IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 4, 25 pages.

https://doi.org/10.62056/ah890lmol

Check for updates

Split-key PRFs and Extended Hybrid Security for KEM Combiners

Lise Millerjord[1], Douglas Stebila[2] and Camryn Steckel[2]

[1] NTNU – Norwegian University of Science and Technology, Trondheim, Norway
[2] University of Waterloo, Waterloo, Canada

Abstract. Key encapsulation mechanism (KEM) combiners allow for the construction of hybrid KEMs that are secure as long as at least one of several underlying ingredient KEMs remains secure. In PKC 2018, Giacon, Heuer, and Poettering showed that parallel KEM combiners whose core function is a split-key pseudorandom function (PRF) satisfy IND-CCA security if at least one of the ingredient KEMs satisfies IND-CCA security. However, their result assumes that public keys of the combined KEM are generated independently from any instances of the ingredient KEMs, which may not hold in real-world applications. To address this, we introduce a new security model which captures adversarial access to both the combined KEM and (post-processed versions of) the ingredient KEMs. We show that security in this extended model can still be achieved if at least one ingredient KEM satisfies IND-CCA security, the core function is a split-key PRF, and the ingredient KEM outputs are post-processed using standard PRFs. We consider an application of this approach to hybrid KEMs in the S/MIME secure email standard. We also provide a new construction for a split-key PRF, which uses a t-resilient extractor to output a string of truly random bits from an input in which the adversary controls t bits, and show that this split-key PRF construction is secure in the standard model.

Keywords: hybrid security · key encapsulation mechanisms · combiners · split-key pseudorandom function

1 Introduction

In choosing a cryptographic protocol, we find ourselves in a situation with many options, but not necessarily well-established ways of deciding which protocol is better suited to our purposes, in terms of security level, properties and assumptions. In particular, this is clear when we are working with key exchange protocols and consider the shift toward protocols that are secure against quantum adversaries.

There are key exchange protocols relying on a variety of different computational hardness assumptions, and in some cases it can be useful to have a protocol that is secure as long as at least one of these computational hardness assumptions holds. For instance, during the transition to post-quantum cryptography, it may be useful to have protocols which rely on both classical and post-quantum assumptions.

We are, after many years of scrutiny, familiar with the classical protocols and the hardness assumptions used in those. However, since they are not secure against quantum adversaries, we are forced to engage with newer protocols whose hardness assumptions may have faced less scrutiny and cryptanalysis. In this context, we look to hybrid solutions. Recently, we have been spurred on in our motivation to do this by the spectacular breaking

E-mail: lise.millerjord@gmail.com (Lise Millerjord), dstebila@uwaterloo.ca (Douglas Stebila), casteckel@uwaterloo.ca (Camryn Steckel)

Received: 2025-10-06 Accepted: 2025-12-02

of SIDH, and consequently SIKE by Castryck and Decru, as well as Maino, Martindale, Panny, Pope and Wesolowski in 2022 [CD23, MMP+23].

Hybrid key exchange. In hybrid protocols, we are combining two different ways of deriving a shared secret key in order to get the benefits of both methods. When we do this, we run into the issue of reliably combining two keys in a way that preserves the security of the strongest of the components.

When we have one key we can use a standard key derivation function or a PRF with the standard security assumptions. When we have two (or more) keys we need to analyze this differently. Since a PRF takes two inputs, it is natural to consider inputting both component keys into a PRF, and using the resulting output as a combined key. In doing this, we would be assuming that the PRF is pseudorandom in either of its inputs. This is called the dual PRF assumption. Backendal, Bellare, Günther, and Scarlata show that HMAC is, under certain conditions and with some idealization, a dual PRF [BBGS23].

Examples of protocol analyses applying the dual PRF assumption include the handshake protocol of TLS 1.3 [Res18, DFGS21]. Others make use of the random oracle model, such as the security analysis of the hybrid group key exchange protocol presented by Boyd, Fondevik, Gjøsteen, and Millerjord [BFGM23].

The TLS 1.3 handshake protocol has many stages which derive multiple intermediate and output keys along the way. In the TLS 1.3 pre-shared key (PSK) handshake with forward secrecy (PSK-ECDHE), the handshake secret is computed as

$$HS \leftarrow \mathsf{HKDF.Extract}(PSK', g^{xy}) \ .$$

Here, we are in the exact scenario we have described with two input keys and the dual PRF assumption. There are also several other places in the protocol where we input a key into either one of the input fields.

However, it turns out that there are many constructions that are not secure under the dual PRF assumption which are still sufficient to preserve the appropriate security properties for hybrid key exchange. This is why we are particularly interested in a weaker primitive, a *split-key PRF*, as introduced by [GHP18].

We also consider key encapsulation mechanisms (KEMs). KEMs are a cornerstone of public key encryption, and as a result they come in many different shapes and forms.

For KEMs, we want to construct a KEM combiner and analyse its security. We will discuss the security of a KEM combiner both as a KEM itself, but also as a hybrid system, where we may also be using the component KEMs outside the combiner with the same key material.

KEM combiners. Giacon, Heuer, and Poettering introduced the idea of KEM combiners in [GHP18]. A KEM combiner is a KEM which accesses a set of "ingredient" KEMs in order to generate a public/secret key pair, encapsulate a public key to produce a shared secret/ciphertext pair, and decapsulate a ciphertext to reproduce a shared secret, in such a way that it is secure if at least one of the ingredient KEMs is secure. [GHP18] achieve this using a parallel construction for their KEM combiners. The key generation algorithm for the combined KEM consists of running the key generation algorithms for each of the n ingredient KEMs in parallel, and then defining the combined keys to be $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$, where $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ and $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ are the ingredient public/secret keys. The encapsulation algorithm for the combined KEM consists of running the encapsulation algorithms for the ingredient KEMs in parallel to produce n shared secret/ciphertext pairs $(k_1, c_1), \ldots, (k_n, c_n)$, putting these shared secrets and ciphertexts into a "core function" W, and outputting $(W(k_1, \ldots, k_n, c_1 \| \cdots \| c_n), c_1 \| \cdots \| c_n)$ as the combined shared secret/ciphertext pair. Finally, the decapsulation algorithm for the combined KEM simply inverts the encapsulation algorithm: it runs the decapsulation

algorithm for each of the ingredient KEMs on their corresponding ciphertext, and then computes W on the resulting outputs.

For parallel KEM combiners, any type of security guarantee is dependent on the core function used. For IND-CCA security, we need a core function which takes in a tuple of keys and a label as input, and outputs a key which is indistinguishable from random if at least one of the keys used is unknown to the adversary. As it turns out, if we take the label to be a nonce, then this is precisely the definition of a split-key PRF. [GHP18] showed that if the core function in a parallel KEM combiner is a split-key PRF, and at least one ingredient KEM satisfies IND-CCA security, then the combined KEM also satisfies IND-CCA security. It is worth noting that in the case of KEM combiners, a nonce is readily available by concatenating the ingredient ciphertexts together with the ingredient public keys (or by simply concatenating the ingredient ciphertexts, like in [GHP18]).

Security of KEM combiners. The notion of IND-CCA security for a KEM assumes that a KEM acts in isolation of its environment. In most cases, this is fine, because the public/secret keys for the KEM are generated independently of any other primitives that the adversary could interact with, and thus the adversary is only able to gain relevant information by interacting with the KEM in question. However, for a KEM combiner, this may not hold true. Consider the case where a user uses the same public/secret key pairs for the ingredient KEMs as they use for the combined KEM. In this scenario, an adversary could theoretically obtain relevant information about the combined KEM by interacting with the ingredient KEMs, which is not captured in the definition of IND-CCA security. Since reuse of public/secret is not explicitly banned in common use-cases such as S/MIME, this is a real concern, and is something that should be considered when analyzing security.

To account for this, in Section 4, we extend the definition of IND-CCA security to model a system of KEMs consisting of a combined KEM and its corresponding ingredient KEMs. The most natural way to do this is to give the adversary unrestricted access to decapsulation oracles for each of the ingredient KEMs, as well as restricted access to a decapsulation oracle for the combined KEM. However, since the combined decapsulation algorithm can be executed by anyone with unrestricted access to the ingredient secret keys, then an adversary with these oracles could trivially win this experiment by simply executing the combined decapsulation algorithm, and querying the appropriate ingredient decapsulation oracle every time an ingredient secret key is required. To solve this problem, we instead provide the adversary with ingredient decapsulation oracles which *post-process* the shared secret before returning it to the adversary. In this new model, security of the combined KEM is not only dependent on the IND-CCA security of the ingredient KEMs, but also on the choice of post-processing function used. Under this new security definition, we show that a parallel KEM combiner is secure if at least one of the ingredient KEMs satisfies IND-CCA security, the core function is a split-key PRF, and the post-processing functions used are PRFs. We also provide a trivial attack in our model on combined KEMs which use a class of post-processing functions (including the identity function), which displays why the post-processing functions are necessary, and also serves as a separating example between our hybrid model and the "regular" model of ind-cca security for KEMs.

As it turns out, the choice of PRFs as post-processing function models the information that an adversary attacking S/MIME would potentially have access to. S/MIME is the Internet standard specified in [SRT19] which specifies how to provide cryptographic security to email messages, and in particular, it supports the use of KEMs. Loosely speaking, in S/MIME (with a KEM), an originator encrypts data using symmetric-key encryption with a randomly-derived content-encryption key, then encrypts the content-encryption key using a key-encryption key obtained by post-processing the shared secret output from the recipient's key encapsulation mechanism (KEM). The originator then sends both of these encryptions to the recipient, who is able to decrypt as they have knowledge of their KEM secret key. A recipient may support multiple KEMs that they are willing to use, and one

of them might be a combined KEM that combines several of the individual KEMs they support. Moreover, it is possible that a recipient might use the same set of public/secret keys for both the individual and combined KEMs, in which case an adversary interacting with the system would be in exactly the position that our security definition models. This setup would provide the adversary with fairly natural decapsulation oracles for both the individual and combined KEMs, as it is common for a recipient to respond to emails and include quoted text in their response. We go into more detail on this setting in Section 4.3.

Split-key PRFs. The remaining challenge is to construct a secure split-key PRF. [GHP18] provide a few options which are secure in the random oracle model, and one option which is secure in the standard model. In Section 3, we provide a new construction for a split-key PRF that is secure in the standard model. Our construction differs from the standard model construction in [GHP18] as it is based on t-resilient extractors, whereas the construction in [GHP18] is based on the XOR function. An advantage of our construction is that it is compliant with NIST's Recommendation [EB20] which approves the formation of hybrid shared secrets via concatenation, but not via XOR. On the other hand, the construction in [GHP18] is more efficient than ours. Nonetheless, we believe it is interesting to have a second split-key PRF construction that is secure in the standard model, which is constructed from an entirely different approach.

As mentioned above, our construction is based on t-resilient extractors, which were first introduced in [BST03]. A t-resilient extractor is a function E^π which takes in a (long) input from a high-entropy source, of which the adversary is allowed to control t bits, as well as a public parameter π, and outputs a string of truly random bits. This lends itself well to building a split-key PRF, as we need a primitive which takes in n keys (with $n-1$ of them potentially being controlled by the adversary), and outputs a pseudorandom string. The t-resilient extractor is an information-theoretically secure construction, which we choose to use since it explicitly models the adversary's control over the input to the function.

