsf{csk}^{(i)}[\text{length}(\mathsf{sk}) :]))), & \text{else;} \end{cases}$$

$$\mathbf{v} = \begin{cases} \mathbf{cv}_P^{(i)} || \mathbf{v}_P^{((i-1) \bmod n)} || \mathbf{v}_P^{(i)} || ((r_j), (X_j), (H(0, X_j, r_j)))_{j=0,\dots,n-1} & \\ \quad || ((c^{(i)}), (t^{(i)}), \dots\dots, (\ell(\mathsf{csk}^{(i)}[: \text{length}(\mathsf{sk})]))), & \text{if } P = \hat{P}^{(i)}, \\ \mathbf{cv}^{(i)} || ((c^{(i)}), (t^{(i)}), (\ell(\mathsf{csk}^{(i)}[: \text{length}(\mathsf{sk})]))), & \text{else.} \end{cases}$$

Following the notation in Mosca *et al.* (2013), here $\ell(\cdot)$ is a unique public label for a private value (chosen independent of any private value, e.g. with a counter).

Correctness. Correctness of the protocol in Fig. 4 follows from the correctness of $\mathsf{GAKE}^{(0)}, \dots, \mathsf{GAKE}^{(n-1)}$ and 2-$\mathsf{AKE}^{(0)}, \dots,$ 2-$\mathsf{AKE}^{(n-1)}$. Before going into the security analysis, it is worth noting that the GAKE in Fig. 4 does in general *not* ensure *strong entity authentication*—which is not a standard security goal: by design, only the cluster leads have to send messages in Steps B and C. This is attractive from the pespective of communication complexity, but as a consequence it is not clear that every party in $\mathcal{P}$ does indeed have possession of the established session key sk.

4.3. *Security Analysis*

The goal of this section is to establish the following result, which we prove by adapting the security analysis of a Kyber-based GAKE by Escribano Pablos *et al.* (2020), which builds on Abdalla *et al.* (2007). Analogously as Escribano Pablos *et al.* (2020), we impose integrity of the underlying 2-AKE and GAKE protocols to defend against an attack as considered by Nam *et al.* (2011).

Theorem 1. *Let* λ *be the security parameter and* MAC *an information-theoretically secure message authentication code with secret-key length* $m(\lambda)$ *for some polynomial* $m(\lambda) \in \Omega(\lambda)$. *Assume further that* 2-$\mathsf{AKE}^{(0)}, \dots,$ 2-$\mathsf{AKE}^{(n-1)}$ *and* $\mathsf{GAKE}^{(0)}, \dots, \mathsf{GAKE}^{(n-1)}$ *are secure key exchange protocols satisfying integrity, and each* 2-$\mathsf{AKE}^{(i)}$ $(i = 0, \dots, n-1)$ *outputs a session key of length* $p(\lambda)$ *for some polynomial* $p(\lambda) \geqslant 8\lambda$,

Step A: GAKE and 2-AKE executions

A1. Intra-cluster interaction. Each cluster $\mathcal{P}^{(i)}$ executes $\mathsf{GAKE}^{(i)}$, to establish a shared key $\mathbf{csk}^{(i)}$ among all parties in $\mathcal{P}^{(i)}$. Let $\mathbf{cv}_P^{(i)}$ and $\mathbf{cu}_P^{(i)}$ be the corresponding vector of public values and authentication vector of party $P \in \mathcal{P}^{(i)}$.

A2. Leaders interaction. Each cluster leader $\hat{P}^{(i)}$ $(i = 0, \ldots, n-1)$ executes 2-$\mathsf{AKE}^{(i)}$ with $\hat{P}^{((i+1) \bmod n)}$ and 2-$\mathsf{AKE}^{((i-1) \bmod n)}$ with $\hat{P}^{((i-1) \bmod n)}$, holding as a result an ephemeral shared key $\mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(i)}}$ with the cluster lead $\hat{P}^{((i+1) \bmod n)}$ and an ephemeral shared key $\mathsf{sk}^{\circlearrowright}_{\hat{P}^{(i)}}$ with $\hat{P}^{((i-1) \bmod n)}$. Let $\mathbf{u}^{(j)}_{\hat{P}^{(i)}}$ and $\mathbf{v}^{(j)}_{\hat{P}^{(i)}}$ be the $\mathbf{u}$- and $\mathbf{v}$-vector of $\hat{P}^{(i)}$ (as defined in the security model), resulting from the execution of 2-$\mathsf{AKE}^{(j)}$ $(j \in \{i, (i-1) \bmod n\})$.

