CoverCrypt: an Early-Abort KEM for Hidden Access Policies with Traceability from the DDH and LWE

Théophile Brézot[1], Paola de Perthuis[1,2], Chloé Hébant[1], and David Pointcheval[2]

[1] Cosmian, Paris, France
[2] DIENS, École normale supérieure, PSL University, CNRS, Inria, Paris, France

Abstract. Attribute-Based Encryption (ABE) is a very attractive primitive to limit access according to specific rights. While very powerful instantiations have been offered, under various computational assumptions, they rely on either classical or post-quantum problems, and are quite intricate to implement, generally resulting in poor efficiency. With the threat of quantum computers, post-quantum solutions are important, but not yet fully convincing enough to rely on such problems only. We thus first study an hybrid approach to rely on the best of the two worlds: the scheme is secure if at least one of the two underlying assumptions is still valid (i.e. the DDH and LWE). Then, we address the ABE problem, with a practical solution delivering encrypted contents such that only authorized users can decrypt, without revealing the target sets, while also granting tracing capabilities. Our scheme is inspired by the Subset Cover framework where the users' rights are organized as subsets and a content is encrypted with respect to a subset covering of the target set. Quite conveniently, we offer black-box modularity: one can easily use any public-key encryption of their choice, such as Kyber, with their favorite library, to combine it with a simple ElGamal variant of key encapsulation mechanisms, providing strong security guarantees.

1 Introduction

Key Encapsulation Mechanisms (KEM) enable the transmission of symmetric keys at the beginning of an interaction while retaining trust that only the intended recipient will be able to get access to this encapsulated key. Once this trusted transmission has been established, users can privately communicate using this encapsulated secret symmetric key with the advantages of symmetric encryption, granting compact ciphertexts of similar size as corresponding cleartexts.

In organizations with complex structures, one will want to have more functionalities, namely being able to share a key among all users verifying a policy on a set of attributes, all at once. To this aim, KEMs constructed out of Attribute-Based Encryption (ABE) have been designed, in which keys can be encapsulated by being encrypted with these schemes for which all users verifying the specified attributes policy will be able to decrypt and thus decapsulate the key. These ABE primitives (stemming from [GPSW06]) are very powerful as they can cover any possible logical combination of the attributes, however this comes at an efficiency cost, and for practical use-cases, one will only need to encrypt for some of these existing combinations, for a limited number of attributes; this work is in this setting's scope, in which one can actually replace ABE constructions with encryption with respect to a union of attribute subsets. In these use-cases, it can also be relevant to get anonymity, meaning that a user should never know for which policy a ciphertext was produced, except if it is the policy they are using to successfully decrypt. In the case of ABE, this is called *attribute hiding*.

Additionally, with current preoccupations with respect to the threat of quantum computers on classical cryptography, granting resistance to these for data that needs to be kept private on the long term is becoming a necessity. However, since post-quantum cryptographic schemes are newer and only beginning to be used, one should try to also keep current schemes' security properties. In fact, some countries' security agencies are handing out guidelines for pre- and post-quantum security hybridization, meaning that cryptographic schemes should retain all their security properties even if one of the two pre- or post-quantum hybridized schemes is broken.

Another area of interest in this context in which users share some common keys, is the ability to still identify them uniquely, in case they choose to send some of there decapsulation capabilities to another party. Thus, if someone leaks some secret information they were supposed to keep to themselves, we would like to trace these so-called traitors, with *traceability*.

1.1 Related Work

This work combines many desirable properties for the use of KEMs in practical contexts, that other previous works had not, and since it covers less features than ABE-based constructions, it compares favorably in efficiency with respect to such post-quantum schemes built from ABE, in addition with providing traceability and post- and pre-quantum hybridization.

Anonymous Broadcast Encryption. Our simplified access structure with strong privacy has a similar flavor as previous works [LPQ12,FP12,LG18] on broadcast encryption with anonymity, with optimizations on the decryption time. However, they do not handle black-box post-quantum security nor traceability.

Post-Quantum Key-Policy ABE. Then, providing post-quantum resistance, the closest related works are Key-Policy ABEs (KP-ABE) based on LWE. Some theoretical works such as [Wee21] provide results with good asymptotic bounds, but are unsuited for use with practical parameters, and others, like [DDP+17], provide implementable results, but even with their comparable lowest policy circuit depth, their encryption time is about a hundred time bigger than ours, their decryption time about ten times bigger, and their RLWE parameters lead to bigger ciphertext sizes than ours. Also, they do not provide hybridization with pre-quantum schemes, nor anonymity nor traceability.

1.2 Our Contributions

Our final instantiation called CoverCrypt provides an efficient KEM for hidden access policies with traceability, withholding both pre- and post-quantum securities, along with a Rust implementation of the scheme[3].

An Efficient KEM with Hidden Acess Policies. Our scheme provides efficiency with respect to the state-of-the-art in post-quantum PK-ABE schemes by restricting its scope to depth-one policy circuits. The attributes for which a key is encapsulated are kept hidden, providing anonymity. Also, we gain time on the decryption with an early-abort paradigm, in which one can quickly test whether a ciphertext was encrypted for one of their attributes, using a tag, and retaining the anonymity properties of the scheme. Our ciphertexts are of size $96 + \#B \times 1088$ Bytes, where B is the list of attribute-subsets the key is encapsulated for. On the other hand, user's keys are of size $(\#A + 1) \times 64$ Bytes, where A is the list of attributes for the user. For $\#B$ ranging from 1 to 5, encapsulation takes from 350 to 950 microseconds, and decapsulation, from 230 to 480 microseconds, with an affine dependency in the user's attributes (see Section 8.2).

Hybridization for Pre and Post-Quantum Security. Our work, in the line of security agency recommendations, enables the hybridization of both pre- and post-quantum schemes, so that its security holds if either one of the schemes do. This is a novel contribution with respect to existing schemes which required to choose between classical and post-quantum securities, regardless of our trust in new post-quantum schemes. Also, the use of the post-quantum scheme is totally black-box, enabling combinations with other semantically secure public-key encryption schemes.

[3] We will make the source code available with an url in the final version.

Traceability. The pre-quantum ElGamal part of our scheme provides traceability under the Decisional Diffie-Hellman (DDH) assumption. It makes sense to consider traceability with pre-quantum security as this is a short-term security requirement, if users are currently misbehaving, whereas the post-quantum security preserves the anonymity and privacy properties, which are important on the long-term, as ciphertexts can be stored until their security is broken in the future. Our implementation covers the case were traitors do not collude; we also show how the scheme can be instantiated for arbitrarily t-large collusions, but the tracing time then grows exponentially in t.

2 Definitions

Public-Key Encryption (PKE) allows the transmission of hidden information that only the intended recipient will be able to uncover. To make the scheme independent of the format of the cleartext message, the usual paradigm for encryption is the KEM-DEM [Sho01], where one first encapsulates a session key that only the recipient can recover, and then encrypts the payload under that key. The former step uses a Key Encapsulation Mechanism (KEM) and the latter a Data Encoding Method (DEM), that is usually instantiated with an Authenticated Encryption, such as AES256-GCM[4], that provides both privacy and authenticity of plaintexts. As our work is built from KEMs, we hereafter recall some formal definitions.

2.1 Notations

Henceforth, many security notions will be characterized by the computational indistinguishability between two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$. It will be measured by the advantage an adversary $\mathcal{A}$ can have in distinguishing them as:

$$\mathsf{Adv}(\mathcal{A}) = \Pr_{\mathcal{D}_1}[\mathcal{A}(x) = 1] - \Pr_{\mathcal{D}_0}[\mathcal{A}(x) = 1] = 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}(x) = b] - 1.$$

Then, we will denote $\mathsf{Adv}(\tau)$ the maximal advantage over all the adversaries with running-time bounded by τ. A first pair of distributions is used in the famous ElGamal encryption scheme, with Diffie-Hellman tuples in $\mathbb{G} = \langle g \rangle$, a group of prime order p, spanned by a generator g, and denoted multiplicatively:

Decisional Diffie-Hellman Problem ($\mathsf{DDH}_{\mathbb{G}}$). The DDH assumption in a group $\mathbb{G}$ of prime order p, with a generator g, states that the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are hard to distinguish for any polynomial-time adversary, where

$$\mathcal{D}_0 = \{(g^a, g^b, g^{ab}), a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p\} \qquad \mathcal{D}_1 = \{(g^a, g^b, g^c), a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_p\}$$

and we will denote $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{A})$ the advantage of an adversary $\mathcal{A}$.

When studying the Kyber post-quantum encryption scheme, we will also need another algebraic structure, with indistinguishable distributions. We will denote $\mathsf{R} = \mathbb{Z}[X]/(X^n + 1)$ (resp. $\mathsf{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$) the ring of polynomials of degree at most $n - 1$ with integer coefficients (resp. with coefficients in $\mathbb{Z}_q$, for a small prime q). We take n as power of 2, where $X^n + 1$ is the $\frac{n}{2}$-th cyclotomic polynomial. We denote $\mathcal{B}_\eta$ the centered binomial distribution of parameter η. When a polynomial is sampled according to $\mathcal{B}_\eta$, it means each of its coefficient is sampled from that distribution. We will also use vectors $\mathbf{e} \in \mathsf{R}_q^k$ and matrices $\mathbf{A} \in \mathsf{R}_q^{m \times k}$ in R_q.

[4] `https://docs.rs/aes-gcm/latest/aes_gcm/`

Decisional Module Learning-with-Error Problem ($\mathsf{DMLWE}_{\mathsf{R}_q,m,k,\eta}$). The DMLWE assumption in R_q states that the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are hard to distinguish for any polynomial-time adversary, where

$$\begin{aligned}\mathcal{D}_0 &= \{(\mathbf{A},\mathbf{b}), \mathbf{A} \xleftarrow{\$} \mathsf{R}_q^{m\times k}, (\mathbf{s},\mathbf{e}) \xleftarrow{\$} \mathcal{B}\eta^k \times \mathcal{B}\eta^m, \mathbf{b} \leftarrow \mathbf{A}\mathbf{s}+\mathbf{e}\}\\ \mathcal{D}_1 &= \{(\mathbf{A},\mathbf{b}), \mathbf{A} \xleftarrow{\$} \mathsf{R}_q^{m\times k}, \mathbf{b} \xleftarrow{\$} \mathcal{B}\eta^m\}\end{aligned}$$

We will denote $\mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,m,k,\eta}(\mathcal{A})$ the advantage of an adversary $\mathcal{A}$.

2.2 Key Encapsulation Mechanism

A Key Encapsulation Mechanism KEM is defined by three algorithms:

- $\mathsf{KEM.KeyGen}(1^\kappa)$: the *key generation algorithm* outputs a pair of public and secret keys $(\mathsf{pk},\mathsf{sk})$;
- $\mathsf{KEM.Enc}(\mathsf{pk})$: the *encapsulation algorithm* generates a session key K and an encapsulation C of it, and outputs the pair (C,K);
- $\mathsf{KEM.Dec}(\mathsf{sk},C)$: the *decapsulation algorithm* outputs the key K encapsulated in C.