The most straightforward thing to do would be to simply concatenate our keys together and use them as input to the t-resilient extractor; for instance:

$$Z(k_1, \ldots, k_\ell) = E^\pi\big(k_1 \| \cdots \| k_\ell\big) \ .$$

However, we immediately run into the problem that the input is not long enough. So, we must first extend each of the inputs by running them through a PRG first:

$$Z(k_1, \ldots, k_\ell) = E^\pi\big(\mathsf{PRG}(k_1) \| \ldots \| \mathsf{PRG}(k_\ell)\big) \ .$$

Unfortunately, this still does not work. Like many information theoretic security properties, the t-resilience only holds if the adversary is only able to obtain a single sample under the hidden key, whereas in any kind of PRF property, the adversary is allowed to obtain multiple samples under the same hidden key. To fix this, we need some way of ensuring uniqueness of the inputs to the t-resilient extractor, even when one of the input keys is left unchanged. A nice solution would be to apply a PRF to each key before sending it into the PRG, using the other keys as the label so as not to increase the number of inputs, as follows:

$$Z(k_1, \ldots, k_\ell) = E^\pi\Big(\mathsf{PRG}\big(\mathsf{PRF}(k_1, k_2 \| \cdots \| k_\ell)\big) \| \ldots \| \mathsf{PRG}\big(\mathsf{PRF}(k_\ell, k_1 \| \cdots \| k_{\ell-1})\big)\Big) \ .$$

With this construction, we run into problems in the security proof, as the keys are used in multiple places. So, our final solution is to use a nonce (denoted n) instead:

$$Z(k_1, \ldots, k_\ell, n) = E^\pi\Big(\mathsf{PRG}\big(\mathsf{PRF}(k_1, n)\big) \| \ldots \| \mathsf{PRG}\big(\mathsf{PRF}(k_\ell, n)\big)\Big) \ .$$

The nonce prevents the reuse of keys, which allows the security proof to go through, and since a split-key PRF already uses a nonce, then this does not cause any problems for

our intended use-cases, where appropriate nonces are readily available. We are able to show that this construction is a secure split-key PRF in the standard model.

The use of the nonce in the split-key PRF is acceptable in the applications we have considered. In the case of a handshake protocol like TLS 1.3, we get the nonce for free by making use of the nonces already used in the protocol, client random and server random. Neither party can be confident that the nonce they received from their peer is unique, but they can be certain of their own (assuming they have a good random number generator), and in fact the security proofs of TLS 1.3 [DFGS15, DFGS21] start by assuming unique nonces via a birthday bound. For the case of a KEM combiner we can use the ciphertexts and public keys of the component KEMs to create a unique nonce per key pair.

Our contributions. In Section 3 we present a new construction of a split-key PRF using a t-resilient extractor. We provide a detailed security proof in the standard model that the construction is a secure split-key PRF. In Section 4.1 we present a new security notion for hybrid KEMs, which captures the use of KEMs both used individually and in a KEM combiner using potentially the same key material. It gives the adversary against the KEM combiner (unrestricted) access to decapsulation oracles for the ingredient KEMs as well as the combined KEM, but with some post processing of the oracle output in order to prevent trivial wins by the adversary. In Section 4.2 we show that a KEM combiner using a split-key PRF as its core function is secure under the new security notion.

2 Background

In this section, we cover the definitions required for our instantiations of a split-key PRF and a KEM combiner.

2.1 Pseudorandom generators, pseudorandom functions, and more

A pseudorandom generator $G : \mathcal{K} \to \mathcal{Y}$ is a function that takes as input a (typically short) uniformly random key $k \in \mathcal{K}$ and outputs a (typically longer) pseudorandom output $y \in \mathcal{Y}$.

Definition 1 (Pseudorandom generator (PRG))**.** Let $G : \mathcal{K} \to \mathcal{Y}$ be a function. Let $\mathcal{A}$ be an adversary. The pr-security game $\mathsf{Exp}^{\mathsf{pr}}_G(\mathcal{A})$ for G is defined in Figure 1. $\mathcal{A}$'s distinguishing advantage for F is

$$\mathsf{Adv}^{\mathsf{pr}}_G(\mathcal{A}) = 2\left|\Pr\left[\mathsf{Exp}^{\mathsf{pr}}_G(\mathcal{A}) \Rightarrow 1\right] - 1/2\right| \quad .$$

We call G a *secure pseudorandom generator (PRG)* if $\mathsf{Adv}^{\mathsf{pr}}_G(\mathcal{A})$ is small for all practical adversaries $\mathcal{A}$.

A pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a function that takes as input a (typically short) uniformly random key $k \in \mathcal{K}$ and a label or nonce $x \in \mathcal{X}$ and outputs a pseudorandom output $y \in \mathcal{Y}$.

Definition 2 (Pseudorandom function (PRF))**.** Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function. Let $\mathcal{A}$ be an adversary. The pr-security game $\mathsf{Exp}^{\mathsf{pr}}_F(\mathcal{A})$ for F is defined in Figure 1. $\mathcal{A}$'s distinguishing advantage for F is

$$\mathsf{Adv}^{\mathsf{pr}}_F(\mathcal{A}) = 2\left|\Pr\left[\mathsf{Exp}^{\mathsf{pr}}_F(\mathcal{A}) \Rightarrow 1\right] - 1/2\right| \quad .$$

We say that F is a *secure pseudorandom function (PRF)* if $\mathsf{Adv}^{\mathsf{pr}}_F(\mathcal{A})$ is small for all practical adversaries $\mathcal{A}$.

Although a PRF takes in two inputs, we only require that its output is pseudorandom in the first input; that is, we only require that the output is indistinguishable from random

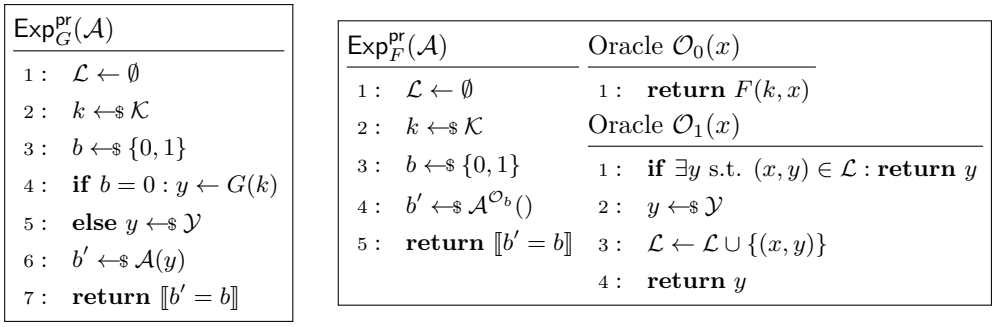

Figure 1: Security experiments for pseudorandomness (pr-security) of (left) a pseudorandom generator $G : \mathcal{K} \to \mathcal{Y}$ and (right) a pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

when the first input is uniform and secret, but make no security guarantees if the first input is known by the adversary. In some applications, such as the TLS 1.3 key schedule [DFGS15, DFGS21], it is desirable for a PRF to be pseudorandom in both inputs; that is, the output of the PRF is indistinguishable from random if either the first or the second input is uniform and secret. Such a function is called a *dual PRF* [Bel06, Bel15].

However, for applications such as KEM combiners, it is desirable to have a function that takes *three* inputs: two keys and label or nonce, and for that function to be pseudorandom as long as either of the two key inputs is uniform and secret. Such a function is called a *split-key PRF* [GHP18], and in general can take n keys and a label. We generalize this slightly to include a public parameter generation algorithm, which is needed for our construction in Section 3.

Definition 3 (Split-key pseudorandom function (skPRF))**.** Let $\mathcal{K}_1, \ldots, \mathcal{K}_n$ be finite key spaces, $\mathcal{X}$ be an input space, $\mathcal{Y}$ be a finite output space, and $\mathcal{P}$ be a finite public parameter space. Let PGen be a (probabilistic) public parameter generation algorithm producing outputs in $\mathcal{P}$, and let $F : \mathcal{P} \times \mathcal{K}_1 \times \cdots \times \mathcal{K}_n \times \mathcal{X} \to \mathcal{Y}$ be a function. Let $\mathcal{A}$ be an adversary. For each index $i \in \{1, \ldots, n\}$, the pr-security game $\mathsf{Exp}^{\mathsf{pr}}_{F,i}(\mathcal{A})$ for F is defined in 2. $\mathcal{A}$'s distinguishing advantage for F is

$$\mathsf{Adv}^{\mathsf{pr}}_{F,i}(\mathcal{A}) = 2\left|\Pr\left[\mathsf{Exp}^{\mathsf{pr}}_{F,i}(\mathcal{A}) \Rightarrow 1\right] - 1/2\right| \quad .$$

We say that F is a *secure split-key pseudorandom function (PRF)* if $\mathsf{Adv}^{\mathsf{pr}}_{F,i}(\mathcal{A})$ is small for all i and all practical adversaries.

Note that the pr security experiment for split-key functions in Figure 2 requires that the adversary use distinct labels x on each oracle query, even if the adversary queries with different keys k'. In our applications, such as KEM combiners, it is straightforward to satisfy this condition by concatenating the public keys and ciphertexts produced by the ingredient KEMs in the execution of the combined KEM. It is still possible to include an unrestricted label in a split-key PRF by adding it as an additional input, or by concatenating it with the nonce.

2.2 t-resilient extractors

Recall that the security definition for split-key pseudorandomness allows the adversary to control the majority of the inputs to the function, but assumes that the remaining input is chosen uniformly at random. So, in order to provide a secure instantiation of a split-key PRF, we need a primitive which takes in a high-entropy source, allows the

$\mathsf{Exp}^{\mathsf{pr}}_{F,i}(\mathcal{A})$	Oracle $\mathcal{O}_0(k', x)$
1 : $\mathcal{L} \leftarrow \emptyset$	1 : **if** $x \in \mathcal{L}$: **return** $\perp$
2 : $\pi \leftarrow\!\!\$\ \mathsf{PGen}()$	2 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{x\}$
3 : $k_i \leftarrow\!\!\$\ \mathcal{K}_i$	3 : parse $(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_n) \leftarrow k'$
4 : $b \leftarrow\!\!\$\ \{0,1\}$	4 : $y \leftarrow F(\pi, k_1, \ldots, k_{i-1}, k_i, k_{i+1}, \ldots, k_n, x)$
5 : $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_b}(\pi)$	5 : **return** y
6 : **return** $[\![b' = b]\!]$	Oracle $\mathcal{O}_1(k', x)$
	1 : **if** $x \in \mathcal{L}$: **return** $\perp$
	2 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{x\}$
	3 : $y \leftarrow\!\!\$\ \mathcal{Y}$
	4 : **return** y

Figure 2: Security experiment for pseudorandomness (pr-security) of a split-key function $F : \mathcal{P} \times \mathcal{K}_1 \times \cdots \times \mathcal{K}_n \times \mathcal{X} \to \mathcal{Y}$

adversary to control most, but not all, of it, and outputs a value indistinguishable from random. This is reminiscent of the scenario of true random number generators (TRNGs) in an adversarial environment, for which Barak et al. [BST03] gave an information theoretic instantiation. They define and give an instantiation of a *t-resilient extractor*, which takes in data from a high-entropy source, and uses an *extractor* to produce a string of truly random bits, crucially allowing the adversary to control t bits of the input. We now recap these definitions.

The notion of min-entropy will be used to capture that a data source is "high entropy".