Step B: Leaders Communication.

B1. Each cluster lead $\hat{P}^{(i)}$ computes the ephemeral value $X_i = \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(i)}} \oplus \mathsf{sk}^{\circlearrowright}_{\hat{P}^{(i)}}$ and broadcasts the message

$$M_i^{B1} = \left(\hat{P}^{(i)}, H(0, X_i, r_i)\right)$$

with a uniformly at random chosen ephemeral $r_i \in \{0,1\}^{8\lambda}$ $(i = 0, \ldots, n-1)$.

B2. After having received all messages from Step B1, each cluster lead $\hat{P}^{(i)}$ broadcasts the message

$$M_i^{B2} = \left(\hat{P}^{(i)}, X_i, r_i\right).$$

Step C: Key establishment.

C1. Key Computation. Each cluster lead $\hat{P}^{(i)}$ verifies that

- the commitments $H(0, X_0, r_0), \ldots, H(0, X_{n-1}, r_{n-1})$ from Step A have been opened correctly, and
- $\bigoplus_{i=0}^{n-1} X_i$ is the all-zero bitstring.

If any of these checks fails, $\hat{P}^i$ aborts by outputting an error symbol $\perp$.
Otherwise, it computes the ephemeral values $\mathsf{sk}^{\circlearrowright}_{\hat{P}(0)}, \ldots, \mathsf{sk}^{\circlearrowright}_{\hat{P}(n-1)}$ as

$$\mathsf{sk}^{\circlearrowright}_{\hat{P}((i+k) \bmod n)} = \mathsf{sk}^{\circlearrowright}_{\hat{P}(i)} \oplus X_{(i+1) \bmod n} \oplus \cdots \oplus X_{(i+k) \bmod n} \quad (k = 1, \ldots, n-1),$$

sets pid as the sequence of identifiers of users in $\mathcal{P} \setminus \{\hat{P}^{(i)}\}$, and sets up the session key as

$$\mathsf{sk} = H(10, \overline{\mathsf{pid}}, \mathsf{sk}^{\circlearrowright}_{\hat{P}(0)}, \ldots, \mathsf{sk}^{\circlearrowright}_{\hat{P}(n-2)})$$

and the session identifier as

$$\mathsf{sid} = H(11, \overline{\mathsf{pid}}, \mathsf{sk}^{\circlearrowright}_{\hat{P}(0)}, \ldots, \mathsf{sk}^{\circlearrowright}_{\hat{P}(n-1)}).$$

C2. Key transport. Each cluster leader defines

$$c^{(i)} := \left(\mathsf{sk} \oplus \left(\mathbf{csk}^{(i)}[: \mathrm{length}(\mathsf{sk})]\right), \mathsf{pid}, \mathsf{sid}\right) \text{ and } t^{(i)} := \mathrm{MAC}_{\mathbf{csk}^{(i)}[\mathrm{length}(\mathsf{sk}):]}(c^{(i)}),$$

and broadcasts within its cluster the message $M_i^C = (\hat{P}^{(i)}, c^{(i)}, t^{(i)})$.
Then, each participant $P_i \in \mathcal{P}$ constructs the output vector $(\mathsf{sk}, \mathsf{pid}, \mathbf{v}, \mathbf{u}, \mathsf{sid})$, with $\mathbf{u}$ and $\mathbf{v}$ as explained in Section 4.2. After deleting all ephemeral values from memory, the session is completed.