Correctness. A correct KEM satisfies $\mathsf{Adv}^{\mathsf{cor}}_{\mathsf{KEM}}(\kappa) = 1 - \Pr_{\mathcal{D}}[\mathsf{Ev}] = \mathsf{negl}(\kappa)$, for

$$\begin{aligned}\mathcal{D} &= \{(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa), (C,K) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}) : (\mathsf{sk},C,K)\}\\ \mathsf{Ev} &= [\mathsf{KEM.Dec}(\mathsf{sk},C) = K]\end{aligned}$$

Session-Key Privacy. On the other hand, such a KEM is said to provide *session-key privacy* (denoted $\mathsf{SK\text{-}IND}$) in the key space $\mathcal{K}$, if the encapsulated key is indistinguishable from a random key in $\mathcal{K}$. More formally, a KEM is $\mathsf{SK\text{-}IND}$-secure if for any adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) = \mathsf{negl}(\kappa)$, for $b \xleftarrow{\$} \{0,1\}$ and

$$\begin{aligned}\mathcal{D}_b &= \left\{\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa),\\ (C,K_0) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}), K_1 \xleftarrow{\$} \mathcal{K}\end{array} : (\mathsf{pk},C,K_b)\right\}\\ \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) &= 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}(\mathsf{pk},C,K_b) = b] - 1.\end{aligned}$$

Public-Key Privacy. One can additionally expect anonymity of the receiver, also known as *public-key privacy* (denoted $\mathsf{PK\text{-}IND}$), if the encapsulation does not leak any information about the public key. More formally, a KEM is $\mathsf{PK\text{-}IND}$-secure if for any adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) = \mathsf{negl}(\kappa)$, for $b \xleftarrow{\$} \{0,1\}$ and

$$\begin{aligned}\mathcal{D}_b &= \left\{\begin{array}{l}\text{For } i = 0,1 :\\ \quad (\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa),\\ \quad (C_i,K_i) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}_i)\end{array} : (\mathsf{pk}_0,\mathsf{pk}_1,C_b)\right\}\\ \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) &= 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}(\mathsf{pk}_0,\mathsf{pk}_1,C_b) = b] - 1.\end{aligned}$$

ElGamal-based KEM. In a group $\mathbb{G}$ of prime order p, with a generator g, one can define:

- $\mathsf{EG.KeyGen}(1^\kappa)$: sample random $\mathsf{sk} = x \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{pk} = h \leftarrow g^x$;
- $\mathsf{EG.Enc}(\mathsf{pk})$: sample a random $r \xleftarrow{\$} \mathbb{Z}_p$ and set $C \leftarrow g^r$ together with $K \leftarrow h^r$;
- $\mathsf{EG.Dec}(\mathsf{sk},C)$: output $K \leftarrow C^x$.

Under the DDH assumption in $\mathbb{G}$, this KEM is $\mathsf{SK\text{-}IND}$ with $\mathcal{K} = \mathbb{G}$. In addition, as the encapsulation $C = g^r$ is independent of the public key, it perfectly provides $\mathsf{PK\text{-}IND}$. The formal security proofs for an extended version of this scheme will be given later, we thus postpone the analysis of this scheme.

2.3 Key Encapsulation Mechanism with Access Control

A KEM with Access Control allows multiple users to access the encapsulated key K from C, according to a rule $\mathcal{R}$ applied on X in the user's key usk and Y in the encapsulation C. It is defined by four algorithms:

- $\mathsf{KEMAC.Setup}(1^\kappa)$ outputs the global public parameters PK and the master secret key MSK;
- $\mathsf{KEMAC.KeyGen}(\mathsf{MSK}, Y)$ outputs the user's secret key usk according to Y;
- $\mathsf{KEMAC.Enc}(\mathsf{PK}, X)$ generates a session key K and an encapsulation C of it according to X;
- $\mathsf{KEMAC.Dec}(\mathsf{usk}, C)$ outputs the key K encapsulated in C.

Correctness. A KEMAC is correct if $\mathsf{Adv}^{\mathsf{cor}}_{\mathsf{KEMAC}}(\kappa) = 1 - \Pr_{\mathcal{D}}[\mathsf{Ev}] = \mathsf{negl}(\kappa)$, for

$$\begin{aligned}\mathcal{D} &= \left\{\begin{array}{l}\forall(X,Y) \text{ such that } \mathcal{R}(X,Y)=1,\\ (\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{KEMAC.KeyGen}(1^\kappa),\\ \mathsf{usk} \leftarrow \mathsf{KEMAC.KeyGen}(\mathsf{MSK},Y),\\ (C,K) \leftarrow \mathsf{KEMAC.Enc}(\mathsf{PK},X)\end{array} : (\mathsf{usk},C,K)\right\}\\ \mathsf{Ev} &= [\mathsf{KEMAC.Dec}(\mathsf{usk},C) = K].\end{aligned}$$

Session-Key Privacy. As for the basic KEM, one may expect some privacy properties. Session-key privacy is modeled by indistinguishability of ciphertexts, even if the adversary has received some decryption keys, as soon as associated Y_i are incompatible with X ($\mathcal{R}(X, Y_i) = 0$). Such a KEMAC is said to be $\mathsf{SK\text{-}IND}$-secure in the key space $\mathcal{K}$ if for any adversary $\mathcal{A}$, that can ask any key usk_i, using oracle $\mathcal{O}\mathsf{KeyGen}(Y_i)$ that stores Y_i in the set $\mathcal{Y}$ and outputs $\mathsf{KEMAC.KeyGen}(\mathsf{MSK}, Y_i)$, $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}) = \mathsf{negl}(\kappa)$, for $b \overset{\$}{\leftarrow} \{0,1\}$ and

$$\begin{aligned}\mathcal{D}_b &= \left\{\begin{array}{l}(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{KEMAC.Setup}(1^\kappa),\\ (\mathsf{state},X) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(\cdot)}(\mathsf{PK}),\\ (C,K_0) \leftarrow \mathsf{KEMAC.Enc}(\mathsf{PK},X), K_1 \overset{\$}{\leftarrow} \mathcal{K}\end{array} : (\mathsf{state},C,K_b)\right\}\\ \mathsf{BadXY} &= [\exists Y_i \in \mathcal{Y}, \mathcal{R}(X,Y_i)=1]\\ \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}) &= 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(\cdot)}(\mathsf{state},C,K_b) = b \,|\, \neg\mathsf{BadXY}] - 1.\end{aligned}$$

We note the bad event BadXY (decided at the end of the game) should be avoided by the adversary, as it reduces its advantage: this indeed leads to a trivial guess, and this is considered as a non-legitimate attack.

Access-Control Privacy. In addition, one could want to hide the parameter X used in the encapsulation C even if the adversary $\mathcal{A}$ can ask any key usk_i for Y_i such that $\mathcal{R}(X_0, Y_i) = \mathcal{R}(X_1, Y_i) = 0$ for all i, using oracle $\mathcal{O}\mathsf{KeyGen}(Y_i)$ that stores Y_i in the set $\mathcal{Y}$ and outputs $\mathsf{KEMAC.KeyGen}(\mathsf{MSK}, Y_i)$. A KEMAC is said to be $\mathsf{AC\text{-}IND}$-secure if for any adversary $\mathcal{A}$, that can ask any key usk_i, using oracle $\mathcal{O}\mathsf{KeyGen}(Y_i)$ that stores Y_i in the set $\mathcal{Y}$ and outputs $\mathsf{KEMAC.KeyGen}(\mathsf{MSK}, Y_i)$, $\mathsf{Adv}^{\mathsf{ac\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}) = \mathsf{negl}(\kappa)$, for $b \overset{\$}{\leftarrow} \{0,1\}$ and

$$\begin{aligned}\mathcal{D}_b &= \left\{\begin{array}{l}(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{KEMAC.Setup}(1^\kappa),\\ (\mathsf{state},X_0,X_1) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(\cdot)}(\mathsf{PK}),\\ (C_i,K_i) \leftarrow \mathsf{KEMAC.Enc}(\mathsf{PK},X_i), \text{ for } i=0,1\end{array} : (\mathsf{state},C_b)\right\}\\ \mathsf{BadXY} &= [\exists Y_i \in \mathcal{Y}, \mathcal{R}(X_0,Y_i)=1 \vee \mathcal{R}(X_1,Y_i)=1]\\ \mathsf{Adv}^{\mathsf{ac\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}) &= 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(\cdot)}(\mathsf{state},C_b) = b \,|\, \neg\mathsf{BadXY}] - 1,\end{aligned}$$

where we again condition the advantage to legitimate attacks only.

Traceability. In any multi-user setting, to avoid abuse of the decryption keys, one may want to be able to trace a user (or their personal key) from the decryption mechanism, and more generally from any *useful* decoder, either given access to the key material in the device (white-box tracing) or just interacting with the device (black-box tracing). Without any keys, one expects session-key privacy, but as soon as one knows a key, one can distinguish the session-key. Then, we will call a *useful* pirate decoder $\mathcal{P}$ a good distinguisher against session-key privacy, that behaves differently with the real and a random key. But of course, this pirate decoder can be built from multiple user' keys, called traitors, and one would like to be able to trace at least one of them.

A weaker variant of traceability is just a confirmation of candidate traitors, and we will target this goal: if a pirate decoder $\mathcal{P}$ has been generated from a list $\mathcal{T} = \{Y_i\}$ of traitors' keys, a confirmer algorithm $\mathcal{C}$ can output, from a valid guess $\mathcal{G}$ for $\mathcal{T}$, at least one traitor in $\mathcal{T}$. More formally, let us consider any adversary $\mathcal{A}$ that can ask for key generation through oracle $\mathcal{O}\mathsf{KeyGen}(Y_i)$, that gets $\mathsf{usk}_i \leftarrow \mathsf{KEMAC.KeyGen}(\mathsf{MSK}, Y_i)$, outputs nothing but appends the new user Y_i in $\mathcal{U}$, and then corrupt some users through the corruption oracle $\mathcal{O}\mathsf{Corrupt}(Y_i)$, that outputs usk_i and appends Y_i in $\mathcal{T}$, to build a *useful* pirate decoder $\mathcal{P}$, then there is a *correct* confirmer algorithm $\mathcal{C}$ that outputs a traitor T, with *negligible error*: for $b \overset{\$}{\leftarrow} \{0,1\}$ and

$$\mathcal{D} = \left\{ \begin{array}{l} (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{KEMAC.Setup}(1^\kappa), \mathcal{P} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(\cdot), \mathcal{O}\mathsf{Corrupt}(\cdot)}(\mathsf{PK}), \\ X \text{ such that } \forall Y_i \in \mathcal{T}, \mathcal{R}(X, Y_i) = 1, \\ (C, K_0) \leftarrow \mathsf{KEMAC.Enc}(\mathsf{PK}, X), K_1 \overset{\$}{\leftarrow} \mathcal{K} : \\ \qquad\qquad\qquad\qquad (\mathsf{MSK}, \mathcal{P}, \mathcal{U}, \mathcal{T}, C, K_0, K_1) \end{array} \right\},$$

we denote

- $\mathcal{P}$ to be useful, if $2 \times \Pr_{\mathcal{D},b}[\mathcal{P}(C, K_b) = b] - 1$ is non-negligible;
- $\mathcal{C}$ to be correct, if $\Pr_{\mathcal{D}}[T \in \mathcal{T} \,|\, T \leftarrow \mathcal{C}^{\mathcal{P}(\cdot,\cdot)}(\mathsf{MSK}, \mathcal{T})]$ is overwhelming;
- $\mathcal{C}$ is error-free if for any $\mathcal{G} \subset \mathcal{U}$, $\Pr_{\mathcal{D}}[T \notin \mathcal{T} \,|\, T \leftarrow \mathcal{C}^{\mathcal{P}(\cdot,\cdot)}(\mathsf{MSK}, \mathcal{G}) \wedge T \neq \perp]$ is negligible.