Definition 4 (Min-entropy)**.** Let X be a probability distribution. The min-entropy of X, denoted by $H_{\min}(X)$, is the maximal number k such that for every $x \in X$, $\Pr[X = x] \leq 2^{-k}$.

Now we can start working towards the definition of a t-resilient extractor. Suppose we have a source with min entropy k which produces n-bit samples. An *extractor* on this source is defined as follows.

Definition 5 (Extractor [BST03])**.** Let S be a set. An *extractor* is a function $E : S \times \{0,1\}^{\ell_E} \to \{0,1\}^{\ell_O}$. For a fixed value $\pi \in S$, we set $E^\pi(x) := E(\pi, x)$.

We are interested in extractors which still guarantee security even if the adversary is able to control t bits of entropy (i.e., the adversary can alter the environment in 2^t different ways). Such an extractor is called *t-resilient*, and this security property is defined as follows.

Definition 6 (t-resilient extractor [BST03])**.** Let S be a set and $t \in \mathbb{N}$. Let $E : S \times \{0,1\}^{\ell_E} \to \{0,1\}^{\ell_O}$ be an extractor. Let $\mathcal{A}$ be an adversary. The security game $\mathsf{Exp}^{t\text{-res}}_E(\mathcal{A})$ is defined in Figure 3. $\mathcal{A}$'s advantage against the t-resilience of E is

$$\mathsf{Adv}^{t\text{-res}}_E(\mathcal{A}) = 2\left|\Pr\left[\mathsf{Exp}^{t\text{-res}}_E(\mathcal{A}) \Rightarrow 1\right] - 1/2\right| \quad .$$

We say that E is an *ϵ-secure t-resilient* if $\mathsf{Adv}^{t\text{-res}}_E(\mathcal{A}) < \epsilon$ for all $\mathcal{A}$.

Barak et al. [BST03] give several constructions of t-resilient extractors. The one we use in our instantiation of a split-key PRF is based on linear functions in the field $GF(2^n)$.

Let $S = \{(a, b) : a, b \in GF(2^\ell_E)\}$, and let $(a, b) \in S$. Suppose we have a source with min-entropy k which produces ℓ_E-bit samples, and let $\ell_O < \ell_E$. If x is sampled from our

$\mathsf{Exp}_E^{t\text{-res}}(\mathcal{A})$

$b \leftarrow\!\!\$\ \{0,1\}$
$(D_1, \ldots, D_{2^t}) \leftarrow\!\!\$\ \mathcal{A}$ s.t. $H_{\min}(D_i) > k$ for all $i \in \{1, \ldots, 2^t\}$
$\pi \leftarrow\!\!\$\ S$ // independently of choices of D_i
$i \leftarrow\!\!\$\ \mathcal{A}(\pi)$
if $b = 0$:
  $X \leftarrow\!\!\$\ D_i$
  $y \leftarrow E^{\pi}(X)$
else $y \leftarrow\!\!\$\ \{0,1\}^{\ell_O}$
$b' \leftarrow\!\!\$\ \mathcal{A}(y)$
return $[\![b = b']\!]$

Figure 3: Security experiment for t-resilience (t-res-security) of extractor $E : S \times \{0,1\}^{\ell_E} \to \{0,1\}^{\ell_O}$

source, then the function $h_{(a,b)}(x) = (a \cdot x + b)_{1,\ldots,m}$ (i.e., the output of $h_{(a,b)}$ is the first m bits of $a \cdot x + b$) is an ϵ-secure t-resilient extractor on public parameter (a, b), where t and ϵ satisfy the relation

$$t = \frac{k - \ell_O}{2} - 2\log(1/\epsilon) - 1. \tag{1}$$

This result follows from a general theorem proved in [BST03]; however, they also note that the distribution of $h_{(a,b)}(x) = (a \cdot x + b)_{1,\ldots,\ell_O}$ is close to uniform if and only if the distribution of $g_a(x) = a \cdot x$ is close to uniform. So, g_a is an ϵ-secure t-resilient extractor as well, with a smaller public parameter and fewer operations than $h_{(a,b)}$. One thing worth noting is that the statistical randomness of this t-resilient extractor is not reliant on any assumptions (cryptographic or otherwise).

2.3 Hybrid key encapsulation mechanisms

A key encapsulation mechanism is a public-key primitive which allows two parties to establish a shared secret.

Definition 7 (Key encapsulation mechanism (KEM))**.** A *key encapsulation mechanism* KEM is a tuple of algorithms $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with shared secret space $\mathcal{K}$:

1. $\mathsf{KGen}() \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$: A probabilistic key generation algorithm that takes in the security parameter as input, and outputs a public/private-key pair.

2. $\mathsf{Encaps}(\mathsf{pk}) \xrightarrow{\$} (k, c)$: A probabilistic encapsulation algorithm that takes in a public key as input, and outputs a shared secret $k \in \mathcal{K}$ and a ciphertext c.

3. $\mathsf{Decaps}(\mathsf{sk}, c) \to k$: A deterministic decapsulation algorithm that takes in a secret key and ciphertext as input, and outputs the corresponding shared secret (or $\perp$, in the case of failure).

Correctness of a KEM requires that, for $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}()$, if $(k, c) \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})$, then $k \leftarrow \mathsf{Decaps}(\mathsf{sk}, c)$ with high probability.

Security of KEMs is modeled as an adversary's inability to distinguish the real shared secret from a random value under a chosen ciphertext attack.

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM}}(\mathcal{A})$	Oracle $\mathcal{O}(c)$
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KEM.KGen}()$	1 : **if** $c = c^* *$: **return** $\perp$
2 : $(k^*, c^*) \leftarrow\!\$\ \mathsf{KEM.Encaps}(\mathsf{pk})$	2 : **return** $\mathsf{KEM.Decaps}(\mathsf{sk}, c)$
3 : $k_0 \leftarrow k^*$	
4 : $k_1 \leftarrow\!\$\ \mathcal{K}$	
5 : $b \leftarrow\!\$\ \{0,1\}$	
6 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}}(\mathsf{pk}, c^*, k_b)$	
7 : **return** $[\![b' = b]\!]$	

Figure 4: Security experiment for ind-cca-security of key encapsulation mechanism KEM

Definition 8 (IND-CCA security for KEMs)**.** Let KEM be a key encapsulation mechanism, and let $\mathcal{A}$ be an adversary. The ind-cca-security game $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM}}(\mathcal{A})$ for KEM is defined in Figure 4. $\mathcal{A}$'s distinguishing advantage for KEM is

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM}}(\mathcal{A}) = 2\left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - 1/2\right| \ .$$

We say that KEM is ind-cca-*secure* if $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM}}(\mathcal{A})$ is small for all practical adversaries $\mathcal{A}$.

With the transition to post-quantum cryptography, it is desirable to have KEMs which are based on both well-tested-but-classical hardness assumptions, as well as post-quantum-but-newer hardness assumptions. This can be done using the notion of a *KEM combiner*, as was introduced in [GHP18]. They proposed using a parallel structure, as defined in Definition 9.

Definition 9 (Parallel KEM Combiner)**.** Let $\mathsf{K}_1, \ldots, \mathsf{K}_n$ be KEMs such that each K_i has public key space $\mathcal{PK}_i$, shared secret space $\mathcal{K}_i$, and ciphertext space $\mathcal{C}_i$. Further, let $\mathcal{K}^* = \mathcal{K}_1 \times \cdots \times \mathcal{K}_n$ and let $\mathcal{C} = \mathcal{PK}_1 \times \cdots \times \mathcal{PK}_n \times \mathcal{C}_1 \times \cdots \times \mathcal{C}_n$, and let $\mathcal{K}$ be an auxiliary finite key space. For any *core function* $W : \mathcal{K}^* \times \mathcal{C} \to \mathcal{K}$, then the *parallel KEM combiner* $\mathsf{K}(\mathsf{K}_1, \ldots, \mathsf{K}_n)$ with respect to W is defined as the tuple of algorithms in Figure 5.

3 New split-key PRF construction

In this section we present a new split-key PRF which we show to be secure in the standard model. Our construction is formulated as a 2-key split-key PRF, but can easily be generalized to the n-key case. The main challenge in constructing a split-key PRF is safely combining the "good" keying material with the "bad" – without knowing which is which. The core idea of our construction is to use a t-resilient extractor which extracts a secret key from a long string that about which the adversary has partial information and partial control. The instantiation we build in Section 3.2 uses a t-resilient extractor that takes inputs that are substantially longer (and have higher entropy) than the target output length. To achieve this longer input, we apply a pseudorandom generator to expand a short key into a longer value with high computational entropy, so that we can then apply the t-resilient extractor.

In summary, the intuition behind our design of a 2-key split-key PRF $M(k_1, k_2, n)$ is as follows:

1. Apply a pseudorandom function F to with key k_1 to label n to produce (short) intermediate keying material x_1; similarly with k_2 and n to produce x_2. Since n is

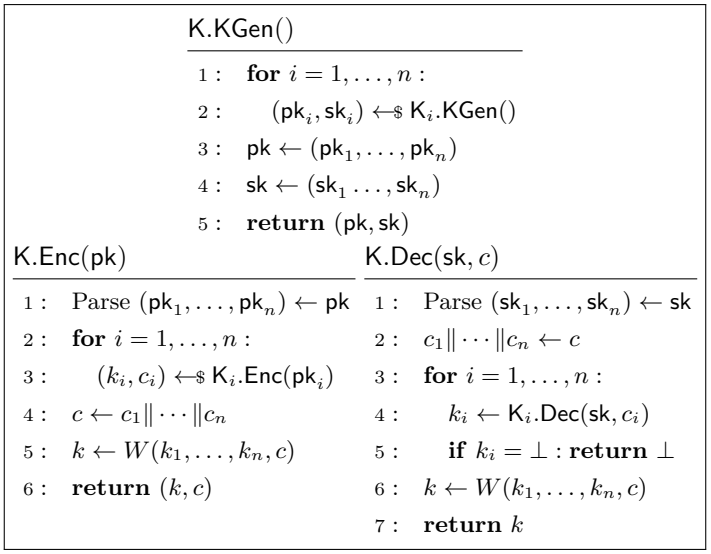

Figure 5: Algorithms for a parallel KEM combiner

a nonce, each call to F with the same key but different nonce will return different output (with high probability).

2. Apply a pseudorandom generator G to (short) intermediate keying material x_1 to produce (long) intermediate keying material y_1; similarly with x_2 to produce y_2.

3. Apply a t-resilient extractor E^π to the concatenation $y_1 \| y_2$ of the long intermediate keying material to produce the final output.

Definition 10 (Our split-key PRF)**.** Let $E^\pi : \{0,1\}^{\ell_E} \to \{0,1\}^{\ell_O}$ be a t-resilient extractor. Let $G : \{0,1\}^{\ell_G} \to \{0,1\}^{\ell_E/2}$ be a pseudorandom generator. Let $F : \{0,1\}^{\ell_F} \times \{0,1\}^* \to \{0,1\}^{\ell_G}$ be a pseudorandom function. Define the split-key PRF $M = M[E^\pi, G, F] : \{0,1\}^{\ell_F} \times \{0,1\}^{\ell_F} \times \{0,1\}^* \to \{0,1\}^{\ell_O}$ as

$$M(k_1, k_2, n) = E^\pi\Big(G\big(F(k_1, n)\big) \Big\| G\big(F(k_2, n)\big)\Big) \ . \tag{2}$$

The parameter generation function PGen for our split-key PRF construction simply samples π from the set S for the t-resilient extractor.