Fig. 4. The proposed GAKE.

while the output key of each $\mathsf{GAKE}^{(i)}$*, for* $i = 0, \ldots, n-1$*, has bitlength* $m(\lambda) + p(\lambda)$*. Then, in the quantum random oracle model, the protocol in Fig. 4 is a secure key exchange protocol fulfilling integrity.*

Moreover, if all of $2\text{-}\mathsf{AKE}^{(0)}, \ldots, 2\text{-}\mathsf{AKE}^{(n-1)}, \mathsf{GAKE}^{(0)}, \ldots, \mathsf{GAKE}^{(n-1)}$ *are long-term secure, then the protocol in Fig. 4 is long-term secure, too.*

Proof. Before establishing security, we want to establish integrity of the proposed protocol.

Integrity. From (Zhandry, 2015, Theorem 3.1) and (Unruh, 2022, Footnote 13) we obtain that the quantum random oracle H is collison-resistant. As a result, if two output vectors have coinciding sids, then the corresponding inputs to H must, with overwhelming probability, coincide, namely, they both share the same $\overline{\mathsf{pid}}$ and the same $\mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(0)}}, \ldots, \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(n-1)}}$. By construction, with overwhelming probability, these two sessions must have identical values for the secret key $\mathsf{sk} = H(\mathtt{10}, \overline{\mathsf{pid}}, \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(0)}}, \ldots, \mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(n-2)}})$.

Security. To establish security of the proposed protocol, we follow to a large extent the game-based reasoning in the proofs of (Abdalla *et al.*, 2007, Theorem 1) and (Escribano Pablos *et al.*, 2020, Theorem 3), making some necessary adaptations. We assume the adversary is interacting with a simulator, which is simulating all oracles and sessions for the adversary.

We 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. From game to game the simulator's behaviour is modified, in a way that allows us to bound the difference in the adversary's advantage between any two consecutive games. We omit the security parameter in the discussion, namely, writing $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ from Definition 3 simply as $\mathsf{Adv}_{\mathcal{A}}$. Furthermore, following standard notation, we denote by $\mathsf{Adv}_{\mathcal{A}}(G_i)$ the advantage of the adversary $\mathcal{A}$ in Game i.

Game 0. This first game corresponds to a real attack, in which all the parameters 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 $i = 0, \ldots, n-1$, we modify the simulation of the SendC and SendQ oracles so that, whenever a session ψ is still considered fresh after Step A1, the corresponding keys $\mathbf{csk}^{(i)}$ shared by the involved users are replaced with random bitstrings of the appropriate size.

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 $\mathsf{GAKE}^{(i)}$ protocols.

As a result, it holds

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_1) - \mathsf{Adv}_{\mathcal{A}}(G_0)\right| \leqslant \max_{i=0}^{n-1} \mathsf{Adv}_{\mathcal{A}}^{\mathsf{GAKE}^{(i)}}.$$

Here, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{GAKE}^{(i)}}$ denotes the advantage of $\mathcal{A}$ against $\mathsf{GAKE}^{(i)}$, which should be a function of the security parameter λ and the number of different involved sessions (i.e. sessions for which either SendC or SendQ have been queried by $\mathcal{A}$).

Game 2. In this game, for $i = 0, \ldots, n-1$, we modify the simulation of the SendC and SendQ oracles so that, whenever a session ψ is still considered fresh after Step A2, the corresponding keys $\mathsf{sk}^{\circlearrowleft}_{\hat{P}^{(i)}}$ shared by the cluster leaders $\hat{P}^{(i)}$ and $\hat{P}^{((i+1) \bmod n)}$ for $i = 0, \ldots, n-1$ are replaced with n elements selected uniformly at random from the intended key space.

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-$\mathsf{AKE}^{(i)}$ protocols.

As a result, it holds

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_2) - \mathsf{Adv}_{\mathcal{A}}(G_1)\right| \leqslant 2 \cdot \max_{i=0}^{n-1} \mathsf{Adv}_{\mathcal{A}}^{2\text{-}\mathsf{AKE}^{(i)}}.$$

Here, $\mathsf{Adv}_{\mathcal{A}}^{2\text{-}\mathsf{AKE}^{(i)}}$ denotes the advantage of $\mathcal{A}$ against a concrete session of 2-$\mathsf{AKE}^{(i)}$ (note that two sessions, i.e. two test queries, each corresponding to one of the involved leaders, are to be linked to each actual execution, thus the factor 2 in the bound above). Again, this advantage is a function of the security parameter λ and the number of different queries to the involved sessions.