More concretely, we say that the decoder $\mathcal{P}$ is *useful* if it can distinguish the real key from a random key with significant advantage. Then, from such a useful decoder, the confirmer $\mathcal{C}$ is *correct* if it outputs a traitor with overwhelming probability, when it starts from the correct set $\mathcal{T}$ of candidates. Eventually, it should be *error-free*: $\mathcal{T}$ does not output an honest user, but with negligible probability. The t-confirmation limits the number of corrupted users in $\mathcal{T}$ to t.

3 Authenticated Key Encapsulation Mechanism

With public-key privacy, one cannot know who is the actual receiver, and needs to check the decapsulated session key with an authenticated encryption scheme to know whether they were a recipient or not. The latter check can be time-consuming when applied on a large data content (or when there are multiple decryption keys to try). We can hope to have quick key confirmation, if the additional *Authentication* (AUTH) property is satisfied.

Authentication. A KEM provides *authentication* (denoted AUTH) if it satisfies $\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{KEM}}(\kappa) = 1 - \Pr_{\mathcal{D}}[\mathsf{Ev}] = \mathsf{negl}(\kappa)$, for

$$\mathcal{D} = \left\{ \begin{array}{l} \forall i \in \{0,1\}, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa), \\ (C, K) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}_0) : (\mathsf{sk}_1, C) \end{array} \right\}$$

$$\mathsf{Ev} = [\mathsf{KEM.Dec}(\mathsf{sk}_1, C) = \perp].$$

We present a generic conversion to add the AUTH property to any KEM, while retaining previous properties (SK-IND and PK-IND). To this aim, we use a hash function $\mathcal{H}$. In the security analysis, it will be modeled by a random oracle that outputs a new random and independent bitstring for any new query.

3.1 Key Encapsulation Mechanism with Authentication

We present below (and in the Appendix A in Figure 1) a KEM' with authentication from a KEM, with two security parameters: k, the length of the new encapsulated key, and ℓ, the length of the verification tag. We also utilize a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{k+\ell}$ that outputs digests on $k+\ell$ bits:

– $\mathsf{KEM}'.\mathsf{KeyGen}(1^\kappa)$ runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM}.\mathsf{KeyGen}(1^\kappa)$;
– $\mathsf{KEM}'.\mathsf{Enc}(\mathsf{pk})$ runs $(c, s) \leftarrow \mathsf{KEM}.\mathsf{Enc}(\mathsf{pk})$ and gets $U\|V \leftarrow \mathcal{H}(s)$. One then outputs $C \leftarrow (c, V)$ together with the encapsulated key $K \leftarrow U$;
– $\mathsf{KEM}'.\mathsf{Dec}(\mathsf{sk}, C = (c, V))$ runs $s \leftarrow \mathsf{KEM}.\mathsf{Dec}(\mathsf{sk}, c)$, gets $U'\|V' \leftarrow \mathcal{H}(s)$, and checks whether $V = V'$. In the positive case, one outputs $K' \leftarrow U'$, otherwise one outputs $\perp$.

Correctness. If the KEM KEM is correct, then the derived KEM' with authentication is also correct, has the decapsulation of c outputs the same s as during encapsulation, and then $\mathcal{H}(s)$ gives the same key and tag.

We will now show the previous security notions still hold, and we really provide authentication.

3.2 Security Properties

We can claim that the above KEM' retains the initial security properties of the KEM scheme, but as the proofs essentially rely of the random oracle property, we deferred the proofs to the Appendix B

Theorem 1 (Session-Key Privacy). *If the KEM KEM is SK-IND and PK-IND-secure, and $\mathcal{H}$ is a hash function modelled as a random oracle, then its derived KEM' with authentication is SK-IND-secure: $\mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}'}(\tau) \leq 2 \cdot \mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\tau) + 2 \cdot q_h/\#\mathcal{K}$, where q_h is the number of $\mathcal{H}$ queries and $\#\mathcal{K}$ the cardinality of the session-key set $\mathcal{K}$ of KEM.*

Theorem 2 (Public-Key Privacy). *If the KEM KEM is both SK-IND and PK-IND-secure, and $\mathcal{H}$ is a hash function modelled as a random oracle, then KEM' is PK-IND-secure: $\mathsf{Adv}^{\textsf{pk-ind}}_{\mathsf{KEM}'}(\mathcal{A}) \leq \mathsf{Adv}^{\textsf{pk-ind}}_{\mathsf{KEM}}(\tau) + 4 \times \mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\tau) + 4q_h/\#\mathcal{K}$.*

However, we develop the authentication property:

Theorem 3 (Authentication). *If the KEM KEM is SK-IND, the KEM KEM' provides authentication in the random oracle model: $\mathsf{Adv}^{\textsf{auth}}_{\mathsf{KEM}'}(\kappa) \leq \mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\tau_{\mathsf{Dec}}) + 1/\#\mathcal{K} + 2^{-\ell}$, where τ_{Dec} is the time of a decapsulation under KEM.*

Proof. For a truly random hash function (in the ROM), different inputs have random and independent outputs, so one can get $V_0 = V_1$ for inputs $s_0 \neq s_1$, but only with probability $2^{-\ell}$.

Let us come back to any KEM KEM, and the event $\mathsf{Ev} = (K_0 = K_1)$ in the distribution

$$\mathcal{D} = \left\{ \begin{array}{l} \forall i \in \{0,1\}, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM}.\mathsf{KeyGen}(1^\kappa), \\ (C, K_0) \leftarrow \mathsf{KEM}.\mathsf{Enc}(\mathsf{pk}_0), K_1 \leftarrow \mathsf{KEM}.\mathsf{Dec}(\mathsf{sk}_1, C) : (K_0, K_1) \end{array} \right\}$$

Let us now consider the following adversary against SK-IND: the challenger runs $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KEM}.\mathsf{KeyGen}(1^\kappa)$, $(C, K_0) \leftarrow \mathsf{KEM}.\mathsf{Enc}(\mathsf{pk}_0)$, and $K_1 \overset{\$}{\leftarrow} \mathcal{K}$, and then returns (pk_0, K_b), for a random bit $b \overset{\$}{\leftarrow} \{0,1\}$; upon receiving (pk, K), the adversary runs $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KEM}.\mathsf{KeyGen}(1^\kappa)$, $K' \leftarrow \mathsf{KEM}.\mathsf{Dec}(\mathsf{pk}_1, C)$, and outputs $b' = 0$ if and only if $K' = K$, and $b' = 1$ otherwise:

– if $b = 1$, the probability to output 0 is pure chance, and thus $1/\#\mathcal{K}$;
– if $b = 0$, the probability to output 0 is exactly the probability of event Ev.

As $\mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\mathcal{A}) = \Pr[b' = 0 \mid b = 0] - \Pr[b' = 0 \mid b = 1]$, we have it to be equal to $\Pr[\mathsf{Ev}] - 1/\#\mathcal{K} \leq \mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\tau_{\mathsf{Dec}})$, as the adversary only computes a decapsulation for its decision. Hence, we can upper-bound $\Pr[\mathsf{Ev}] \leq \mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\tau_{\mathsf{Dec}}) + 1/\#\mathcal{K}$.

As a consequence, $\mathsf{Adv}^{\textsf{auth}}_{\mathsf{KEM}'}(\kappa) \leq \mathsf{Adv}^{\textsf{sk-ind}}_{\mathsf{KEM}}(\tau_{\mathsf{Dec}}) + 1/\#\mathcal{K} + 2^{-\ell}$. □

4 Subset-Cover KEMAC

The above notion of access control is quite general and includes both key-policy ABE and ciphertext-policy ABE, where one can have policies $\mathcal{P}$ and attributes such that given a subset of attributes, this defines a list of Boolean B (according to the presence or not of the attribute), and $\mathcal{P}(B)$ is either true or false.

4.1 Basic (Without Anonymity) Subset-Cover KEMAC

For efficiency consideration, we will focus on the subset-cover approach: during the Setup, one defines multiple sets S_i; when generating a user key usk_j, a list A_j of subsets if specified, which implicitly means user $U_j \in S_i$ for all $i \in A_j$; at encapsulation time, a target set T is given by B, such that $T = \cup_{i \in B} S_i$.

Intuitively, S_i's are subsets of the universe of users, and to specify the receivers, one encapsulates the key K for a covering of the target set T. A KEMAC, for a list Σ of sets S_i, can then be defined from any KEM in $\mathcal{K}$ that is a group with internal law denoted $\oplus$:

– $\mathsf{KEMAC.Setup}(\Sigma)$, for each $S_i \in \Sigma$, runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa)$. Then $\mathsf{PK} \leftarrow (\mathsf{pk}_i)_i$ and $\mathsf{MSK} \leftarrow (\mathsf{sk}_i)_i$;
– $\mathsf{KEMAC.KeyGen}(\mathsf{MSK}, A_j)$ defines the user's secret key $\mathsf{usk}_j \leftarrow (i, \mathsf{sk}_i)_{i \in A_j}$;
– $\mathsf{KEMAC.Enc}(\mathsf{PK}, B)$ generates a random session key $K \leftarrow \mathcal{K}$, and runs $(C_i, K_i) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}_i)$ for all $i \in B$, and outputs $C \leftarrow (i, C_i, E_i = K \oplus K_i)_{i \in B}$ together with the encapsulated key K;
– $\mathsf{KEMAC.Dec}(\mathsf{usk}_j, C)$ looks for $i \in \mathsf{usk}_j \cap C$, to run $K_i' \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}_i, C_i)$ and output $K \leftarrow K_i' \oplus E_i$.

In terms of attributes, one can consider that each S_i is associated to an attribute a_i, and being in S_i for a user U_j means owning the attribute a_i. At encapsulation time, B lists the attributes that allow to decrypt: as soon as a_i is in B, any user U_j owning a_i can decrypt.

For the above scheme, we can claim the $\mathsf{SK\text{-}IND}$ security, but unfortunately not the $\mathsf{AC\text{-}IND}$ security. As attributes are known, the correctness of the KEM implies the correctness of the KEMAC.

Theorem 4 (Session-Key Privacy). *If the underlying KEM is $\mathsf{SK\text{-}IND}$-secure, the above basic subset-cover KEMAC is $\mathsf{SK\text{-}IND}$-secure, for selective key-queries: $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEMAC}}(\tau) \leq 2q_k \times \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau)$, where q_k is the number of key-queries.*

As this proof uses a classical hybrid technique to replace each keys by random keys, we defer the proof to the Appendix B.3. One notes that we need B to be provided in the ciphertext: indices i are given. We definitely exclude access-control privacy. In order to get anonymity, theses indices should not be given.