3.1 Security of our split-key PRF

We now show that our split-key PRF is secure in the standard model, assuming a secure pseudorandom function, pseudorandom generator, and t-resilient extractor.

Theorem 1. *Let E^π, G, F, and $M = M[E^\pi, G, F]$ be as in Definition 10. If E^π is a t-res-secure t-resilient extractor, G is a pr-secure pseudorandom generator, and F is a pr-secure pseudorandom function, then M is a pr-secure split-key PRF. More precisely, for every adversary $\mathcal{A}$ against pr-security of M, there exists adversaries $\mathcal{B}_{\mathsf{Ext}}$, $\mathcal{B}_{\mathsf{PRG}}$, and $\mathcal{B}_{\mathsf{PRF}}$ against the pr-securities of E^π, G, and F respectively such that*

$$\mathsf{Adv}^{\mathsf{pr}}_M(\mathcal{A}) \le 2q \times \Big(\mathsf{Adv}^{\mathsf{pr}}_G(\mathcal{B}_{\mathsf{PRG}}) + \mathsf{Adv}^{t\text{-}\mathsf{res}}_E(\mathcal{B}_{\mathsf{Ext}})\Big) + \mathsf{Adv}^{\mathsf{pr}}_F(\mathcal{B}_{\mathsf{PRF}})$$

where the runtimes of $\mathcal{B}_{\mathsf{Ext}}$, $\mathcal{B}_{\mathsf{PRG}}$, and $\mathcal{B}_{\mathsf{PRF}}$ are about the same as the runtime of $\mathcal{A}$, and we have $2^t = 2^1 \times |F$ output space$|$, that is $t = 1 + \ell_G$.

$G^{\mathcal{A}}_{0-3.j}$

$X \leftarrow \emptyset$
$\pi \leftarrow\!\!\$\ \mathsf{PGen}()$
$k_1 \leftarrow\!\!\$\ \{0,1\}^{\ell_F}$ // Game 0
$s \leftarrow 0$ // Game 2-3
$R_F \leftarrow\!\!\$\ \left\{\{0,1\}^* \to \{0,1\}^{\ell_G}\right\}$ // Game 1-2
$R_E \leftarrow\!\!\$\ \left\{\{1,2\} \times \{0,1\}^{\ell_F} \times \{0,1\}^* \to \{0,1\}^{\ell_O}\right\}$ // Game 3
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_b}(\pi)$
return $[\![b' = b]\!]$

$\mathcal{O}_1(k', x)$

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$y \leftarrow\!\!\$\ \{0,1\}^{\ell_O}$
return y

$\mathcal{O}_0(k', x)$ (Game 0)

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$y \leftarrow E^{\pi}\Big(G\big(F(k_1, x)\big) \Big\| G\big(F(k', x)\big)\Big)$
return y

$\mathcal{O}_0(k', x)$ (Game 1)

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$y \leftarrow E^{\pi}\Big(G\big(\mathcal{O}_{\mathsf{PRF}}(x)\big) \Big\| G\big(F(k', x)\big)\Big)$
return y

$\mathcal{O}_0(k', x)$ (Game 2)

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$s \leftarrow s + 1$
if $s \leq j$ **then**
  $z_j \leftarrow\!\!\$\ \{0,1\}^{\ell_E/2}$
  $y \leftarrow E^{\pi}\Big(z_j \Big\| G\big(F(k', x)\big)\Big)$
else $y \leftarrow E^{\pi}\Big(G\big(R_F(x)\big) \Big\| G\big(F(k', x)\big)\Big)$
return y

$\mathcal{O}_0(k', x)$ (Game 3)

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$s \leftarrow s + 1$
if $s \leq j$ **then** $y \leftarrow R_E(i, k', x)$
else
  $z_j \leftarrow\!\!\$\ \{0,1\}^{\ell_E/2}$
  $y \leftarrow E^{\pi}\Big(z_j \Big\| G\big(F(k', x)\big)\Big)$
return y

Figure 6: Games in proof of Theorem 1

Proof. This proof proceeds with two cases, $i \in \{1, 2\}$. These two cases indicate which key is the challenge key and which is provided by the adversary. The rest of the proof shows the case of $i = 1$; the case $i = 2$ is analogous. Figure 6 shows the pseudocode of all games in the proof.

We start by fixing a set of distributions $\mathcal{D} = \{D_{(i,w)} : i \in \{1, 2\}, w \in \mathsf{range}(F)\}$, as:

$$\mathcal{D} = \left\{D_{(1,w)} = \mathcal{U}(\{0,1\}^{\ell_E/2}) \times G(w), D_{(2,w)} = G(w) \times \mathcal{U}(\{0,1\}^{\ell_E/2})\right\} \tag{3}$$

Game 0: This is the pr security experiment for split-key PRF M. Hence,

$$\mathsf{Adv}^{\mathsf{pr}}_{M}(\mathcal{A}) = 2\left|\Pr[G^{\mathcal{A}}_0 \Rightarrow 1] - 1/2\right| \ .$$

Game 1: In this game we replace the pseudorandom function evaluation $F(k_i, n)$ with a random function evaluation $R(n)$ for $R \leftarrow\!\!\$\ \left\{\{0,1\}^* \to \{0,1\}^{\ell_G}\right\}$. The pseudocode for Game 1 is shown in Figure 6.

Claim:

$$\left|\Pr[G^{\mathcal{A}}_0 \Rightarrow 1] - \Pr[G^{\mathcal{A}}_1 \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{pr}}_{F}(\mathcal{B}_{\mathsf{PRF}}) \ .$$

$\mathcal{B}_{\mathsf{PRF}}^{\mathcal{O}_{\mathsf{PRF}},\mathcal{A}}()$

1 : $X \leftarrow \emptyset$
2 : $\pi \leftarrow\!\$\ \mathsf{PGen}()$
3 : $b_{\mathcal{B}_{\mathsf{PRF}}} \leftarrow\!\$\ \{0,1\}$
4 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_b}(\pi)$
5 : **return** $[\![b' = b_{\mathcal{B}_{\mathsf{PRF}}}]\!]$

$\mathcal{O}_0(k', x)$

1 : **if** $x \in X$: **return** $\perp$
2 : $X \leftarrow X \cup \{x\}$
3 : $y \leftarrow E^{\pi}\Big(G\big(\mathcal{O}_{\mathsf{PRF}}(x)\big) \Big\| G\big(F(k', x)\big)\Big)$
4 : **return** y

$\mathcal{O}_1(k', x)$

1 : **if** $x \in X$: **return** $\perp$
2 : $X \leftarrow X \cup \{x\}$
3 : $y \leftarrow\!\$\ \{0,1\}^{\ell_O}$
4 : **return** y

Figure 7: Security reduction for Game 1 in proof of Theorem 1

Reduction: We construct a reduction $\mathcal{B}_{\mathsf{PRF}}$, playing the pr security game for F as follows, shown in Figure 7.

$\mathcal{B}_{\mathsf{PRF}}$ receives queries (k_1, n) from $\mathcal{A}$. When $\mathcal{A}$ is querying $\mathcal{B}_{\mathsf{PRF}}$ with $(k1, n)$, $\mathcal{B}_{\mathsf{PRF}}$ will forward to its PRF challenger the query (n) and receive either $z_{\mathsf{PRF}} \leftarrow R(n)$ for some random function R or $z_{\mathsf{PRF}} \leftarrow \mathsf{PRF}(k, n)$. If the pr experiment for F is in the random case this simulates Game 1. If the pr experiment for F is in the real case, this simulates Game 0.

Game 2 hybrids: In this game we replace the pseudorandom generator G's output with random values. We proceed with a hybrid argument. We show the pseudocode in Figure 6.

Game 2.0 is the same as Game 1, and hence $\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{2.0}^{\mathcal{A}} \Rightarrow 1]$.

Let q be the number of queries $\mathcal{A}$ makes to the $\mathcal{O}_0$ oracle. In game $2.j$, for $j = 1, \ldots, q$, the experiment will replace the evaluation of $G(z_{\mathsf{PRF}})$ for the jth unique call (and all previous calls) to the game's $\mathcal{O}_0$ oracle with a randomly sampled value.

In Game $2.q$, all calls to $G(z_{\mathsf{PRF}})$ have been replaced with a randomly sampled value.

Claim:

$$\Big|\Pr[G_{2.(j-1)}^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{2.j}^{\mathcal{A}} \Rightarrow 1]\Big| \leq \mathsf{Adv}_G^{\mathsf{pr}}(\mathcal{B}_{\mathsf{PRG}}) \ .$$

Reduction: We construct a reduction $\mathcal{B}_{\mathsf{PRG}}$, playing the pr security game for G as follows, shown in Figure 8. $\mathcal{B}_{\mathsf{PRG}}$ receives queries (k_1, n) from $\mathcal{A}$. For the jth unique query from $\mathcal{A}$ to $\mathcal{B}_{\mathsf{PRG}}$, $\mathcal{B}_{\mathsf{PRG}}$ will respond with its challenge z_{PRG} in place of the call $G(R(n))$. If the pr experiment for G is in the real case, this perfectly simulates Game $2.(j-1)$. If the pr experiment for G is in the random case, this perfectly simulates Game $2.j$.

Game 3 hybrids: In this game we replace the evaluation of the extractor E^{π} with a random value. We proceed with a hybrid argument. We show the pseudocode in Figure 6.

Game 3.0 is the same as Game $2.q$, and hence $\Pr[G_{2.q}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.0}^{\mathcal{A}} \Rightarrow 1]$.

Let q be the number of queries $\mathcal{A}$ makes to the $\mathcal{O}_0$ oracle. In Game $3.j$, for $j = 1, \ldots, q$, we replace the jth call (and all previous calls) to the extractor E^{π} with a random function evaluation.

In Game $3.q$, all the calls to E^{π} have been replaced with random function evaluations.