Game 3. In this game, we change the simulation of the SendC oracle so that a *fresh* instance ψ does not accept in Step C1 whenever for some i a message M_i^{B2} was not faithfully simulated (as would have been generated in that same session by the intended cluster leader).

Note that the adversary will only notice the difference if both the message M_i^{B2} is being fabricated or replayed from another execution in a way that it is consistent with M_i^{B1} (i.e. the hash $H(0, X_i, r_i)$ is consistent with the X_i value from M_i^{B2}) and, moreover, the bitstring $X_0 \oplus \cdots \oplus X_{n-1}$ is the all-zero bitstring. Given that, at this point, these X_i-values are distributed uniformly at random and independent of previous messages, the probability that the replayed/fabricated messages involve a matching X_i is negligible.

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_3) - \mathsf{Adv}_{\mathcal{A}}(G_2)\right| \leqslant \mathrm{negl}(\lambda).$$

Game 4. This game reproduces the modification also for the commitments in Step B1: The simulation of the SendC oracle changes so that a *fresh* instance ψ does not accept in Step C1 whenever one value $H(0, X_j, r_j)$ for $j \neq i$ is not faithfully simulated. The adversary will only notice the difference if the simulator can fabricate a matching value on the range of H that is actually equal to $H(0, X_j, r_j)$. Due to the preimage resistance of the (quantum) random oracle H (cf. Yamakawa and Zhandry, 2021, Corollary 4.13), the adversary's advantage diverges only negligibly from the previous game:

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_4) - \mathsf{Adv}_{\mathcal{A}}(G_3)\right| \leqslant \mathrm{negl}(\lambda).$$

Game 5. Note that, at this point, all messages produced in Step B are faithfully generated by the simulator. Now, in the simulation of the SendC oracle we modify how the values in the first component of $c^{(i)}$ are generated: they are simply bitstrings of the appropriate length, chosen uniformly at random. Indeed, as these values were one-time pad encryptions (with one-time keys, chosen uniformly at random) of the session key, they should again be indistinguishable from random, as a result

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_5) - \mathsf{Adv}_{\mathcal{A}}(G_4)\right| \leqslant \mathrm{negl}(\lambda).$$

Game 6. We now modify the simulation of the SendC oracle at the point of computing the session key and the corresponding session identifier. At this, the simulator chooses a session key $\mathsf{sk} \in \{0,1\}^{\lambda}$ uniformly at random and does the same with the sid.

Note that the (quantum) random oracle outputs are indistinguishable from random, so, indeed,

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_6) - \mathsf{Adv}_{\mathcal{A}}(G_5)\right| \leqslant \mathrm{negl}(\lambda).$$

Moreover, as the session key is now chosen uniformly at random (independently of any other value the adversary may have access to), it holds that

$$\mathsf{Adv}_{\mathcal{A}}(G_6) = 0,$$

which concludes the proof.

Long-term security. Suppose that an adversary has access to a protocol transcript and faces the challenge of distinguishing the corresponding established secret key from a random bitstring of the same length. More specifically, he has access to a public output vector

$$(\mathsf{pid}, \mathbf{v}, \mathbf{u}, \mathsf{sid})$$

and needs to decide whether a bit string sk^* has been chosen uniformly at random (in the appropriate space) or is the actual session key established in that session. We may assume, without loss of generality, that the output vector comes from a cluster leader $\hat{P}^{(i)}$. That is

$$\mathbf{u} = \left(\mathbf{cu}^{(i)}||\mathbf{u}^{(i-1)}||\mathbf{u}^{(i)}||\ell\left(\left[\mathsf{csk}^{(i)}\right]_R\right)\right)$$

and

$$\mathbf{v} = \left(\mathbf{cv}^{(i)}||\mathbf{v}^{(i-1)}||\mathbf{v}^{(i)}||\left\{r_j||X_j||H(0, X_j, r_j)\right\}_{j=0,\ldots,n-1}||c^{(i)}||t^{(i)}||\ell\left(\left[\mathsf{csk}^{(i)}\right]_L\right)\right).$$

We can again follow a game-based structure where the adversary interacts with a simulator providing him with the transcript. Note that in this case, we are considering a passive attack when the adversary just gets the public output vector and cannot interact with any party involved in the actual session.