4.2 Anonymous Subset-Cover KEMAC with Early Aborts

To avoid sending B together with the ciphertext, but still being able to quickly find the correct matching indices in the ciphertext and the user's key, one can use a KEM' with authentication:

– $\mathsf{KEMAC.Setup}(\Sigma)$, for each $S_i \in \Sigma$, runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM}'.\mathsf{KeyGen}(1^\kappa)$: $\mathsf{PK} \leftarrow (\mathsf{pk}_i)_i$ and $\mathsf{MSK} \leftarrow (\mathsf{sk}_i)_i$;
– $\mathsf{KEMAC.KeyGen}(\mathsf{MSK}, A_j)$ defines the user's secret key $\mathsf{usk}_j \leftarrow (\mathsf{sk}_i)_{i \in A_j}$;
– $\mathsf{KEMAC.Enc}(\mathsf{PK}, B)$ generates a random session key $K \overset{\$}{\leftarrow} \{0,1\}^k$, and, for all $i \in B$, runs $(C_i, K_i) \leftarrow \mathsf{KEM}'.\mathsf{Enc}(\mathsf{pk}_i)$ and outputs $C \leftarrow (C_i, E_i = K \oplus K_i)_{i \in B}$ together with the encapsulated key K;
– $\mathsf{KEMAC.Dec}(\mathsf{usk}, C)$, for all sk_i in usk and all (C_j, E_j) in C, runs $K_{i,j}' \leftarrow \mathsf{KEM}'.\mathsf{Dec}(\mathsf{sk}_i, C_j)$. It stops for the first valid $K_{i,j}'$, outputs $K \leftarrow K_{i,j}' \oplus E_j$.

For this above scheme, we can claim both the SK-IND security and the AC-IND security, for selective key queries. But first, let us check the correctness:

Theorem 5 (Correctness). *If the underlying KEM′ is AUTH-secure, the above subset-cover KEMAC is correct:* $\mathsf{Adv}^{cor}_{KEMAC}(\kappa) \leq S_A S_B \times \mathsf{Adv}^{auth}_{KEM'}(\kappa)$*, where* S_A *and* S_B *are the sizes of the user' sets of attributes and the number of subsets in the ciphertext, respectively.*

Proof. To make the correctness fail, a wrong key, among the $S_A S_B$ possibilities, should make accept, which is bounded by $\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{KEM}'}(\kappa)$ for each wrong pair.

About SK-IND and AC-IND security, the former proof is exactly the same as for the above scheme, and the latter again uses a classical hybrid technique. The proof is thus deferred to the Appendix B.4.

Theorem 6 (Session-Key Privacy). *If the underlying KEM′ is SK-IND-secure, the above subset-cover KEMAC is also SK-IND-secure, for selective key-queries:* $\mathsf{Adv}^{sk\text{-}ind}_{KEMAC}(\tau) \leq 2q_k \times \mathsf{Adv}^{sk\text{-}ind}_{KEM'}(\tau)$*, where* q_k *is the number of key-queries.*

Theorem 7 (Access-Control Privacy). *If the underlying KEM′ is AC-IND-secure, the above subset-cover KEMAC is AC-IND-secure, for selective key-queries and constant-size sets* B*:* $\mathsf{Adv}^{ac\text{-}ind}_{KEMAC}(\tau) \leq 2S_B \times \mathsf{Adv}^{pk\text{-}ind}_{KEM}(\tau)$*, where* S_B *is the constant-size of the sets* B*.*

We just stress that B must have a constant size to achieve access-control privacy.

5 Hybrid KEM

While one can never exclude an attack against a cryptographic scheme, combining several independent approaches reduces the risks. This is the way one suggests to apply post-quantum schemes, in combination with classical schemes, in order to be sure to get the best security.

5.1 Hybrid KEM Construction

Let us first study the combination of two KEMs (KEM_1 and KEM_2), so that as soon as one of them achieves SK-IND security, the hybrid KEM achieves SK-IND security too.

We need both KEMs to generate keys in $\mathcal{K}$, with a group structure and internal law denoted $\oplus$. One can also find it in the Appendix A, Figure 2:

- $\mathsf{KEM.KeyGen}(1^\kappa)$ calls $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM}_i.\mathsf{KeyGen}(1^\kappa)$, for $i \in \{1,2\}$ and outputs $\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$ and $\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$;
- $\mathsf{KEM.Enc}(\mathsf{pk})$ parses pk as $(\mathsf{pk}_1, \mathsf{pk}_2)$, calls $(C_i, K_i) \leftarrow \mathsf{KEM}_i.\mathsf{Enc}(\mathsf{pk}_i)$ for $i \in \{1,2\}$, and outputs $(C = (C_1, C_2), K = K_1 \oplus K_2)$;
- $\mathsf{KEM.Dec}(\mathsf{sk}, C)$ parses sk as $(\mathsf{sk}_1, \mathsf{sk}_2)$ and C as (C_1, C_2), then calls both $K_i \leftarrow \mathsf{KEM}_i.\mathsf{Dec}(\mathsf{sk}_i, C_i)$, and outputs $K = K_1 \oplus K_2$.

5.2 Security Properties

As expected, we can prove that as soon as one of them achieves SK-IND security, the hybrid KEM achieves SK-IND security too. However, for PK-IND security of KEM, we need both the underlying schemes to be PK-IND secure.

Theorem 8 (Session-Key Privacy). *If at least one of the underlying* KEM_1 *and* KEM_2 *is SK-IND-secure, the hybrid KEM is SK-IND-secure.*

$$\mathsf{Adv}^{sk\text{-}ind}_{KEM}(\tau) \leq \min\{\mathsf{Adv}^{sk\text{-}ind}_{KEM_1}(\tau), \mathsf{Adv}^{sk\text{-}ind}_{KEM_2}(\tau)\}.$$

Proof. We present the sequence of games, first exploiting the session-key privacy of KEM_1.

Game $\mathbf{G}_0$: In the initial game, the adversary receives $\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$, both keys having been generated by the respective key generation algorithms, together with $C = (C_1, C_2)$ and K that is either $K_1 \oplus K_2$ (each encapsulated in C_1 and C_2) or a random key from $\mathcal{K}$, according to the random bit b. The adversary outputs its guess b'. We denote P_0 the probability of event $b' = b$, which is $(1 + \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}))/2$.

Game $\mathbf{G}_1$: In this game, we replace K_1's by $K_1 \overset{\$}{\leftarrow} \mathcal{K}$ in the generation of $K_1 \oplus K_2$: $P_0 - P_1 \leq \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}_1}(t)$, where t is the maximum running-time of adversary $\mathcal{A}$. In this final game, this is clear that $P_1 = 1/2$, as K is truly random in $\mathcal{K}$ in both cases.

Hence, $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}_1}(t)$. In the same way, one can prove $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}_2}(t)$, Hence $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) \leq \min\{\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}_1}(\tau), \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}_2}(\tau)\}$. □

Theorem 9 (Public-Key Privacy). *If both underlying KEM_1 and KEM_2 are PK-IND-secure, the hybrid KEM is PK-IND-secure:*

$$\mathsf{Adv}^{\mathit{pk\text{-}ind}}_{\mathit{KEM}}(\tau) \leq \mathsf{Adv}^{\mathit{pk\text{-}ind}}_{\mathit{KEM}_1}(\tau) + \mathsf{Adv}^{\mathit{pk\text{-}ind}}_{\mathit{KEM}_2}(\tau).$$

Proof. The proof is quite similar to the previous one, but with the security of both schemes:

Game $\mathbf{G}_0$: In the initial game, the adversary receives $\mathsf{pk}^{(0)} \leftarrow (\mathsf{pk}_1^{(0)}, \mathsf{pk}_2^{(0)})$ and $\mathsf{pk}^{(1)} \leftarrow (\mathsf{pk}_1^{(1)}, \mathsf{pk}_2^{(1)})$, with keys having been generated by the respective key generation algorithms, together with $C = (C_1, C_2)$ and $K = K_1 \oplus K_2$ where $(C_i, K_i) \leftarrow \mathsf{KEM}_i.\mathsf{Enc}(\mathsf{pk}_i^{(b)})$, according to the random bit b. The adversary outputs its guess b'. We denote P_0 the probability of event $b' = b$, which is $(1 + \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}))/2$.

Game $\mathbf{G}_1$: In this game, we replace (C_1, K_1) by $(C_1, K_1) \leftarrow \mathsf{KEM}_1.\mathsf{Enc}(\mathsf{pk}_1^{(0)})$ in the generation of $C = (C_1, C_2)$ and $K = K_1 \oplus K_2$: $P_0 - P_1 \leq \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}_1}(t)$, where t is the maximum running-time of adversary $\mathcal{A}$.

Game $\mathbf{G}_2$: In this game, we replace (C_2, K_2) by $(C_2, K_2) \leftarrow \mathsf{KEM}_2.\mathsf{Enc}(\mathsf{pk}_2^{(0)})$ in the generation of $C = (C_1, C_2)$ and $K = K_1 \oplus K_2$: $P_1 - P_2 \leq \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}_2}(t)$. In this final game, this is clear that $P_2 = 1/2$, as (C, K) is independent of b.

Hence, we can claim $\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}_1}(\tau) + \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}_2}(\tau)$. □

6 Traceable KEM

In a subset-cover-based KEMAC, a same decapsulation key sk_i is given to multiple users, for a public key pk_i. In case of abuse, one cannot know who is the defrauder. We offer an ElGamal-based KEM that allows some kind of traceability, in the same vein as [BF99].

6.1 Traceable ElGamal-based TKEM

Let $\mathbb{G}$ be a group of prime order q, with a generator g, in which the Computational Diffie-Hellman problem is hard. We describe below a TKEM with n multiple decapsulation keys for a specific public key, allowing to deal with collusions of at most t users:

- $\mathsf{TKEM.KeyGen}(1^\kappa, n, t)$: generates a public key pk and n different secret keys usk_j:
  - it samples random $s, s_k \overset{\$}{\leftarrow} \mathbb{Z}_q$, for $k = 1 \ldots, t+1$ and sets $h \leftarrow g^s$ as well as $h_k \leftarrow g^{s_k}$ for each k;
  - for users U_j, for $j = 1 \ldots, n$, one samples random $(v_{j,k})_k \overset{\$}{\leftarrow} \mathbb{Z}_q^{t+1}$, such that $\sum_k v_{j,k} s_k = s$, for $j = 1 \ldots, n$. Then, $\mathsf{pk} \leftarrow ((h_k)_k, h)$, while each $\mathsf{usk}_j \leftarrow (v_{j,k})_k$.

- $\mathsf{TKEM.Enc}(\mathsf{pk} = ((h_k)_k, h))$: it samples a random $r \xleftarrow{\$} \mathbb{Z}_q$, and sets $C = (C_k \leftarrow h_k^r)_k$, as well as $K \leftarrow h^r$.
- $\mathsf{TKEM.Dec}(\mathsf{usk}_j = (v_{j,k})_k, C = (C_k)_k)$: it outputs $K \leftarrow \prod_k C_k^{v_{j,k}}$

One can note that

$$\prod_k C_k^{v_{j,k}} = \prod_k h_k^{rv_{j,k}} = \prod_k (g^r)^{s_k v_{j,k}} = g^{r \sum_k s_k v_{j,k}} = g^{sr} = h^r = K.$$

6.2 Security Properties

First, we will show that the above TKEM construction achieves both $\mathsf{SK\text{-}IND}$ and $\mathsf{PK\text{-}IND}$ security. But it also allows to confirm traitors, from a stateless pirate decoder $\mathcal{P}$ (in particular, this means that $\mathcal{P}$ never blocks itself after several invalid ciphertexts).