Claim:

$$\Big|\Pr[G_{3.(j-1)}^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{3.j}^{\mathcal{A}} \Rightarrow 1]\Big| \leq \mathsf{Adv}_E^{t\text{-}\mathsf{res}}(\mathcal{B}_{\mathsf{Ext}}) \ .$$

Reduction: We construct a reduction $\mathcal{B}_{\mathsf{Ext}}$, playing the t-res security game for E^{π}, as shown in Figure 9. According to the t-res security experiment, $\mathcal{B}_{\mathsf{Ext}}$ must provide 2^t

$\mathcal{B}^{\mathcal{A}}_{\mathsf{PRG}}(z_{\mathsf{PRG}})$ (Game 2.j)

1 : $X \leftarrow \emptyset$
2 : $\pi \leftarrow\!\$\ \mathsf{PGen}()$
3 : $s \leftarrow 0$
4 : $R_F \leftarrow\!\$\ \left\{\{0,1\}^* \to \{0,1\}^{\ell_G}\right\}$
5 : $b_{\mathcal{B}_{\mathsf{PRG}}} \leftarrow\!\$\ \{0,1\}$
6 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_{b_{\mathcal{B}_{\mathsf{PRG}}}}}(\pi)$
7 : **return** $[\![b' = b_{\mathcal{B}_{\mathsf{PRG}}}]\!]$

$\mathcal{O}_1(k', x)$

1 : **if** $x \in X$: **return** $\perp$
2 : $X \leftarrow X \cup \{x\}$
3 : $y \leftarrow\!\$\ \{0,1\}^{\ell_O}$
4 : **return** y

$\mathcal{O}_0(k', x)$

1 : **if** $x \in X$: **return** $\perp$
2 : $X \leftarrow X \cup \{x\}$
3 : $s \leftarrow s + 1$
4 : **if** $s < j$ **then**
5 : $\quad z_j \leftarrow\!\$\ \{0,1\}^{\ell_E/2}$
6 : $\quad y \leftarrow E^{\pi}\Big(z_j \Big\| G\big(F(k', x)\big)\Big)$
7 : **elseif** $s = j$ **then**
8 : $\quad y \leftarrow E^{\pi}\Big(z_{\mathsf{PRG}} \Big\| G\big(F(k', x)\big)\Big)$
9 : **else** $y \leftarrow E^{\pi}\Big(G\big(R_F(x)\big) \Big\| G\big(F(k', x)\big)\Big)$
10 : **return** y

Figure 8: Security reduction for Game 2 in proof of Theorem 1

distributions to the experiment in advance. These are the distributions $\mathcal{D} = \{D_{(i,w)}\}$ defined in Equation (3). $\mathcal{B}_{\mathsf{Ext}}$ receives the public parameter π from the experiment after choosing the distributions. (Since, in our proof, $\mathcal{B}_{\mathsf{Ext}}$'s choice of distributions is fixed, the order of choosing the distributions and π is irrelevant.) When $\mathcal{B}_{\mathsf{Ext}}$ receives the jth unique query (k'_j, x_j) from $\mathcal{A}$, $\mathcal{B}_{\mathsf{Ext}}$ will choose the distribution for the t-res experiment according to the query, $\mathsf{D}_{i,F(k'_j,x_j)}$. $\mathcal{B}_{\mathsf{Ext}}$ will receive the challenge z_{Ext} and use this as the output value for this query. If $\mathcal{B}_{\mathsf{Ext}}$ is in the real version of its security experiment, then this perfectly simulates Game 3.$(j-1)$. If $\mathcal{B}_{\mathsf{Ext}}$ is in its random game, this perfectly simulates Game 3.j.

In these reductions to the t-res security of E^{π}, the 'honest' portion of the distribution $D_{(1,w)}$ has entropy $\ell_E/2$. This may seem counterintuitive, given that we started from a short honest PRF key of length ℓ_F which was then pseudorandomly expanded to a longer key of length $\ell_E/2$. However, by the end of game 2, all of those pseudorandom expanded keys of length $\ell_E/2$ had been replaced by uniformly random full-entropy keys of length $\ell_E/2$ assuming the security of the PRG. Thus, at the time we applied the reduction to t-res security of E^{π}, we were doing so with half of the input being full entropy.

Analysis of Game 3.q**:** In Game 3.q, the distribution of outputs in the two cases ($b = 0$ or $b = 1$) are equivalent. When $b = 1$, all oracle responses y via $\mathcal{O}_1$ are randomly sampled (see line 3 in $\mathcal{O}_1$). When $b = 0$, all oracle responses y via $\mathcal{O}_0$ are computed from a random function (see line 5 in $\mathcal{O}_0$). Lines 1 and 2 in $\mathcal{O}_0$ prevent there from being repeated calls to the function with the same input. Since there are no repeated calls to the random function, this is equivalent to random sampling every output value, so the two oracles behave identically. Therefore the adversary has zero advantage in Game 3.q.

Case $i = 2$**:** The argument above dealt with the case $i = 1$, where in the 2-key split-key PRF we assume that the first input key is honest and the second input key is adversarial. The case for $i = 2$ is an exact analog of the argument above, notably using the distribution $D_{2,w}$ in Game 3. We therefore end up with a factor of 2 in the advantage bound in the theorem for the second case. □

$\mathcal{B}^{\mathcal{A}}_{\mathsf{Ext}}$ (Game 3.j)

$X \leftarrow \emptyset$
$b_{\mathcal{B}_{\mathsf{Ext}}} \leftarrow\$ \{0,1\}$
$s \leftarrow 0$
$R_E \leftarrow\$ \left\{\{1,2\} \times \{0,1\}^{\ell_F} \times \{0,1\}^* \to \{0,1\}^{\ell_O}\right\}$
send $\mathcal{D}$ (Equation (3)) to t-res challenger
receive π from t-res challenger
$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}_{b_{\mathcal{B}_{\mathsf{Ext}}}}}(\pi)$
return $[\![b' = b_{\mathcal{B}_{\mathsf{Ext}}}]\!]$

$\mathcal{O}_1(k', x)$

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$y \leftarrow\$ \{0,1\}^{\ell_O}$
return y

$\mathcal{O}_0(k', x)$

if $x \in X$: **return** $\perp$
$X \leftarrow X \cup \{x\}$
$s \leftarrow s + 1$
if $s < j$ **then**
  $y \leftarrow R_E(i, k', x)$
elseif $s = j$ **then**
  $m \leftarrow (1, F(k', x))$
  send m to t-res challenger
  receive z_{Ext} from t-res challenger
  $y \leftarrow z_{\mathsf{Ext}}$
else
  $z_j \leftarrow\$ \{0,1\}^{\ell_E/2}$
  $y \leftarrow E^{\pi}\left(z_j \middle\| G\big(F(k', x)\big)\right)$
return y

Figure 9: Security reduction for Game 3 in proof of Theorem 1

3.2 Instantiation

Our construction uses a t-resilient extractor (see Definition 6). We can instantiate this using the construction provided by Barak et al. [BST03], which we now recall. Working in $GF(2^n)$, the field of polynomials over $GF(2)$ modulo an irreducible polynomial of degree n, randomly choose the public parameter $\pi \leftarrow\$ \{0,1\}^n$ and let a be the polynomial in $GF(2^n)$ with coefficients corresponding to the bits of π. Interpret x as a polynomial in $GF(2^n)$ and then we have $E^{\pi}(x) = a \cdot x$ (perform the multiplication in the field) and then take the first m coefficients of the resulting polynomial as the output bits.

[BST03] also specifies the relationship of the adversary control, parameter length and security level by Equation (1). We are expecting output of $m = 256$ bits, for the adversary to control $t = 257$ bits and require the output to be within distance $\epsilon = 2^{-256}$ of uniform. To achieve this, Equation (1) implies we need $\ell_E/2 = k = 1796$ bits and use a public parameter of size $\ell_E = 2k = 3592$ bits.

Our construction also makes use of a PRF and a PRG. For the PRF it would be reasonable to use HMAC, using the key k_i as the key input and the nonce as the message input. The PRG could be reasonably instantiated using AES in CTR mode or a stream cipher.

4 Hybrid IND-CCA security for compound KEMs

4.1 Definitions for compound KEMs

[GHP18] introduce the concept of KEM combiners, which take in a handful of KEMs as input ("ingredient KEMs"), and output a combined KEM. The main idea here is that the combined KEM should be secure so long as at least one of the ingredient KEMs is secure.

Furthermore, they propose a construction for a KEM combiner (Definition 9), and show that it satisfies the usual definition of IND-CCA security for KEMs (under the assumption that at least one of the ingredient KEMs is secure). Specifically, they show that even when an adversary is allowed to query a decapsulation oracle for the combined KEM on any

non-challenge ciphertext, it is infeasible for them to distinguish whether the challenge key is real or random. However, this model does not account for the fact that we are dealing with a system of KEMs, and not just the combined KEM being used in isolation, as it is likely that a user will continue using the ingredient KEMs on their own, as well. We call this a *compound KEM system*.

Definition 11 (Compound KEM system)**.** Let $\mathsf{K}_1, \dots \mathsf{K}_n$ be KEMs, with public key/secret key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_n, \mathsf{sk}_n)$, respectively. Let $\Pi : \mathsf{KEM}^n \to \mathsf{KEM}$ and $\Gamma_1, \dots, \Gamma_n : \mathsf{KEM} \to \mathsf{KEM}$. We say that $\Sigma = (\mathsf{K}_1, \dots, \mathsf{K}_n, \Pi, \Gamma_1, \dots, \Gamma_n)$ is a *compound KEM system*. Let $\Pi.\mathcal{K}$ denote the shared secret space of Π.

In the situation described above, Π is a KEM combiner, and $\Gamma_1, \dots, \Gamma_n$ are post-processing functions that are applied to the ingredient KEMs $\mathsf{K}_1, \dots, \mathsf{K}_n$, respectively. In the simplest case, such as the parallel KEM combiner of [GHP18], $\Gamma_1, \dots, \Gamma_n$ are simply identity functions.

A potential security risk of a compound KEM system is the overlap in the public key of the combined KEM and the public keys of the ingredient KEMs: the component public keys of the combined public key *are* valid public keys for their respective ingredient KEM, and it is possible that a user might re-use an individual algorithm's public key in both a standalone setting and in the compound KEM setting.

A naive solution to model the security of such a scenario is to give the adversary unrestricted access to decapsulation oracles for each of the ingredient KEMs. However, it is easy to show that an adversary with this power could win the security experiment for the combined KEM by simply running the combined KEM algorithm themselves, and querying the ingredient decapsulation oracles to obtain any values that they would otherwise need the secret key to compute.

Thus we need to consider a somewhat restricted security experiment. We could restrict the adversary from querying the decapsulation oracles for each of the ingredient KEMs on parts of the challenge ciphertext, but this is unsatisfying as it just defines away the problem. Our solution is to allow the adversary unrestricted access to oracles for each of the ingredient KEMs, but to perform some post-processing on the outputs of said oracles. The post-processing need not be the same for each oracle, and security depends on the choice of post-processing functions. Although this seems non-standard, it captures real-world situations that the previous definition did not. In particular, the Internet standard for Secure/Multipurpose Internet Mail Extensions (S/MIME) uses this set-up when used with a KEM, as we discuss in Section 4.3.

Formally, our new security definition is as follows. It is worth noting that we implicitly assume that the corresponding public/secret keys for the ingredient KEMs are extractable from the public/secret keys of the combined KEM.

Definition 12 (Hybrid security of a compound KEM system)**.** Let $\Sigma = (\mathsf{K}_1, \dots, \mathsf{K}_n, \Pi, \Gamma_1, \dots, \Gamma_n)$ be a compound KEM system. We say that Σ is $\mathsf{ind\text{-}hycca}$-secure if the adversary's advantage in $\mathsf{Exp}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A})$ is negligible. The experiment is shown in Figure 10.