Game 0. Again, we start with a game corresponding to the real attack, in which the transcript is faithfully simulated. By definition, $\mathsf{Adv}_{\mathcal{A}}(G_0) = \mathsf{Adv}_{\mathcal{A}}$.

Game 1. Now, we replace the X_j-values in the $\mathbf{v}$-vector with a bitstring of the same length, yet chosen uniformly at random. If we assume that all involved 2-$\mathsf{AKE}^{(i)}$s produce two-party keys that are indistinguishible from values selected uniformly at random (in a long-term sense), so are the corresponding exclusive-or-values of each neighbouring pair of keys, i.e. $X_i = \mathsf{sk}^{\circlearrowleft}_{\hat{P}(i)} \oplus \mathsf{sk}^{\circlearrowright}_{\hat{P}(i)}$. As a result,

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_1) - \mathsf{Adv}_{\mathcal{A}}(G_0)\right| \leqslant 2 \cdot \max_{i=0}^{n-1} \mathsf{Adv}_{\mathcal{A}}^{2\text{-}\mathsf{AKE}^{(i)}}.$$

Game 2. Now, replace each of the left-most components of the values $c^{(i)}$ with a uniformly at random chosen value from the corresponding key space. Assuming the $\mathsf{GAKE}^{(i)}$s are long-term secure, this should not be noticeable, for each session key is one-time pad encrypted with a bitstring that is indistinguishable from one selected uniformly at random, thus:

$$\left|\mathsf{Adv}_{\mathcal{A}}(G_2) - \mathsf{Adv}_{\mathcal{A}}(G_1)\right| \leqslant \max_{i=0}^{n-1} \mathsf{Adv}_{\mathcal{A}}^{\mathsf{GAKE}^{(i)}}.$$

Moreover, note that the session identifier

$$\mathsf{sid} = H\left(\mathtt{11}, \overline{\mathsf{pid}}, \mathsf{sk}^{\circlearrowright}_{\hat{P}(0)}, \dots, \mathsf{sk}^{\circlearrowright}_{\hat{P}(n-1)}\right)$$

leaks nothing about the established session key, for it can be seen as a statistically hiding commitment on the values $\mathsf{sk}^{\circlearrowright}_{\hat{P}(0)}, \dots, \mathsf{sk}^{\circlearrowright}_{\hat{P}(n-2)}$ using as randomness the value $\mathsf{sk}^{\circlearrowright}_{\hat{P}(n-1)}$ (which is, by assumption, long-term secure).

Summing up, at this point no information linked to the actual established session key is available to the adversary:

$$\mathsf{Adv}_{\mathcal{A}}(G_2) = 0. \qquad \square$$

5. Design Choices

We have provided a general compiled protocol that can be constructed using various concrete cryptographic tools, following the principles of *cryptoagility*, which aims for modular designs that can be easily updated if any of the underlying tools need to be replaced. In this section, we offer guidance on how to make appropriate choices for the building blocks of our protocol.

There are many candidates in the literature for deriving the 2-AKE and GAKE constructions needed for our design. Prominent two-party quantum-key distribution protocols as analysed by Mosca *et al.* (2013) offer QKD building blocks, which we can now combine to GAKE solutions in a flexible manner.

For post-quantum cryptography, one can try to leverage known techniques to build 2-party authenticated key-exchange protocols from simpler cryptographic primitives, e.g.