Theorem 10 (Session-Key Privacy). *The above TKEM achieves SK-IND security under the DDH assumption in $\mathbb{G}$: $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{TKEM}}(\tau) \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\tau)$.*

Proof. From a Diffie-Hellman tuple (A, B, C), one can derive, for random $s_k \xleftarrow{\$} \mathbb{Z}_q$, for $k = 1 \ldots, t+1$

$$h_k \leftarrow g^{s_k} \qquad h \leftarrow B \qquad C_k \leftarrow A^{s_k} \qquad K \leftarrow C$$

where s, r are implicitly defined as $A = g^r$ and $B = g^s$. If C is indeed the Diffie-Hellman value for (g, A, B), then $K = C = g^{sr} = h^r$, we are in the real case ($b = 0$). If C is a random value, we are in the random case ($b = 1$):

$$\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{TKEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\tau).$$

$\square$

Theorem 11 (Public-Key Privacy). *The above TKEM achieves PK-IND security under the DDH assumption in $\mathbb{G}$: $\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{TKEM}}(\tau) \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\tau)$.*

Proof. From a Diffie-Hellman tuple (A, B, C), one can derive, for random scalars $z_k^{(0)}, z_k^{(1)}, s_k^{(0)}, s_k^{(1)}, z^{(0)}, z^{(1)}, s^{(0)}$
$\mathbb{Z}_q$, for $k = 1 \ldots, t+1$

$$\begin{aligned} h_k^{(0)} &\leftarrow A^{z_k^{(0)}} \cdot g^{s_k^{(0)}} & h^{(0)} &\leftarrow A^{z^{(0)}} \cdot g^{s^{(0)}} \\ h_k^{(1)} &\leftarrow A^{z_k^{(1)}} \cdot g^{s_k^{(1)}} & h^{(1)} &\leftarrow A^{z^{(1)}} \cdot g^{s^{(1)}} \\ C_k &\leftarrow C^{z_k^{(b)}} \cdot B^{s_k^{(b)}} \end{aligned}$$

where r is implicitly defined as $B = g^r$. If C is indeed the Diffie-Hellman value for (g, A, B), then $C_k = A^{rz_k^{(b)}} \cdot g^{rs_k^{(b)}} = (A^{z_k^{(b)}} \cdot g^{s_k^{(b)}})^r = (h_k^{(b)})^r$. If C is a random value $C = A^{r+c}$, for a random $c \xleftarrow{\$} \mathbb{Z}_q$:

$$C_k = A^{(r+c)z_k^{(b)}} \cdot g^{rs_k^{(b)}} = (A^{z_k^{(b)}} \cdot g^{s_k^{(b)}})^r \cdot A^{cz_k^{(b)}} = (h_k^{(b)})^r \cdot A^{cz_k^{(b)}}.$$

As $z_k^{(b)}$ is perfectly hidden in the public key, C_k follows a uniform distribution in $\mathbb{G}$, independently of the public key, and thus of b: $\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{TKEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\tau)$. $\square$

Theorem 12 (t-Confirmation). *A collusion of at most t keys can be confirmed from a useful stateless pirate decoder $\mathcal{P}$: starting from a correct guess for $\mathcal{T}$, the traitors' keys used for building the pirate decoder $\mathcal{P}$, by accessing the decoder, one can confirm a traitor in $\mathcal{T}$, with negligible error.*

Proof. To prove this theorem, we first give a description of the confirmer algorithm $\mathcal{C}$, then we provide the indistinguishability analysis, and eventually prove $\mathcal{C}$ will give a correct answer.

Description of the Confirmer $\mathcal{C}$: The confirmer algorithm $\mathcal{C}$ can proceed as follows, for a candidate subset $\mathcal{G}$: $\{\mathsf{usk}_j = (v_{j,k})_k\}_{j\in\mathcal{G}}$, for $\mathcal{G}$ of size at most t: it chooses $(u_k)_k$ orthogonal to the subvector-space spanned by $\{(v_{j,k})_k\}_{j\in\mathcal{G}}$, which means that

$$\sum_k u_k v_{j,k} = 0, \forall j \in \mathcal{G}.$$

This is possible as $(v_{j,k})_{k\in[1,t+1],j\in\mathcal{G}}$ is of rank at most t in $\mathbb{Z}_q^{t+1}$. Then the kernel is of dimension at least 1. One generates a fake ciphertext $C = (C_k)_k$, with $C_k \leftarrow h_k^r \cdot g^{u_k s'}$, for random $r, s' \overset{\$}{\leftarrow} \mathbb{Z}_q$, and then $K \leftarrow h^r$:

– Any key usk_j in $\mathcal{G}$ will lead to

$$\prod_k C_k^{v_{j,k}} = \prod_k g^{(rs_k+s'u_k)\cdot v_{j,k}} = g^{r\sum_k s_k v_{j,k}+s'\sum_k u_k v_{j,k}} = g^{rs+s'\times 0} = g^{rs} = K.$$

– Similarly, any key usk_j outside $\mathcal{G}$ will lead to

$$\prod_k C_k^{v_{j,k}} = K \times (g^{\sum_k u_k v_{j,k}})^{s'} \neq K.$$

we will show this allows to confirm at least one traitor from a candidat subset of traitors.

Indistinguishability Analysis. The above remark about the output key from a pirate decoder $\mathcal{P}$ assumes an honest behavior, whereas it can stop answering if it detects the fake ciphertext. We first need to show that, with the public key $\mathsf{pk} = ((h_k)_k, h)$ and only $\{\mathsf{usk}_j = (v_{j,k})_k\}_{j\in\mathcal{G}}$, one cannot distinguish the fake ciphertext from a real ciphertext, generated as above: from a Diffie-Hellman tuple $(A = g^a, B = g^r, C)$, one can derive, from random scalars $s, s'_k, u_k \overset{\$}{\leftarrow} \mathbb{Z}_q$, such that $\sum_k v_{j,k}s'_k = s$ and $\sum_k v_{j,k}u_k = 0$, for $j = 1 \ldots, n$:

$$h_k \leftarrow A^{u_k} \cdot g^{s'_k} = g^{au_k+s'_k} \qquad h \leftarrow g^s \qquad \mathsf{usk}_j = (v_{j,k})_k \text{ for } j \in \mathcal{G}$$

where we implicitly define $s_k \leftarrow au_k + s'_k$, that satisfy

$$\sum_k v_{j,k}s_k = \sum_k v_{j,k}(s'_k + au_k) = \sum_k v_{j,k}s'_k + a\sum_k v_{j,k}u_k = s + 0 = s.$$

Then, one defines $C_k \leftarrow C^{u_k} \cdot B^{s'_k}$ and $K \leftarrow B^s$.

Let us note $C = g^{r-c}$, where c is either 0 (a Diffie-Hellman tuple) or random:

$$C_k = A^{(r+c)u_k} \cdot g^{rs'_k} = (A^{u_k} \cdot g^{s'_k})^r \cdot A^{cu_k} = h_k^r \cdot (A^c)^{u_k}.$$

One can remark that: when $c = 0$ (Diffie-Hellman tuple), $C = (C_k)_k$ is a normal ciphertext; when $c = s'$ (random tuple), this is a fake ciphertext. Under the DDH assumption, they are thus indistinguishable for an adversary knowing the keys $(\mathsf{usk}_i)_{i\in\mathcal{G}}$.

Confirmation of a Traitor. The above analysis shows that a pirate decoder $\mathcal{P}$ built from $(\mathsf{usk}_i)_{i\in\mathcal{G}}$ cannot distinguish the fake ciphertext from a real ciphertext. A useful pirate decoder should necessarily distinguish real key from random key. Then, several situations may appear, according to the actual set $\mathcal{T}$ of traitors' keys used to build the pirate decoder $\mathcal{P}$ by the adversary $\mathcal{A}$:

– If $\mathcal{T} \subseteq \mathcal{G}$, a useful decoder $\mathcal{P}$ can distinguish keys;
– If $\mathcal{T} \cap \mathcal{G} = \emptyset$, $\mathcal{P}$ cannot distinguish keys, as it can get several candidates, independent from the real or random keys.

Let us now assume we started from $\mathcal{G} \supseteq \mathcal{T}$, then the advantage of $\mathcal{P}$ in distinguishing real and random keys, denoted $p_{\mathcal{G}}$, is non-negligible, from the usefulness of the decoder. The following steps would also work if one starts with $\mathcal{G} \cap \mathcal{T} \neq \emptyset$, so that the advantage $p_{\mathcal{G}}$ is significant.

One then removes a user J from $\mathcal{G}$ to generate $\mathcal{G}'$ and new ciphertexts to evaluate $p_{\mathcal{G}'}$: if $J \notin \mathcal{T}$, usk_J is not known to the adversary, and so there is no way to check whether $\sum_k v_{J,k} s'_k = s$ and $\sum_k v_{J,k} u_k = 0$, even for a powerful adversary. So necessarily, $p_{\mathcal{G}'} = p_{\mathcal{G}}$.

On the other hand, we know that $p_{\emptyset} = 0$. So, one can sequentially remove users until a significant gap appears: this is necessarily for a user in $\mathcal{T}$. $\square$

Corrolary 1 *In the particular case of $t = 1$, one can efficiently trace one traitor, from a useful stateless pirate decoder: by trying $\mathcal{G} = \{J\}$ sequentially for each $J = 1, \ldots, n$, and evaluating $p_{\mathcal{G}}$, one should get either a significant advantage (for the traitor) or 0 (for honest keys).*

7 Our KEMAC Scheme

We have already presented a traceable KEM that is secure against classical adversaries. If we combine it with another scheme expected secure against quantum adversaries, we can thereafter combine them into an hybrid-KEM, that inherits security properties from both schemes, with still traceability against classical adversaries. But we will actually exploit the properties of a Public-Key Encryption (PKE) scheme in order to improve efficiency of the combination.

7.1 Public-Key Encryption

A Public-Key Encryption (PKE) is defined by 3 algorithms:

- $\mathsf{PKE.KeyGen}(1^\kappa)$: the *key generation algorithm* outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$;
- $\mathsf{PKE.Enc}(\mathsf{pk}, m)$: the *encryption algorithm* encrypts the input message m under the public key pk and outputs the ciphertext C;
- $\mathsf{PKE.Dec}(\mathsf{sk}, C)$: the *decryption algorithm* outputs the message m encrypted in C.

We will use the classical notion of indistinguishability and of anonymity of such a PKE, which are informally recalled below, and more formally described in the Appendix C:

- Indistinguishability. For an honestly generated pk, if the adversary chooses two messages m_0 and m_1, it cannot distinguish an encryption of m_0 from an encryption of m_1, both under pk.
- Anonymity. For two honestly generated pk_0 and pk_1, if the adversary chooses a message m, it cannot distinguish an encryption of m under pk_0 from the encryption of m under pk_1.

Given such a PKE, that is both indistinguishable and anonymous, we can trivially get a KEM that is both SK-IND and PK-IND secure:

- $\mathsf{KEM.KeyGen}(1^\kappa)$ gets $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\kappa)$, and outputs $(\mathsf{pk}, \mathsf{sk})$;
- $\mathsf{KEM.Enc}(\mathsf{pk})$ generates $K \overset{\$}{\leftarrow} \mathcal{K}$, gets $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, K)$, and outputs (K, C);
- $\mathsf{KEM.Dec}(\mathsf{sk}, C)$ outputs $\mathsf{PKE.Dec}(\mathsf{sk}, C)$.