Under this definition, security of a compound KEM system is not only dependent on the choice of Π, but also the choice of $\Gamma_1, \dots, \Gamma_n$. For instance, if $\Gamma_1, \dots, \Gamma_n$ are all chosen to be the identity function, then Definition 12 is unsatisfiable, as an adversary could simply run the decapsulation algorithm for the combined KEM, and query the ingredient decapsulation oracles every time an ingredient secret key is needed (this is the same situation we described at the beginning of this section). As it turns out, this is a particular instance of a more general case: if there exists a (polynomial time) function $\hat{\Pi}$ such that for all $\mathsf{K}_1, \dots, \mathsf{K}_n$, $\Pi(\mathsf{K}_1, \dots, \mathsf{K}_n) = \hat{\Pi}(\Gamma_1(\mathsf{K}_1), \dots, \Gamma_n(\mathsf{K}_n))$, then Definition 12 is not satisfiable, as an adversary can use the same strategy of running the combined decapsulation algorithm, and querying the ingredient decapsulation oracles any time an

$\mathsf{Exp}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A})$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \Pi(\mathsf{K}_1, \ldots, \mathsf{K}_n).\mathsf{KGen}()$
2 : $(k^*, c^*) \leftarrow\!\$\ \Pi(\mathsf{K}_1, \ldots, \mathsf{K}_n).\mathsf{Encaps}(\mathsf{pk})$
3 : $k_0 \leftarrow k^*$
4 : $k_1 \leftarrow\!\$\ \Pi.\mathcal{K}$
5 : $b \leftarrow\!\$\ \{0,1\}$
6 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_n, \mathcal{O}_{\mathsf{comb}}}(\mathsf{pk}, k_b, c^*)$
7 : **return** $[\![b' = b]\!]$

$\mathcal{O}_i(c)$

1 : $k_i \leftarrow \Gamma_i(\mathsf{K}_i).\mathsf{Decaps}(\mathsf{sk}_i, c)$
2 : **return** k_i

$\mathcal{O}_{\mathsf{comb}}(c)$

1 : **if** $c = c^*$: **return** $\perp$
2 : $k \leftarrow \Pi(\mathsf{K}_1, \ldots, \mathsf{K}_n).\mathsf{Decaps}(\mathsf{sk}, c)$
3 : **return** k

Figure 10: Security experiment for a compound KEM system

ingredient secret key is needed. Furthermore, it is possible in each of these cases for the combined KEM to satisfy Definition 8, so combined KEMs of this form constitute separating examples between the two security definitions.

4.2 Hybrid-secure compound KEM from split-key PRFs

In this section, we show how to construct a compound KEM system that achieves the hybrid-security notion from Definition 12.

Our construction is shown in Figure 11. For the combined KEM Π, the combined shared secret is derived by applying PRFs (domain separated using label @) to the shared secrets produced from each ingredient KEM K_i, and then applying a split-key PRF to the result using a concatenation of the ingredient KEM ciphertexts as the label. Additionally, for each individual KEM $\Gamma(\mathsf{K}_i)$, the shared secret produced by K_i is post-processed by applying the same PRF as above, this time using the relevant ciphertext as the label.

Theorem 2 (Hybrid security of our compound KEM construction)**.** *Let W be a* pr*-secure split-key PRF, and for $i = 1, \ldots, n$, let PRF_i be a* pr*-secure PRF. Let $\mathsf{K}_1, \ldots, \mathsf{K}_n$ be KEMs, at least one of which is* ind-cca*-secure. Then the compound KEM system $\Sigma = (\mathsf{K}_1, \ldots, \mathsf{K}_n, \Pi, \Gamma_1, \ldots, \Gamma_n)$ defined in Figure 11 is* ind-hycca*-secure.*

Specifically, for $\ell = 1, \ldots, n$, we have that for all adversaries $\mathcal{A}$ against the Σ-security of Π, there exist adversaries $\mathcal{B}$ and $\mathcal{B}'$ against the pr*-security of PRF_ℓ, $\mathcal{C}$ and $\mathcal{C}'$ against the* ind-cca*-security of K_ℓ, and $\mathcal{D}$ and $\mathcal{D}'$ against the* pr*-security of W such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A}) \leq\ & \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{B}^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{C}^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{pr}}_{W}(\mathcal{D}^{\mathcal{A}}) \\ & + \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{B}'^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{C}'^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{pr}}_{W}(\mathcal{D}'^{\mathcal{A}}) \ .\end{aligned}$$

Furthermore, if $\mathcal{A}$ calls its oracles $\mathcal{O}_\ell$ and $\mathcal{O}_{comb}$ at most q_ℓ and q_d times respectively, then $\mathcal{B}$ and $\mathcal{B}'$ call their real/random oracles at most $q_\ell + q_d$ times each, $\mathcal{C}$ and $\mathcal{C}'$ call their decapsulation oracles at most $q_\ell + q_d$ times each, and $\mathcal{D}$ and $\mathcal{D}'$ call their real/random oracles at most $q_d + 1$ times each.

Combining all this into a single advantage across all ingredient KEMs gives us that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A}) \leq \min_{\ell=1,\ldots,n} \Big(& \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{B}^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{C}^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{pr}}_{W}(\mathcal{D}^{\mathcal{A}}) \\ & + \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{B}'^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{C}'^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{pr}}_{W}(\mathcal{D}'^{\mathcal{A}}) \Big)\end{aligned}$$

Overview of proof. Our goal is to show that an adversary should not be able to distinguish between being given a real or random key in $\mathsf{Exp}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A})$. To do this, we will perform a series of game hops, starting from $\mathsf{Exp}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A})$ with a real key, and ending with

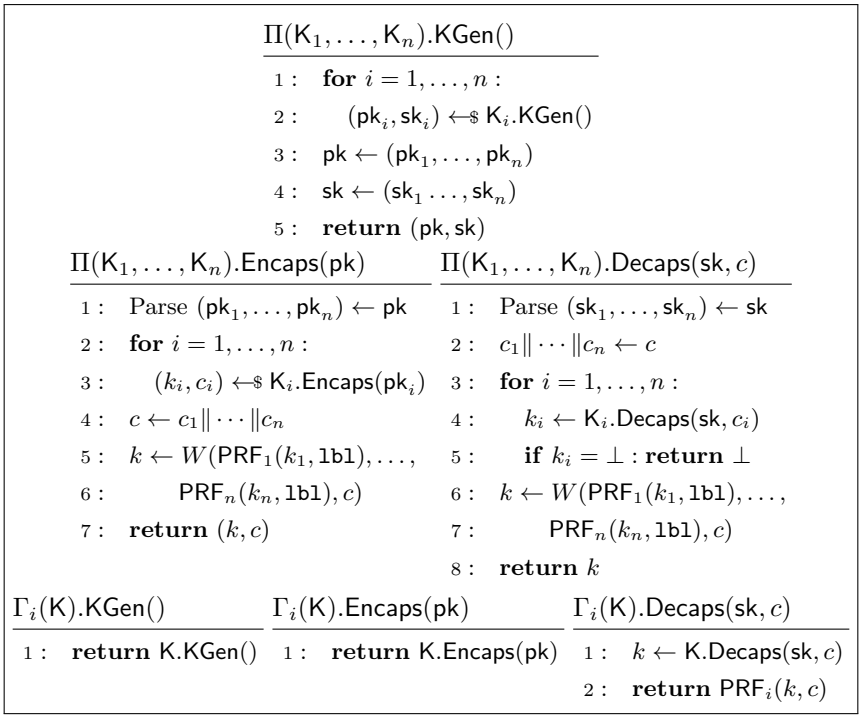

Figure 11: Our compound KEM construction $\Sigma = (\mathsf{K}_1, \ldots, \mathsf{K}_n, \Pi, \Gamma_1, \ldots, \Gamma_n)$, built from a split-key PRF W and PRFs PRF_i

$\mathsf{Exp}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A})$ with a random key, and show that every pair of consecutive games is indistinguishable. In one of our game hops, indistinguishability depends on the security of one of the ingredient KEMs, K_ℓ. The proof is carried through using each of the ingredient KEMs as K_ℓ, and since we bound the adversary's advantage in each case, their overall advantage is bounded by the minimum advantage obtained from using each of the ingredient KEMs. For each ingredient KEM K_ℓ, our game hops (and reductions) are as follows:

- Game 1 is equivalent to using a real key in $\mathsf{Exp}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A})$.
- In Game 2, instead of computing k^*_ℓ (the shared secret from K_ℓ corresponding to sk_ℓ and c^*_ℓ) by decapsulating c^*_ℓ in the oracle $\mathcal{O}_\ell$, we use prior knowledge to do so. We get that
$$\Pr\big[G^{\mathcal{A}}_1 \Rightarrow 1\big] = \Pr\big[G^{\mathcal{A}}_2 \Rightarrow 1\big] \ .$$
- In Game 3, we replace all instances of the honestly-generated key k^*_ℓ with a randomly generated key. We get that
$$\big|\Pr\big[G^{\mathcal{A}}_2 \Rightarrow 1\big] - \Pr\big[G^{\mathcal{A}}_3 \Rightarrow 1\big]\big| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{B}^{\mathcal{A}}) \ .$$
- In Game 4, we replace all instances of $\mathsf{PRF}_\ell(k^*_\ell, \cdot)$ with a truly random function $\mathcal{F}$. We get that
$$\big|\Pr\big[G^{\mathcal{A}}_3 \Rightarrow 1\big] - \Pr\big[G^{\mathcal{A}}_4 \Rightarrow 1\big]\big| \leq \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}}) \ .$$
- In Game 5, we replace the honestly generated challenge key k^* with a randomly chosen key. In order to make the reduction go through, we also add a few lines to

our combined KEM oracle, $\mathcal{O}_{comb}$, which we will remove in the next game hop. We get that

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Adv}_W^{\mathsf{pr}}(\mathcal{D}^{\mathcal{A}}) \ .$$

- In Game 6, we remove the lines added to the combined KEM oracle in previous step. We get that

$$\left|\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Adv}_W^{\mathsf{pr}}(\mathcal{D}^{\mathcal{A}}) \ .$$

- In Game 7, we undo the change that we made in Game 4: we replace all instances of the truly random function $\mathcal{F}$ with $\mathsf{PRF}_\ell(k_\ell^*, \cdot)$. We get that

$$\left|\Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Exp}_{\mathsf{PRF}_\ell}^{\mathsf{pr}}(\mathcal{B}^{\mathcal{A}}) \ .$$

- In Game 8, we undo the change that we made in Game 3: we replace all instances of the randomly generated key k_ℓ^* with an honestly generated key. We get that

$$\left|\Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_8^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}}(\mathcal{C}^{\mathcal{A}}) \ .$$

- Game 8 is equivalent to using a random challenge key in $\mathsf{Exp}_{\Sigma}^{\mathsf{ind\text{-}hycca}}(\mathcal{A})$.

For each $\ell \in \{1, \ldots, n\}$, we have the following series of game hops. Note that since the parameter for the split-key PRF is fixed throughout the experiement, we omit it from the code.
Suppose that K_ℓ is the secure ingredient KEM.

Game 1: This game is equivalent to using a real challenge key in the security experiment, with our Π and Γ functions inlined, where `lbl` is some domain-separated special label.

Game 2: In this game, we modify oracle $\mathcal{O}_\ell$ such that when it is queried on the relevant component of the challenge ciphertext, instead of computing k_ℓ^* by decapsulating c_ℓ^*, it uses prior knowledge of k_ℓ^* to obtain the return value. No reduction is needed for this game hop; this is equivalent to Game 1, under the assumption that KEM K_ℓ satisfies correctness.

Game 3: In this game, we replace the honestly generated key k_ℓ^* with a randomly chosen key. Indistinguishability from Game 2 follows assuming KEM K_ℓ is indistinguishable.

We will write a reduction $\mathcal{B}$, shown in Figure 13, which uses an adversary capable of distinguishing between Game 2 and Game 3 to win the KEM indistinguishability experiment, $\mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}}()$, for K_ℓ. Let $\mathcal{A}$ be an adversary which plays Game 2 and Game 3, and consider reduction $\mathcal{B}$, which acts as a challenger for Game 2 and Game 3, and an adversary for the real/random indistinguishability experiment for K_ℓ (note that $\mathcal{B}$ has access to the decapsulation oracle, $\mathcal{O}_{\mathsf{Dec}}$).