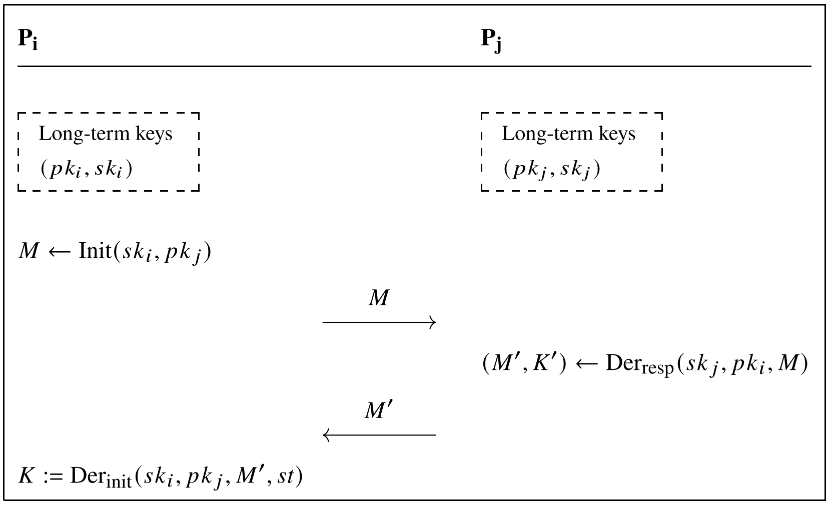
$\mathbf{P_i}$
$\mathbf{P_j}$
Long-term keys
(pk_i, sk_i)
Long-term keys
(pk_j, sk_j)
$M \leftarrow \mathrm{Init}(sk_i, pk_j)$
$\xrightarrow{M}$
$(M', K') \leftarrow \mathrm{Der}_{\mathrm{resp}}(sk_j, pk_i, M)$
$\xleftarrow{M'}$
$K := \mathrm{Der}_{\mathrm{init}}(sk_i, pk_j, M', st)$

Fig. 5. High-level decryption of a two-message 2-AKE.

from a given *Key Encapsulation Mechanism* (KEM) (cf., for instance, Hövelmanns *et al.* (2020)). A high-level description of such a 2-AKE construction is shown in Fig. 5. At this, both parties have a pair of long-term authentication keys. In the figure, P_i initiates the key exchange by executing an initiation algorithm Init and sending a message M to P_j, which through a key derivation algorithm $\mathrm{Der}_{\mathrm{resp}}$ computes the key K' and a message M'. Upon receipt of M', the receiver P_i is able to obtain the key K with a respective key derivation algorithm $\mathrm{Der}_{\mathrm{init}}$.

If the key exchange is successful, $K = K'$. In the quantum random oracle model, many such constructions (e.g. derived from Kyber, McEliece, or NTRU) are proven to be secure in a suitable sense. Moreover, as the key derivation algorithms involved are not deterministic, it is often the case that (some flavour of) forward security is attained – thus paving the way for *long-term* security in our setting. To this aim, often the Init algorithm involves a fresh key generation for a KEM, so that there will be an encapsulated key involved in the final output key that cannot be retrieved from the static long term keys of the involved participants.

For the classical group setting, solutions building on these two-party designs are available and may be implemented from different code- or lattice-based cryptographic building blocks (see Escribano Pablos *et al.*, 2020, 2022). It is a natural topic for follow-up research to try to establish general statements that enable an automatic lifting of security guarantees in one of the popular classical security frameworks for 2-AKE or GAKE protocols to security guarantees in our model.

6. Conclusion

In this paper, we have proposed the first (to our knowledge) construction of a group key establishment protocol including entities that may have access to quantum or classical technologies. In particular, we integrate clusters of users that may obtain keying material through quantum technologies (i.e. may execute QKD protocols in pairs) and clusters (of $n \geqslant 2$ entities) that are able to establish secure keys through post-quantum protocols. All keying material can be combined to establish a group key shared by all involved entities.

Our construction is proven secure in a formal model including adversaries that have access to quantum resources, thus attaining a very high security level. The security model introduced in this paper is based on Mosca *et al.* (2013), one of the few works exploring formal models of security for hybrid scenarios. We believe that this work is a step forward towards these kinds of formalization efforts, which are needed in order to derive sound security proofs in any scenario where quantum/classical technologies are combined.

Furthermore, we believe our construction could be adapted to support diverse network infrastructures. In particular, refining it to function effectively in scenarios with communication constraints would be especially valuable (e.g. to prevent aborts if some leaders from an initially designated cluster fail to complete the protocol). Exploring such a dynamic adaptation of our construction would be a promising avenue for future research.

Funding

This work has been supported by NATO SPS project G5985.