7.2 CRYSTALS-Kyber PKE

We recall the algorithms of the CRYSTALS-Kyber [ABD+21] public-key encryption whose both indistinguishability and anonymity rely on the hardness of Module-LWE [LS15]. We identify R_q with $\mathbb{Z}_q^n$ that contains the plaintext space $\mathcal{K} = \{0,1\}^n$, and use two noise parameters $\eta_1 \geq \eta_2$, for the Gaussian distributions $\mathcal{B}_{\eta_1}$ and $\mathcal{B}_{\eta_2}$:

- $\mathsf{Kyber.KeyGen}(1^\kappa)$: sample random $\mathbf{A} \overset{\$}{\leftarrow} \mathsf{R}_q^{k\times k}$ and $(\mathbf{s},\mathbf{e}) \overset{\$}{\leftarrow} \mathcal{B}_{\eta_1}^k \times \mathcal{B}_{\eta_1}^k$, then set $\mathsf{pk} \leftarrow (\mathbf{A}, \mathbf{b} = \mathbf{As}+\mathbf{e})$ and $\mathsf{sk} \leftarrow \mathbf{s}$.
- $\mathsf{Kyber.Enc}(\mathsf{pk}, K)$: $\mathbf{r} \overset{\$}{\leftarrow} \mathcal{B}_{\eta_1}^k$, and $(\mathbf{e}_1, e_2) \overset{\$}{\leftarrow} \mathcal{B}_{\eta_2}^k \times \mathcal{B}_{\eta_2}$, then set $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ and $v = \mathbf{b}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot K$, and return $C = (\mathbf{u}, v)$.
- $\mathsf{Kyber.Dec}(\mathsf{sk}, C)$: compute $w \leftarrow v - \mathbf{s}^T\mathbf{u}$ and output $K = \lceil \frac{2}{q} \cdot w \rfloor$.

In the Appendix D, we recall the correctness, and the security proofs for indistinguishability and anonymity of this scheme, assuming $\eta_1 \geq \eta_2$ (while in practice, we often have $\eta_1 = \eta_2$:

Theorem 13 (Indistinguishability of Kyber.). Kyber *is IND-secure under the decisional Module-LWE assumption:*

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{Kyber}}(\tau) \leq \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k,k,\eta_1}(\tau) + \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k+1,k,\eta_2}(\tau) \leq 2 \times \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k+1,k,\eta_2}(\tau).$$

Theorem 14 (Anonymity of Kyber.). Kyber *is ANO-secure under the decisional Module-LWE assumption:*

$$\mathsf{Adv}^{\mathsf{ano}}_{\mathsf{Kyber}}(\tau) \leq 2 \times \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k,k,\eta_1}(\tau) + \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k+1,k,\eta_2}(\tau) \leq 3 \times \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k+1,k,\eta_2}(\tau).$$

7.3 Hybrid KEM, from KEM and PKE

Using the ElGamal KEM that is SK-IND-secure under the DDH assumption, and unconditionally PK-IND-secure, together with the Kyber PKE that is both IND and ANO-secure under the DMLWE assumption, the hybrid KEM is

- SK-IND-secure, as soon as either the DDH or the DMLWE assumptions hold;
- PK-IND-secure, under the DMLWE assumption.

according to Section 5.2. But with a PKE, we can optimize a little bit:

- $\mathsf{Hyb.KeyGen}(1^\kappa)$: generate both pairs of keys $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa)$ and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{PKE.KeyGen}(1^\kappa)$, then output $\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$ and $\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$;
- $\mathsf{Hyb.Enc}(\mathsf{pk})$: parse pk as $(\mathsf{pk}_1, \mathsf{pk}_2)$, choose a random $K \overset{\$}{\leftarrow} \mathcal{K}$, call $(C_1, K_1) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}_1)$ and $C_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, K \oplus K_1)$. Output $(C = (C_1, C_2), K)$;
- $\mathsf{Hyb.Dec}(\mathsf{sk}, C)$: parse sk as $(\mathsf{sk}_1, \mathsf{sk}_2)$ and C as (C_1, C_2), then call both $K_1 \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}_1, C_1)$, $K_2 \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_2, C_2)$, and output $K = K_1 \oplus K_2$.

7.4 Hybrid Traceable KEMAC

We can apply the above generic combination to build an anonymous subset-cover KEMAC with early abort, with the traceable ElGamal KEM and Kyber PKE to get a Key Encapsulation Mechanism with Access Control and Black-Box traceability (without collusions, so with $t = 1$ using notations from Section 6), where all the privacy notions (message-privacy and target-set privacy) hold as soon as at least the DDH or the DMLWE assumptions hold, while traceability works under the DDH assumption.

Detailled Description. The straightforward construction of the hybrid traceable KEMAC with early abort is the simple instanciation of the KEMAC scheme from Section 4.2 from a KEM with authentication (from Section 3.1), itself based on our hybrid KEM from the previous subsection. A naïve instanciation would draw independent keys in the hybrid schemes and send their $\oplus$'s with the encapsulated key. But as K is chosen beforehand, the same K can be chosen for all the subsets. This optimized version is described with the following algorithms, where $\mathcal{H}$ is a hash function, with output length $k + \ell$, where k is the length of the encapsulated key, ℓ the length of the verification tag, and Σ the set of subsets $(S_i)_i$ (or attributes). We instantiate it with the Kyber PKE, but it would work with any PKE that is both indistinguishable and anonymous:

– $\mathsf{HT.KEMAC.Setup}(\Sigma, 1^\kappa)$:

1. For a group $\mathbb{G}$ of prime order p, generated by g, one samples $s, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$, then sets $h = g^s$, and $g_1 = g^{s_1}, g_2 = g^{s_2}$ (for tracing purposes).
2. Then, for tracing, we set $\mathsf{tsk} = (s, s_1, s_2, \mathcal{ID})$, where $\mathcal{ID}$ is the set of the users' identifiers uid, initialized as an empty set here, and $\mathsf{tpk} = (g, h, g_1, g_2)$.
3. For each $S_i \in \Sigma$, one samples a random scalar $x_i \xleftarrow{\$} \mathbb{Z}_p$, a $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Kyber.KeyGen}(1^\kappa)$, then sets $\mathsf{pk}'_i \leftarrow (h_i = g^{x_i}, \mathsf{pk}_i)$, and $\mathsf{sk}'_i \leftarrow (x_i, \mathsf{sk}_i)$.
4. Finally, the global public key is set to $\mathsf{PK} \leftarrow (\mathsf{tpk}, \{\mathsf{pk}'_i\}_i)$, and the master secret key to $\mathsf{MSK} \leftarrow (\mathsf{tsk}, \{\mathsf{sk}'_i\}_i, \mathcal{UP})$, where $\mathcal{UP}$ is the set of user's secret keys, showing their permissions, but initialized as an empty set. One returns $(\mathsf{MSK}, \mathsf{PK})$.

– $\mathsf{HT.KEMAC.KeyGen}(\mathsf{MSK}, U, A)$:

1. For a user U, with attributes A (a list of subsets, or equivalently the indices), one samples $(\alpha, \beta) \in \mathbb{Z}_p^2$ such that $\alpha s_1 + \beta s_2 = s$, and sets the corresponding user secret identifier $\mathsf{uid} \leftarrow (\alpha, \beta)$.
2. The tracing secret key tsk is then updated as tsk' by adding (U, uid) in $\mathcal{ID}$.
3. Finally, the user's secret key is defined as $\mathsf{usk} \leftarrow (\mathsf{uid}, \{\mathsf{sk}'_j\}_{j \in A})$, and one outputs it along with MSK', the master secret key MSK updated with usk added in $\mathcal{UP}$, and tsk' instead of tsk.

– $\mathsf{HT.KEMAC.Enc}(\mathsf{PK}, B)$:

1. For a target set that covers all the users with an attribute in B (or equivalently the indices, such that $A \cap B \neq \emptyset$), one generates a random seed for the key to be encapsulated, $S \xleftarrow{\$} \{0,1\}^k$, then draws $r \xleftarrow{\$} \mathbb{Z}_p^*$, sets $c = (C_1 = g_1^r, C_2 = g_2^r)$, and, for each $i \in B$, with $\mathsf{pk}'_i = (h_i = g^{x_i}, \mathsf{pk}_i)$, sets $K_i = \mathcal{H}(h_i^r)$, and then sets $E_i \leftarrow \mathsf{Kyber.Enc}(\mathsf{pk}_i, S \oplus K_i)$[5].
2. One then computes $K||V \leftarrow \mathcal{H}(S)$, in order to grant the early aborts paradigm, and sets the encapsulation as: $C \leftarrow (c, \{E_i\}_{i \in B}, V)$, the encapsulated key as K, and outputs: (K, C).

– $\mathsf{HT.KEMAC.Dec}(\mathsf{usk} = (\mathsf{uid} = (\alpha, \beta), \{\mathsf{sk}_j\}_{j \in A}), C = (c, \{E_i\}_{i \in B}, V))$: For each $\mathsf{sk}'_j = (x_j, \mathsf{sk}_j)$ in usk and (c, E_i, V) in C (for one index i), one decapsulates the underlying hybrid KEM to get the potential seed S used for the key:

- first, $K'_{i,j} \leftarrow \mathsf{Kyber.Dec}(\mathsf{sk}_j, E_i)$;
- for ElGamal, from $c = (C_1, C_2)$, one computes $K_j \leftarrow \mathcal{H}((C_1^\alpha C_2^\beta)^{x_j})$;
- $S_{i,j}$ is then computed as $S_{i,j} \leftarrow K'_{i,j} \oplus K_j$.

In the early-abort check, one computes $U'_{i,j}||V'_{i,j} \leftarrow \mathcal{H}(S_{i,j})$, and checks whether $V'_{i,j} = V$. In the positive case, one returns $K \leftarrow U'_{i,j}$, for this first valid (i, j), as the session key. Else, if $V'_{i,j} \neq V$, the ciphertext is rejected and the loop on the i, j indices goes on[6].

Security Analysis. Our $\mathsf{HT.KEMAC}$ scheme inherits its security properties from the underlying hybrid KEM scheme using both the Kyber PKE and the traceable ElGamal KEM, and as such, is $\mathsf{SK\text{-}IND}$-secure as soon as either the DDH or the DMLWE assumptions hold, and $\mathsf{PK\text{-}IND}$-secure under the DMLWE assumption. Correctness also follows from the authentication property of the hybrid KEM, and thus under either the DDH or the DMLWE assumptions.

[5] Note that this is the optimized version of a generic one where one would have drawn $|B|$ extra session keys K'_i, E_i would actually have been a Kyber encryption of these K'_i's instead of the $S \oplus K_i$, and one would have had to send $|B|$ extra $F_i \leftarrow K_i \oplus K' \oplus S$.

[6] Again, this corresponds to our optimized version, taking advantage of the encrypting properties of Kyber. For a generic hybrid KEMAC, one would have output $U'_{i,j} \oplus F_i$ when $V'_{i,j} = V$ (cf. previous footnote for the definition of F_i).