When k_ℓ^* is the real key, then $\mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}}(\mathcal{B}^{\mathcal{A}})$ is identical to Game 2, and when k_ℓ^* is a random key, then $\mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}}(\mathcal{B}^{\mathcal{A}})$ is identical to Game 3 (note that line 1 in $\mathcal{B}_{\mathcal{O}_\ell}$ ensures this, since decapsulation is deterministic). So,

$$\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}-0}(\mathcal{B}^{\mathcal{A}}) \Rightarrow 1\right]$$

and

$$\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}-1}(\mathcal{B}^{\mathcal{A}}) \Rightarrow 1\right] \ .$$

Hence, for any adversary $\mathcal{A}$,

$$\left|\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Adv}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}}(\mathcal{B}^{\mathcal{A}}) \ .$$

Since $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{Dec}}$ only if $\mathcal{A}$ queries either their decapsulation oracle for K_ℓ or their decapsulation oracle for the combined KEM, then $\mathcal{B}$ makes at most $q_\ell + q_d$ queries to $\mathcal{O}_{\mathsf{Decaps}}$.

Games $G_1^{\mathcal{A}}, \ldots, G_7^{\mathcal{A}}$

1 : **for** $i = 1, \ldots, n$:
2 : $\quad (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{K}_i.\mathsf{KGen}()$
3 : $\vec{\mathsf{pk}} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
4 : $\vec{\mathsf{sk}} \leftarrow (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
5 : **for** $i = 1, \ldots, n$:
6 : $\quad (k_i^*, c_i^*) \leftarrow\!\$\ \mathsf{K}_i.\mathsf{Encaps}(\mathsf{pk}_i)$
7 : $k_\ell^* \leftarrow\!\$\ \mathcal{K}_\ell$ // Games 3-7
8 : $\vec{c^*} \leftarrow (c_1^*, \ldots, c_n^*)$
9 : $k^* \leftarrow W(\mathsf{PRF}_1(k_1^*, \mathtt{lbl}),$
10 : $\qquad \ldots, \mathsf{PRF}_\ell(k_\ell^*, \mathtt{lbl}), \ldots,$
11 : $\qquad \mathsf{PRF}_n(k_n^*, \mathtt{lbl}), \vec{c^*})$ // Games 1-3
12 : $k^* \leftarrow W(\mathsf{PRF}_1(k_1^*, \mathtt{lbl}),$
13 : $\qquad \ldots, \mathcal{F}(\mathtt{lbl}), \ldots,$
14 : $\qquad \mathsf{PRF}_n(k_n^*, \mathtt{lbl}), \vec{c^*})$ // Game 4
15 : $k^* \leftarrow\!\$\ \mathcal{K}$ // Games 5-8
16 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_n, \mathcal{O}_{comb}}(\vec{\mathsf{pk}}, k^*, \vec{c^*})$
17 : **return** $[\![b' = b]\!]$

$\mathcal{O}_\ell(c)$

1 : **if** $c = c_\ell^*$:
2 : $\quad$ **return** $\mathsf{PRF}_\ell(k_\ell^*, c)$ // Games 2-3, 7-8
3 : **if** $c = c_\ell^*$:
4 : $\quad$ **return** $\mathcal{F}(c)$ // Games 4-6
5 : $k_\ell \leftarrow \mathsf{K}_\ell.\mathsf{Decaps}(\mathsf{sk}_\ell, c)$
6 : **return** $\mathsf{PRF}_\ell(k_\ell, c)$

$\mathcal{O}_{i=1,\ldots,n,\ i\neq\ell}(c)$

1 : $k_i \leftarrow \mathsf{K}_i.\mathsf{Decaps}(\mathsf{sk}_i, c)$
2 : **return** $\mathsf{PRF}_i(k_i, c)$

$\mathcal{O}_{comb}(\vec{c})$

1 : **if** $\vec{c} = \vec{c^*}$: $\quad$ **return** $\perp$
2 : Parse $(c_1, \ldots, c_n) \leftarrow \vec{c}$
3 : **for** $i = 1, \ldots, n$:
4 : $\quad$ **if** $c_i = c_i^*$: $k_i \leftarrow k_i^*$
5 : $\quad$ **else** : $k_i \leftarrow \mathsf{K}_i.\mathsf{Decaps}(\mathsf{sk}_i, c_i)$
6 : $\quad$ **if** $k_i = \perp$: $\quad$ **return** $\perp$
7 : $k \leftarrow W(\mathsf{PRF}_1(k_1, \mathtt{lbl}), \ldots, \mathsf{PRF}_\ell(k_\ell, \mathtt{lbl}),$
8 : $\qquad \ldots, \mathsf{PRF}_n(k_n, \mathtt{lbl}), \vec{c})$
9 : **if** $c_\ell = c_\ell^*$: // Games 4-6
10 : $\quad k \leftarrow W(\mathsf{PRF}_1(k_1, \mathtt{lbl}), \ldots, \mathcal{F}(\mathtt{lbl}),$
11 : $\qquad \ldots, \mathsf{PRF}_n(k_n, \mathtt{lbl}), \vec{c})$ // Games 4, 6
12 : $\quad k \leftarrow\!\$\ \mathcal{K}$ // Game 5
13 : **return** k

Figure 12: Games in proof of Theorem 2

Game 4: In this game, we replace all instances of $\mathsf{PRF}_\ell(k_\ell^*, \cdot)$ with a truly random function. Indistinguishability from Game 2 follows from the assumption that PRF_ℓ is a secure PRF.

We will write a reduction $\mathcal{C}$, shown in Figure 14, which uses an adversary capable of distinguishing between Game 3 and Game 4 to win the PRF real/random indistinguishability experiment, $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}()$ for PRF_ℓ. Let $\mathcal{A}$ be an adversary which can play Game 3 and Game 4, and consider the reduction $\mathcal{C}$ which acts as a challenger for $\mathcal{A}$, and an adversary for the real/random indistinguishability experiment for PRF_ℓ (note that $\mathcal{C}$ has access to oracle $\mathcal{O}_{\mathsf{PRF}}$, which either returns real or random values).

When $\mathcal{O}_{\mathsf{PRF}}$ is returning values from PRF_ℓ, then $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}})$ is identical to Game 3, and when $\mathcal{O}_{\mathsf{PRF}}$ is returning values from a truly random function, then $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}})$ is identical to Game 4. We use the notation $\mathsf{Exp}^{\mathsf{pr}-0}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}})$ for $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}})$ where the PRF oracle is returning values from PRF_ℓ, and similarly with $\mathsf{pr}-1$ for when the PRF oracle is return random values. Thus

$$\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{pr}-0}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}}) \Rightarrow 1\right]$$

and

$$\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{pr}-1}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}}) \Rightarrow 1\right] ,$$

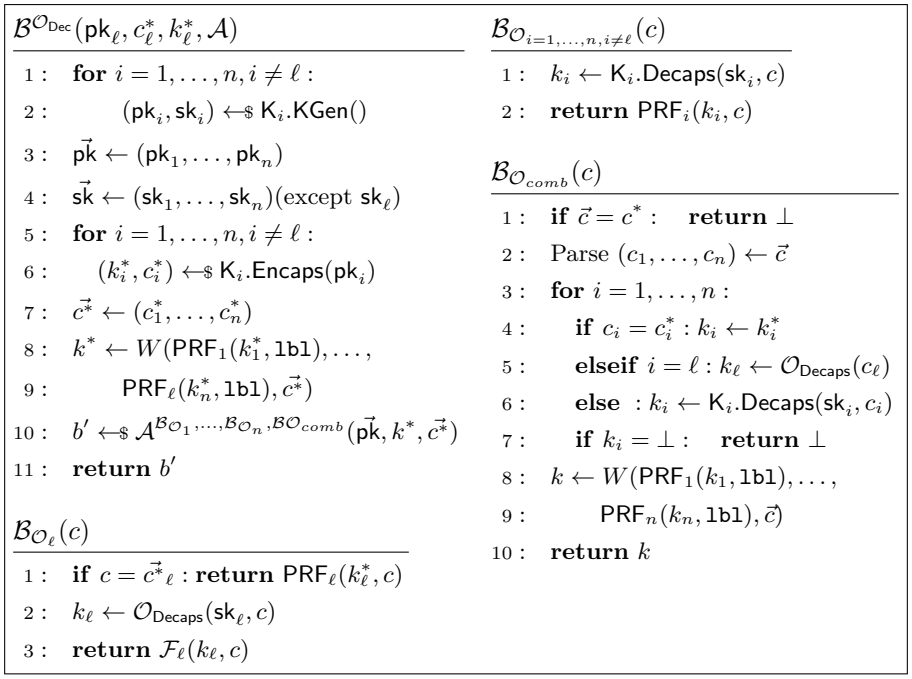

Figure 13: Reduction for Game 3 in the ind-hycca security proof

and hence, for any adversary $\mathcal{A}$,

$$\left|\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{C}^{\mathcal{A}}) \ .$$

Since $\mathcal{C}$ queries $\mathcal{O}_{\mathsf{PRF}}$ only if $\mathcal{A}$ queries their decapsulation oracle for K_ℓ or their decapsulation oracle for the combined KEM, then $\mathcal{C}$ makes at most $q_\ell + q_d$ queries to $\mathcal{O}_{\mathsf{Decaps}}$.

Game 5: In this game, we replace the challenge key k^* obtained using W with a random challenge key, and return a random key from $\mathcal{O}_{comb}$ if the adversary queries $\mathcal{O}_{comb}$ on a ciphertext whose ℓ-th component is c^*_ℓ. Indistinguishability from Game 4 follows from pseudorandomness of the split-key PRF W.

We will write a reduction $\mathcal{D}$, shown in Figure 15, which uses an adversary capable of distinguishing between Game 4 and Game 5 to win the split key PRF real/random indistinguishability experiment, $\mathsf{Exp}^{\mathsf{pr}}_W()$, for the ℓ-th key of skPRF. Let $\mathcal{A}$ be an adversary which plays Game 4 and Game 5, and consider reduction $\mathcal{D}$ which acts as a challenger for Game 4 and Game 5, and an adversary for the PRF real/random indistinguishability of W (note $\mathcal{B}$ has access to oracle $\mathcal{O}_{\mathsf{skPRF}}$, which returns real or random values).