References

Abdalla, M., Bohli, J.-M., González Vasco, M.I., Steinwandt, R. (2007). (Password) Authenticated key establishment: from 2-party to group. In: Vadhan, S.P. (Ed.), *Theory of Cryptography*. Springer, Berlin, Heidelberg, pp. 499–514. https://doi.org/10.1007/978-3-540-70936-7_27.

Alagic, G., Cooper, D., Dang, Q., Dang, T., Kelsey, J.M., Lichtinger, J., Liu, Y.-K., Miller, C.A., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Apon, D. (2022). *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST Interagency/Internal Report (NISTIR). National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.IR.8413.

Apon, D., Dachman-Soled, D., Gong, H., Katz, J. (2019). Constant-round group key exchange from the ring-LWE assumption. In: *Post-Quantum Cryptography – 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers*, pp. 189–205. https://doi.org/10.1007/978-3-030-25510-7_11.

Bohli, J.-M., Gonzalez Vasco, M.I., Steinwandt, R. (2007). Secure group key establishment revisited. *International Journal of Information Security*, 6, 243–254. https://doi.org/10.1007/s10207-007-0018-x.

Brauer, M., Vicente, R.J., Buruaga, J.S., Mendez, R.B., Braun, R.-P., Geitz, M., Rydlichkowski, P., Brunner, H.H., Fung, F., Peev, M., Pastor, A., Lopez, D.R., Martin, V., Brito, J.P. (2024). Linking QKD testbeds across Europe. *Entropy*, 26(2), 123. https://doi.org/10.3390/e26020123.

Braun, R.-P., Geitz, M. (2021). The OpenQKD Testbed in Berlin. In: *2021 Asia Communications and Photonics Conference (ACP)*, pp. 1–3. https://doi.org/10.1364/ACPC.2021.M4C.2.

Bruckner, S., Ramacher, S., Striecks, C. (2023). Muckle+: end-to-end hybrid authenticated key exchanges. In: Johansson, T., Smith-Tone, D. (Eds.), *Post-Quantum Cryptography – 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16–18, 2023, Proceedings*. *Lecture Notes in Computer Science*, Vol. 14154. Springer, pp. 601–633. https://doi.org/0.1007/978-3-031-40003-2_22.

Cid, M.I.G., Martín, L.O., Ayuso, V.M. (2021). Madrid Quantum Network: a first step to quantum internet. In: Reinhardt, D., Müller, T. (Eds.), *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17–20, 2021*. ACM, pp. 102–11027. https://doi.org/10.1145/3465481.3470056.

Dowling, B., Hansen, T.B., Paterson, K.G. (2020). Many a mickle makes a muckle: a framework for provably quantum-secure hybrid key exchange. In: Ding, J., Tillich, J. (Eds.), *Post-Quantum Cryptography – 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings*. *Lecture Notes in Computer Science*, Vol. 12100. Springer, pp. 483–502. https://doi.org/10.1007/978-3-030-44223-1_26.

Escribano Pablos, J.I., Marriaga, M.E., del Pozo, A.L.P. (2022). Design and implementation of a post-quantum group authenticated key exchange protocol with the LibOQS Library: a comparative performance analysis from classic McEliece, Kyber, NTRU, and Saber. *IEEE Access*, 10, 120951–120983. https://doi.org/10.1109/access.2022.3222389.

Escribano Pablos, J.I., Gonzalez Vasco, M.I., Marriaga, M.E., Perez del Pozo, A.L. (2020). Compiled constructions towards post-quantum group key exchange: a design from kyber. *Mathematics*, 8(10). https://doi.org/10.3390/math8101853.

Geitz, M., Doering, R., Braun, R.-P. (2023). Hybrid QKD and PQC protocols implemented in the Berlin OpenQKD testbed. In: *2023 8th International Conference on Frontiers of Signal Processing (ICFSP)*, pp. 69–74. https://doi.org/10.1109/icfsp59764.2023.10372894.

Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D. (2020). Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (Eds.), *Public-Key Cryptography – PKC 2020*. Springer International Publishing, Cham, pp. 389–422. 978-3-030-45388-6. https://doi.org/10.1007/978-3-030-45388-6_14.