Traceability. The traceability is inherited from the underlying traceable ElGamal KEM scheme, with $t = 1$ in Section 6's notations; it relies on the DDH. To check whether a user U with $\mathsf{uid} = (\alpha, \beta)$ using the key sk – which is shared among her and other users – is corrupted, one encapsulates a key that only this user can decapsulate with sk, because the ElGamal encapsulations are group elements with exponent a random linear combination of a vector which is orthogonal to (α, β), following the confirmer construction from Section 6. We stress that our construction with $t = 1$ does not allow collusions. But it could be extended to confirm larger collusions of traitors.

8 Implementation

8.1 Parameters of CoverCrypt

The parameters of Kyber are recalled in the Appendix D, Table 3, with the sizes (in Bytes) of the keys and ciphertexts, using the compression/decompression, as this does not impact our security results, and the Kyber PKE can be used as a black-box from any library.

We have done an implementation in Rust of CoverCrypt (an hybrid anonymous Subset-Cover KEMAC with Early-Abort) scheme with optimization for a security of 128 bits. The source will be made publicly available. We use Kyber-768 (and its `pqd_kyber` librairie[7]) and ElGamal on the Curve25519, as group that is of prime order $p = 2^{255} - 19$. The hash algorithm used to generate the Early-Abort tags (256 bits) and the keys (256 bits) generated by the KEM is SHAKE-256. Then we present the sizes of the keys and ciphertexts, according to the sizes of A and B, in Table 1.

Table 1. Sizes of Keys and Ciphertexts (in Bytes) according the sizes of A and B

Size of A	1	2	3	4	5
Secret Key usk	128	192	256	320	384
Size of B	1	2	3	4	5
Ciphertext C	1184	2272	3360	4448	5536

8.2 Benchmarks

The following benchmarks are performed on an Intel(R) Core(TM) i5-6200U CPU @2.8GHz Table 2 provides the time required to generate CoverCrypt encapsulations and decapsulations for a 32-Byte symmetric key, depending on the number of attributes in the user key (size of A) and the size of the target set B.

Table 2. Time (in μs) for Encapsulation/Decapsulation

Size of B				
1	2	3	4	5
355	495	608	796	959

Encapsulation

	Size of B				
Size of A	1	2	3	4	5
1	234	291	396	395	480
2	231	337	499	577	663
3	220	388	594	711	870

Decapsulation

[7] `https://docs.rs/pqc_kyber/latest/pqc_kyber/`

References

ABD^{+}21. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation. `https://pq-crystals.org/kyber/resources.shtml`, 2021.

BF99. Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Heidelberg, August 1999.

DDP^{+}17. Wei Dai, Yarkın Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savaş, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. Cryptology ePrint Archive, Report 2017/601, 2017. `https://eprint.iacr.org/2017/601`.

FP12. Nelly Fazio and Irippuge Milinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 225–242. Springer, Heidelberg, May 2012.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

LG18. Jiangtao Li and Junqing Gong. Improved anonymous broadcast encryptions - tight security and shorter ciphertext. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 497–515. Springer, Heidelberg, July 2018.

LPQ12. Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 206–224. Springer, Heidelberg, May 2012.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

Sho01. Victor Shoup. A proposal for an iso standard for public key encryption. `https://shoup.net/papers/iso-2_1.pdf`, 2001.

Wee21. Hoeteck Wee. ABE for DFA from LWE against bounded collusions, revisited. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 288–309. Springer, Heidelberg, November 2021.

Appendix

A Some Figures

```
KEM'.KeyGen(1^κ):
1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa)$
2. return $(\mathsf{pk}, \mathsf{sk})$

KEM'.Enc(pk):
1. $(c, s) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$
2. $U \| V \leftarrow \mathcal{H}(s)$
3. $C \leftarrow (c, V)$, $K \leftarrow U$
4. return $(C, K)$

KEM'.Dec(sk, pk, C = (c, V)):
1. $s \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}, c)$
2. $U' \| V' \leftarrow \mathcal{H}(s)$
3. if $V = V'$, return $U'$, otherwise reject
```

Fig. 1. Key-Confirmation KEM'

```
KEM.KeyGen(1^κ):
1. $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}_i(1^\kappa)$, for $i = 1, 2$
2. $\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$; $\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
3. return $(\mathsf{pk}, \mathsf{sk})$

KEM.Enc(pk):
1. $(C_i, K_i) \leftarrow \mathsf{KEM}_i.\mathsf{Enc}(\mathsf{pk}_i)$, for $i = 1, 2$
2. $C \leftarrow (C_1, C_2)$; $K \leftarrow K_1 \oplus K_2$
3. return $(C, K)$

KEM.Dec(sk, C):
1. $K_i \leftarrow \mathsf{KEM}_i.\mathsf{Dec}(\mathsf{sk}_i, C_i)$, for $i = 1, 2$
2. $K \leftarrow K_1 \oplus K_2$
3. return $K$
```

Fig. 2. Hybrid KEM

B Some Deferred Proofs

B.1 Proof of Theorem 1

We present a sequence of games, starting from the initial $\mathsf{SK\text{-}IND}$ security game against KEM':

Game $\mathbf{G}_0$: In the initial game, a challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa)$, $(c, s) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$, and evaluates $U \| V \leftarrow \mathcal{H}(s)$ to output $(\mathsf{pk}, C = (c, V), K_b)$, where $b \overset{\$}{\leftarrow} \{0, 1\}$ is a random bit, $K_0 = U$, and $K_1 \overset{\$}{\leftarrow} \{0, 1\}^k$. From this output, the adversary returns her guess b' for b. We denote P_0 the probability of event $b' = b$ happening, which is $(1 + \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}'}(\mathcal{A}))/2$.

Game $\mathbf{G}_1$: In this game, $s \overset{\$}{\leftarrow} \mathcal{K}$ is this time drawn uniformly at random from the session-key space of KEM, and one sets: $U \| V \leftarrow \mathcal{H}(s)$. The difference is the $\mathsf{SK\text{-}IND}$-game on the underlying KEM. Hence, $P_0 - P_1 \leq \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau)$, where τ is the maximum running-time of adversary $\mathcal{A}$.

Game $\mathbf{G}_2$: In this game, one takes $U \| V \xleftarrow{\$} \{0,1\}^{k+\ell}$, which is indistinguishable unless the query s has been asked to $\mathcal{H}$ by the adversary, which is only possible with probability bounded by $q_h/\#\mathcal{K}$. Hence, $P_1 - P_2 \leq q_h/\#\mathcal{K}$. In this final game, this is clear that $P_2 = 1/2$, as U and K are both randomly drawn in $\{0,1\}^k$.

Hence, $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}'}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau) + 2 \cdot q_h/\#\mathcal{K}$. $\square$

B.2 Proof of Theorem 2

We present a sequence of games, starting from the initial PK-IND security game against KEM':

Game $\mathbf{G}_0$: In this game, for every i in $\{0;1\}$, a challenger runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa)$, $(c_i, s_i) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}_i)$, and gets $U_i \| V_i \leftarrow \mathcal{H}(s_i)$, then outputs $(\mathsf{pk}_0, \mathsf{pk}_1, C_b = (c_b, V_b))$, for a random $b \xleftarrow{\$} \{0,1\}$. From the output, an adversary $\mathcal{A}$ outputs her guess b' for b. We denote P_0 the probability of the event $b' = b$ happening, which is equal to $(1 + \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}'}(\mathcal{A}))/2$.

Game $\mathbf{G}_1$: This game is as the previous one, except that now, one uses $s_i \xleftarrow{\$} \mathcal{K}$ to evaluate $U_i \| V_i \leftarrow \mathcal{H}(s_i)$, for every i in $\{0,1\}$. The correct guess probability difference is the SK-IND-game on the underlying KEM, for both i's in $\{0,1\}$. Hence, $P_0 - P_1 \leq 2 \times \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau)$, where τ is the maximum running-time of adversary $\mathcal{A}$.

Game $\mathbf{G}_2$: This game is as the previous one, except that now, for each $i \in \{0,1\}$, the challenger draws $U_i \| V_i \xleftarrow{\$} \{0,1\}^{k+\ell}$. This is indistinguishable unless a query s_i has been asked to $\mathcal{H}$ by the adversary, which is only possible with probability bounded by $2q_h/\#\mathcal{K}$, as there are two possible bad queries. Hence, $P_1 - P_2 \leq 2q_h/\#\mathcal{K}$.

Game $\mathbf{G}_3$: Now, one outputs $(\mathsf{pk}_0, \mathsf{pk}_1, C_b = (c_b, V))$, for a random $b \xleftarrow{\$} \{0,1\}$ and a random $V \xleftarrow{\$} \{0,1\}^\ell$. This simulation is perfectly indistinguishable from the previous game: $P_2 = P_3$. And this final game is exactly the PK-IND-game on the underlying KEM: $P_3 = (1 + \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}}(\mathcal{A}))/2$.

Hence, $\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}'}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}}(\tau) + 4 \times \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau) + 4q_h/\#\mathcal{K}$. $\square$

B.3 Proof of Theorem 4

In the selective setting, the adversary asks, from the beginning, the keys it wants to get, before seeing the global public parameters PK.

Game $\mathbf{G}_0$: In the initial game, the adversary thus asks for the keys it wants: for several sets A_j. One calls $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}(1^\kappa)$, for each $S_i \in \Sigma$, and provides PK together with all the asked keys sk_i, for $i \in A = \cup A_j$ (all the asked sets). The adversary answers with a set B, but with the constraint that $A \cap B = \emptyset$, and the challenger flips a random bit $b \xleftarrow{\$} \{0,1\}$, generates two random session keys $K'_0, K'_1 \leftarrow \mathcal{K}$, runs $(C_i, K_i) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}_i)$ for all $i \in B$, and outputs $C \leftarrow (i, C_i, E_i = K'_0 \oplus K_i)_{i \in B}$ together with the challenged key K'_b (that is either the really encapsulated key if $b = 0$ or a random key if $b = 1$). The adversary outputs its guess b'. We denote P_0 the probability of event $b' = b$, which is $(1 + \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}))/2$.

Game $\mathbf{G}_1$: In this game, we replace all the K_i's by $K_i \xleftarrow{\$} \mathcal{K}$ in the generation of E_i. To show this game is indistinguishable from the previous one, we define a sequence of hybrid games, for index I, such that for all $i < I$, one replaces K_i by a random element in $\mathcal{K}$. For $I = 1$, this is $\mathbf{G}_0$, whereas for $I = q_k + 1$, where q_k is the maximal number of keys, this is $\mathbf{G}_1$. And the gap between I and $I + 1$ is the SK-IND-game on the underlying KEM. Hence, $P_0 - P_1 \leq q_k \times \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau)$, where τ is the maximum running-time of adversary $\mathcal{A}$.

Game $\mathbf{G}_2$: In this game, we replace all the E_i's by $E_i \xleftarrow{\$} \mathcal{K}$, which is perfectly indistinguishable from $K'_0 \oplus K_i$ for a random K_i, under the group-law property. Hence, $P_1 = P_2$. In this final game, this is clear that $P_2 = 1/2$, as K'_0 and K'_1 do not appear anymore in C, and so K'_b is just a random key.

Hence, $\mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}) \leq 2q_k \times \mathsf{Adv}^{\mathsf{sk\text{-}ind}}_{\mathsf{KEM}}(\tau)$. $\square$

B.4 Proof of Theorem 7

The proof is quite similar to above proof. In the selective setting, the adversary asks, from the beginning, the keys it wants to get, before seeing the global public parameters PK.

Game $\mathbf{G}_0$: In the initial game, the adversary thus asks for the keys it wants: for several sets A_j. One calls $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM}'.\mathsf{KeyGen}(1^\kappa)$, for each $S_i \in \Sigma$, and provides PK together with all the asked keys sk_i, for $i \in A = \cup A_j$ (all the asked sets). The adversary answers with two sets B_0 and B_1, but with the constraints that $A \cap B_0 = A \cap B_1 = \emptyset$ and $|B_0| = |B_1|$, and the challenger flips a random bit $b \overset{\$}{\leftarrow} \{0,1\}$, generates a random session key $K \leftarrow \mathcal{K}$, runs $(C_i, K_i) \leftarrow \mathsf{KEM}'.\mathsf{Enc}(\mathsf{pk}_i)$ for all $i \in B_0 \cup B_1$, and outputs $C \leftarrow (C_i, E_i = K \oplus K_i)_{i \in B_b}$ together with the challenged key K. The adversary outputs its guess b'. We denote P_0 the probability of event $b' = b$, which is $(1 + \mathsf{Adv}^{\mathsf{ac\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}))/2$.

Game $\mathbf{G}_1$: In this game, one always outputs $C \leftarrow (C_i, E_i = K \oplus K_i)_{i \in B_0}$. To show this game is indistinguishable from the previous one, we define a sequence of hybrid games, for index I, such that for the I first $(C_i, E_i = K \oplus K_i)$ in C are for indices in B_0 and the last are for indices in B_b. For $I = 0$, this is $\mathbf{G}_0$, whereas for $I = |B_0| = |B_1|$, this is $\mathbf{G}_1$. And the gap between I and $I+1$ is the $\mathsf{PK\text{-}IND}$-game on the underlying KEM'. Hence, $P_0 - P_1 \leq |B_0| \times \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}'}(\tau)$, where τ is the maximum running-time of adversary $\mathcal{A}$.

Furthermore, in this final game, this is clear that $P_1 = 1/2$, as the challenge C does not depend on b anymore.

Hence, $\mathsf{Adv}^{\mathsf{ac\text{-}ind}}_{\mathsf{KEMAC}}(\mathcal{A}) \leq 2S_B \times \mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{KEM}'}(\tau)$, where S_B is the constant-size of the sets B. □

C Public-Key Encryption

Indistinguishability. A PKE is *indistinguishable* (denoted IND) if for an honestly generated pk, when the adversary chooses two messages m_0 and m_1, it cannot distinguish an encryption of m_0 from an encryption of m_1, both under pk. More formally, a PKE is IND-secure if for any adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{PKE}}(\mathcal{A}) = \mathsf{negl}(\kappa)$, for $b \overset{\$}{\leftarrow} \{0,1\}$ and

$$\mathcal{D}_b = \left\{ \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE}.\mathsf{KeyGen}(1^\kappa), \\ (m_0, m_1, s) \leftarrow \mathcal{A}(\mathsf{pk}), C \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}, m_b) \end{array} : (\mathsf{pk}, C, s) \right\}$$
$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{PKE}}(\mathcal{A}) = 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}(\mathsf{pk}, C, s) = b] - 1.$$

where s is an internal state the adversary keeps between the two steps.

Anonymity. A PKE is *anonymous*, also known as *public-key privacy* (denoted $\mathsf{PK\text{-}IND}$) if fr two honestly generated pk_0 and pk_1, when the adversary chooses a message m, it cannot distinguish an encryption of m under pk_0 from the encryption of m under pk_1. More formally, a PKE is $\mathsf{PK\text{-}IND}$-secure if for any adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{PKE}}(\mathcal{A}) = \mathsf{negl}(\kappa)$, for $b \overset{\$}{\leftarrow} \{0,1\}$ and

$$\mathcal{D}_b = \left\{ \begin{array}{l} \text{For } i = 0, 1 : \\ \quad (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{PKE}.\mathsf{KeyGen}(1^\kappa), \\ (m, s) \leftarrow \mathcal{A}(\mathsf{pk}_0, \mathsf{pk}_1), C \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_b, m) \end{array} : (\mathsf{pk}_0, \mathsf{pk}_1, C, s) \right\}$$
$$\mathsf{Adv}^{\mathsf{pk\text{-}ind}}_{\mathsf{PKE}}(\mathcal{A}) = 2 \times \Pr_{\mathcal{D}_b}[\mathcal{A}(\mathsf{pk}_0, \mathsf{pk}_1, C, s) = b] - 1.$$

D CRYSTALS-Kyber

D.1 Correctness

We can first show that decryption indeed gives back K. To this aim, one can note that we have

$$\begin{aligned} w = v - \mathbf{s}^T\mathbf{u} &= (\mathbf{b}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot K) - \mathbf{s}^T \cdot (\mathbf{A}^T\mathbf{r} + \mathbf{e}_1) \\ &= (\mathbf{s}^T\mathbf{A}^T + \mathbf{e}^T) \cdot \mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot K - \mathbf{s}^T \cdot (\mathbf{A}^T\mathbf{r} + \mathbf{e}_1) \\ &= \mathbf{s}^T\mathbf{A}^T\mathbf{r} + \mathbf{e}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot K - \mathbf{s}^T\mathbf{A}^T\mathbf{r} + \mathbf{s}^T\mathbf{e}_1 \\ &= \mathbf{e}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot K + \mathbf{s}^T\mathbf{e}_1 = \lceil \frac{q}{2} \rfloor \cdot K + z \end{aligned}$$

with $z = \mathbf{e}^T\mathbf{r} + \mathbf{s}^T\mathbf{e}_1 + e_2$. Then,

$$\lceil \frac{2}{q} \cdot w \rfloor = \lceil \frac{2}{q} \cdot \left(\lceil \frac{q}{2} \rfloor \cdot K + z \right) \rfloor.$$

As $K \in \{0,1\}^n$, if $\|z\|_\infty < q/4$, then $\lceil \frac{2}{q} \cdot w \rfloor = K$. And since $\mathbf{e}, \mathbf{r}, \mathbf{s}$ are drawn according to $\mathcal{B}_{\eta_1}$, and $\mathbf{e}_1, e_2$ according to $\mathcal{B}_{\eta_2}$, with appropriate n, k, q, η_1, η_2, $\|z\|_\infty < q/4$, with overwhelming probability [ABD+21], even with the additional compression function, that introduces an additional noise. And we omit it in the current description, as it is for efficiency purpose but does not impact the security notions we are interested in.

D.2 Indistinguishability of Kyber

We proceed with a sequence of games, so that the final game does not leak any information about the bit b used by the challenger when encrypting K_b among K_0 and K_1 chosen by the adversary.

Game $\mathbf{G}_0$: The initial game is the security experiment, where a public key $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$ is generated, as well as a ciphertext C of K_b.

Game $\mathbf{G}_1$: We change the way the challenger generates the public key pk, with randomly chosen $\mathbf{b} \overset{\$}{\leftarrow} \mathsf{R}_q$. The distance of the distributions of the outputs of the adversary is then bounded by $\mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k,k,\eta_1}(\tau)$. But still, the ciphertext is built as

$$\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{A} | \mathbf{b} \end{bmatrix}^T \times \mathbf{r} + \begin{bmatrix} \mathbf{e}_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor \cdot K_b \end{bmatrix}.$$

Game $\mathbf{G}_2$: We change now the simulation of the ciphertext, with $(\mathbf{w}, z) \overset{\$}{\leftarrow} \mathsf{R}_q^k \times \mathsf{R}_q = \mathsf{R}_q^{k+1}$, and

$$\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{w} \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor \cdot K_b \end{bmatrix}.$$

The distance of the distributions of the outputs of the adversary is then bounded by $\mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k+1,k,\eta_2}(\tau)$, as $\mathbf{r} \overset{\$}{\leftarrow} \mathcal{B}^k_{\eta_1}$ and $(\mathbf{e}_1, e_2) \overset{\$}{\leftarrow} \mathcal{B}^k_{\eta_2} \times \mathcal{B}_{\eta_2} = \mathcal{B}^{k+1}_{\eta_2}$, and $\eta_1 \geq \eta_2$.

Game $\mathbf{G}_3$: We eventually replace C by a random sampling in $\mathsf{R}_q^k \times \mathsf{R}_q$, which is perfectly indistinguishable. And then, the ciphertext is perfectly independent from K_b.

D.3 Anonymity of Kyber

We will show that the ciphertexts are statistically close to uniform in the ciphertext space, i.e. in $\mathsf{R}_q^k \times \mathsf{R}_q$, whatever the public key.

Game $\mathbf{G}_0$: The initial game is the security experiment, where two public keys $\mathsf{pk}_0 = (\mathbf{A}_0, \mathbf{b}_0)$ and $\mathsf{pk}_1 = (\mathbf{A}_1, \mathbf{b}_1)$ are generated and a ciphertext C is generated according to pk_b, an a message K chosen by the adversary.

Game $\mathbf{G}_1$: We change the way the challenger generates the public keys pk_0 and pk_1, with randomly chosen $\mathbf{b}_0, \mathbf{b}_1 \xleftarrow{\$} \mathsf{R}_q$. The distance of the distributions of the outputs of the adversary is then bounded by $2 \times \mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k,k,\eta_1}(\tau)$. But still, the ciphertext is built as

$$\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{A}_b | \mathbf{b}_b \end{bmatrix}^T \times \mathbf{r} + \begin{bmatrix} \mathbf{e}_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor \cdot K \end{bmatrix}.$$

Game $\mathbf{G}_2$: We change now the simulation of the ciphertext, with $(\mathbf{w}, z) \xleftarrow{\$} \mathsf{R}_q^k \times \mathsf{R}_q = \mathsf{R}_q^{k+1}$, and

$$\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{w} \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor \cdot K \end{bmatrix}.$$

The distance of the distributions of the outputs of the adversary is then bounded by $\mathsf{Adv}^{\mathsf{dmlwe}}_{\mathsf{R}_q,k+1,k,\eta_2}(\tau)$, as $\mathbf{r} \xleftarrow{\$} \mathcal{B}_{\eta_1}^k$ and $(\mathbf{e}_1, e_2) \xleftarrow{\$} \mathcal{B}_{\eta_2}^k \times \mathcal{B}_{\eta_2} = \mathcal{B}_{\eta_2}^{k+1}$, and $\eta_1 \geq \eta_2$.

Game $\mathbf{G}_3$: We eventually replace C by a random sampling in $\mathsf{R}_q^k \times \mathsf{R}_q$, which is perfectly indistinguishable. And then, the ciphertext is perfectly independent from the used public key.

D.4 Parameters

In Table 3, we recall the parameters of Kyber, where δ is the security level (one can see $\delta = 2^{-\kappa}$), and (d_u, d_v) are some parameters for the compression function, with the sizes (in Bytes) of the keys and ciphertexts, using the compression/decompression.

Table 3. Parameter Sets for Kyber

Parameters	n	k	q	η_1	η_2	(d_u, d_v)	pk	sk	c	δ
Kyber-512	256	2	3329	3	2	(10,4)	800B	32B	736B	2^{-139}
Kyber-768	256	3	3329	2	2	(10,4)	1184B	32B	1088B	2^{-164}
Kyber-1024	256	4	3329	2	2	(11,5)	1568B	32B	1568B	2^{-174}