James, P., Laschet, S., Ramacher, S., Torresetti, L. (2023). Key management systems for large-scale quantum key distribution networks. In: *Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES 2023, Benevento, Italy, 29 August 2023–1 September 2023*. ACM, pp. 126–11269. https://doi.org/10.1145/3600160.3605050.

Kozlovics, S., Petrucena, K., Larins, D., Viksna, J. (2023). Quantum key distribution as a service and its injection into TLS. In: Meng, W., Yan, Z., Piuri, V. (Eds.), *Information Security Practice and Experience – 18th International Conference, ISPEC 2023, Copenhagen, Denmark, August 24–25, 2023, Proceedings*. *Lecture Notes in Computer Science*, Vol. 14341. Springer, pp. 527–545. https://doi.org/10.1007/978-981-99-7032-2_31.

Mosca, M., Stebila, D., Ustaoğlu, B. (2013). Quantum key distribution in the classical authenticated key exchange framework. In: Gaborit, P. (Ed.), *Post-Quantum Cryptography*. Springer, Berlin, Heidelberg, pp. 136–154. https://doi.org/10.1007/978-3-642-38616-9_9.

Müller-Quade, J., Unruh, D. (2010). Long-term security and universal composability. *Journal of Cryptology*, 23(4), 594–671. https://doi.org/10.1007/978-3-540-70936-7_3.

Nam, J., Paik, J., Won, D. (2011). A security weakness in Abdalla et als generic construction of a group key exchange protocol. *Information Sciences*, 181(1), 234–238. https://doi.org/10.1016/j.ins.2010.09.011.

Pass, R. (2004). *Alternative Variants of Zero-Knowledge Proofs*. Licentiate thesis. KTH Numerical Analysis and Computer Science, Stockholm. Available at http://www.cs.cornell.edu/~rafael/papers/raf-lic.pdf.

Unruh, D. (2022). Computationally binding quantum commitments. Cryptology ePrint Archive, Paper 2015/361. Recition 2. Available at https://eprint.iacr.org/2015/361. Major revision of an IACR publication in EUROCRYPT 2016.

Viksna, J., Kozlovics, S., Rencis, E. (2023). Integrating quantum key distribution into hybrid quantum-classical networks. In: *Applied Cryptography and Network Security Workshops – ACNS 2023*. *Lecture Notes in Computer Science*, Vol. 13907. Springer, pp. 695–699. https://doi.org/10.1007/978-3-031-41181-6_42.

Wu, W., Chung, J., Kanter, G., Lauk, N., Valivarthi, R., Ceballos, R.R., Pena, C., Sinclair, N., Thomas, J.M., Eastman, E.M., Xie, S., Kettimuthu, R., Kumar, P., Spentzouris, P., Spiropulu, M. (2021). Illinois express quantum network for distributing and controlling entanglement on metro-scale. In: *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, pp. 35–42. https://doi.org/10.1109/QCS54837.2021.00008.

Yamakawa, T., Zhandry, M. (2021). Classical vs quantum random oracles. In: *Advances in Cryptology— EUROCRYPT 2021*. *LNCS*, Vol. 12697. Springer, pp. 568–597.

Zhandry, M. (2015). A note on the quantum collision and set equality problems. *Quantum Information & Computation*, 15(7–8), 0557–0567. Preprint available as arXiv:1312.1027v3. https://doi.org/10.5555/2871411.2871413.

M.I. González Vasco is a full professor of Applied Mathematics at Universidad Carlos III de Madrid, specializing in mathematical cryptography. She holds a PhD in mathematics from the University of Oviedo and focuses on provable security and quantum-resistant cryptography. She has co-directed NATO-funded projects and collaborates with international institutions. González Vasco is currently vicepresident of the Real Sociedad Matemática Española.

R. Steinwandt is currently dean of the College of Science at the University of Alabama in Huntsville (UAH). Previously, he chaired the Department of Mathematical Sciences and directed the Center for Cryptology and Information Security at Florida Atlantic University (FAU). He earned his MS and PhD in computer science from the University of Karlsruhe, Germany, specializing in computer algebra. His research focuses on cryptology, including quantum cryptanalysis and quantum-safe cryptography, with funding from agencies such as the National Science Foundation, the National Institute of Standards and Technology, and NATO SPS.