This reduction is identical to the one presented in [GHP18]. When $\mathcal{A}$ makes a query, $\mathcal{D}$ will respond the same way regardless of how $\mathcal{O}_{\mathsf{skPRF}}$ is behaving, unless $\mathcal{A}$ queries the combined KEM oracle on a ciphertext $\vec{c} \neq \vec{c^*}$ such that $\vec{c}_\ell = c^*_\ell$, in which case $\mathcal{A}$ will receive back output from $\mathcal{O}_{\mathsf{skPRF}}$. When $\mathcal{O}_{\mathsf{skPRF}}$ is returning real values, then $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{skPRF}}(\mathcal{D}^{\mathcal{A}})$ is identical to Game 3, and when $\mathcal{O}_{\mathsf{skPRF}}$ returns random values, then $\mathsf{Exp}^{\mathsf{pr}}_{\mathsf{skPRF}}(\mathcal{D}^{\mathcal{A}})$ is identical to Game 4. So,

$$\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{pr}-0}_{\mathsf{skPRF}}(\mathcal{D}^{\mathcal{A}}) \Rightarrow 1\right]$$

and

$$\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{pr}-1}_{\mathsf{skPRF}}(\mathcal{D}^{\mathcal{A}}) \Rightarrow 1\right] \ .$$

$\mathcal{C}^{\mathcal{O}_{\mathsf{PRF}}}(\mathcal{A}^{\mathcal{C}_{\mathcal{O}_1},\ldots,\mathcal{C}_{\mathcal{O}_n},\mathcal{C}_{\mathcal{O}_{comb}}})$

$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{K}_i.\mathsf{KGen}()$
$\vec{\mathsf{pk}} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
$\vec{\mathsf{sk}} \leftarrow (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
for $i = 1, \ldots, n$:
  $(k_i^*, c_i^*) \leftarrow\!\$\ \mathsf{K}_i.\mathsf{Encaps}(\mathsf{pk}_i)$
$k_\ell^* \leftarrow\!\$\ \mathcal{K}_\ell$
$\vec{c^*} \leftarrow (c_1^*, \ldots, c_n^*)$
$k^* \leftarrow W(\mathsf{PRF}_1(k_1^*, \mathtt{lbl}), \ldots,$
  $\mathcal{O}_{\mathsf{PRF}}(\mathtt{lbl}), \ldots, \mathsf{PRF}_n(k_n^*, \mathtt{lbl}), \vec{c^*})$
$b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n,\mathcal{O}_{comb}}(\vec{\mathsf{pk}}, k^*, \vec{c^*})$
return $[\![b' = b]\!]$

$\mathcal{C}_{\mathcal{O}_\ell}(c)$

if $c = c_\ell^*$: **return** $\mathcal{O}_{\mathsf{PRF}}(c)$
$k_\ell \leftarrow \mathsf{K}_\ell.\mathsf{Decaps}(\mathsf{sk}_\ell, c)$
return $\mathsf{PRF}_\ell(k_\ell, c)$

$\mathcal{C}_{\mathcal{O}_{i=1,\ldots,n,i\neq\ell}}(c)$

$k_i \leftarrow \mathsf{K}_i.\mathsf{Decaps}(\mathsf{sk}_i, c)$
return $\mathsf{PRF}_i(k_i, c)$

$\mathcal{C}_{\mathcal{O}_{comb}}(c)$

if $\vec{c} = \vec{c^*}$: **return** $\perp$
Parse $(c_1, \ldots, c_n) \leftarrow \vec{c}$
for $i = 1, \ldots, n$:
  if $c_i = \vec{c^*}_i$: $k_i \leftarrow k_i^*$
  else : $k_i \leftarrow \mathsf{K}_i.\mathsf{Decaps}(\mathsf{sk}_i, c_i)$
  if $k_i = \perp$: **return** $\perp$
if $i = \ell$:
  $k \leftarrow W(\mathsf{PRF}_1(k_1, \mathtt{lbl}), \ldots,$
    $\mathcal{O}_{\mathsf{PRF}}(\mathtt{lbl}), \ldots, \mathsf{PRF}_n(k_n, \mathtt{lbl}), \vec{c})$
else :
  $k \leftarrow W(\mathsf{PRF}_1(k_1, @), \ldots,$
    $\mathsf{PRF}_n(k_n, @), \vec{c})$
return k

Figure 14: Reduction for Game 4 in the ind-hycca security proof

Hence, for any adversary $\mathcal{A}$,

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Adv}_W^{\mathsf{pr}}(\mathcal{D}^{\mathcal{A}}) \ .$$

Since $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{skPRF}}$ once at the beginning, and then only if $\mathcal{A}$ queries their decapsulation oracle for the combined KEM, $\mathcal{D}$ makes at most $q_d + 1$ queries to $\mathcal{O}_{\mathsf{PRF}}$.

Game 6: In this game, we get rid of the lines added to $\mathcal{O}_{comb}$ in Game 5. Indistinguishability from Game 5 follows from the split-key PRF assumption on W. Just as in the proof presented in [GHP18], we can use an almost identical reduction to $\mathcal{D}$, which we will call $\mathcal{D}'$. This gives us that for any adversary $\mathcal{A}$,

$$\left|\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Adv}_W^{\mathsf{pr}}(\mathcal{D}'^{\mathcal{A}})$$

and similarly to the previous step, our reduction makes at most $q_d + 1$ queries to $\mathcal{O}_{\mathsf{skPRF}}$.

Game 7: In this game, we replace the truly random function $\mathcal{F}$ with the original pseudorandom function, PRF_ℓ. The reduction here, which we will call $\mathcal{C}'$ is almost identical to the reduction $\mathcal{C}$ for distinguishing between Game 3 and Game 4. This gives us that

$$\left|\Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Exp}_{\mathsf{PRF}_\ell}^{\mathsf{pr}}(\mathcal{B}^{\mathcal{A}})$$

and once again our reduction makes at most q_ℓ queries to $\mathcal{O}_{\mathsf{PRF}}$.

Game 8: In this game, we switch back from using a random k_ℓ^* to using an honestly generated k_ℓ^*. Indistinguishability from Game 7 follows from the assumption that K_ℓ satisfies KEM indistinguishability. The reduction here, which we will call $\mathcal{B}'$, is almost identical to the reduction $\mathcal{B}$ for distinguishing between Game 2 and Game 3. This gives us that

$$\left|\Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_8^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathsf{Exp}_{\mathsf{K}_\ell}^{\mathsf{ind\text{-}cca}}(\mathcal{B}'^{\mathcal{A}})$$

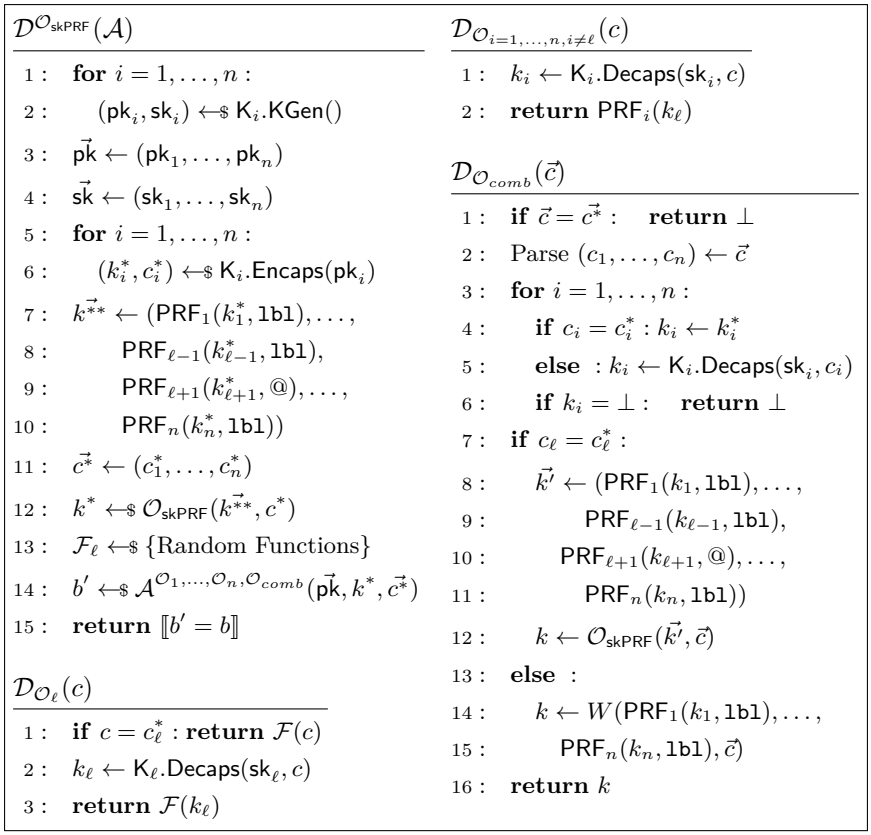

Figure 15: Reduction for Game 5 in the ind-hycca security proof

and once again our reduction makes at most q_ℓ queries to $\mathcal{O}_{\mathsf{Dec}}$.

Analysis of Game 8: Notice that Game 8 is equivalent to using a random challenge key in our original ind-hycca security experiment. So, combining the advantages from each of the games gives us that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}hycca}}_{\Sigma}(\mathcal{A}) \leq\ & \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{B}^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{C}^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{pr}}_{W}(\mathcal{D}^{\mathcal{A}}) \\ & + \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{PRF}_\ell}(\mathcal{B}'^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{K}_\ell}(\mathcal{C}'^{\mathcal{A}}) + \mathsf{Adv}^{\mathsf{pr}}_{W}(\mathcal{D}'^{\mathcal{A}}) \ .\end{aligned}$$

4.3 Application to S/MIME

Multipurpose Internet Mail Extensions (MIME) is an Internet standard specified in RFC 2045 [FB96b], RFC 2046 [FB96c], RFC 2047 [Moo96], RFC 4288 [Moo96], RFC 4289 [KF05], and RFC 2049 [FB96a], which extends the format of email messages and allows them to support various non-ASCII characters, as well as attachments. Secure/Multipurpose Internet Mail Extensions (S/MIME) is the Internet standard which provides authentication, message integrity, non-repudiation of origin, and confidentiality of MIME data (RFC 8551 [SRT19]), in accordance with the Cryptographic Message Syntax (CMS) Internet standard. CMS has always supported key agreement algorithms, and more recently, it has also been extended to support KEMs in RFC 9629 [HGO24].

As a very general overview, S/MIME with a KEM works as follows. Suppose an originator wants to send MIME data to a recipient who has published their public key pk for a KEM K. First, the originator will randomly generate a content-encryption key $\mathtt{cek}$, which they will then use to encrypt the MIME data using a symmetric key

encryption scheme. Then, they will generate a shared secret and ciphertext by running $(k, c) \leftarrow\!\!\$\ \mathsf{K.Encaps}(\mathsf{pk})$. Using a key derivation function on k (and, optionally, additional data present in the MIME data), the originator will then compute a pairwise symmetric key-encryption key `kek`, which they will then use to encrypt `cek`. Finally, they will send the encrypted MIME data, the encrypted `cek`, and c to the recipient, who can then recover the MIME data by running $k \leftarrow \mathsf{K.Decaps}(\mathsf{sk}, c)$, applying the key derivation function to k to get `kek`, using `kek` to decrypt `cek`, and finally, using `cek` to decrypt the MIME data.

Consider our use case for a combined KEM: a situation where a recipient has public/secret key pairs for KEMs $\mathsf{K}_1, \ldots, \mathsf{K}_n$, where K_n is a combined KEM which uses $\mathsf{K}_1, \ldots, \mathsf{K}_{n-1}$ as ingredient KEMs. In this case, an adversary observing network traffic would have access to all the data sent between any originator and the recipient; namely, the encrypted `cek` and the encrypted MIME data. For each S/MIME instance, these values are encrypted using the `kek` and the `cek` for that S/MIME instance, respectively, so the adversary would not have (immediate) access to anything encrypted using the shared secret k for that S/MIME instance. For this reason, when modeling this as a security experiment, it is more realistic to give the adversary a `kek`-oracle for each S/MIME instance, rather than a k-oracle. This aligns with our approach in Definition 12. In this case, Π is the KEM combiner used to construct K_n, and Γ is the key derivation function that is applied to k to get `kek` for each S/MIME instance.

Acknowledgements

The authors are grateful for helpful discussions with Daniel Apon, Jonathan Katz, and Mike Ounsworth. L.M. was supported by the Research Council of Norway under Project No. 288545. D.S. was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2022-03187 and NSERC Alliance grant ALLRP 578463-22. C.S. was supported by NSERC Alexander Graham Bell Canada Graduate Scholarship - Doctoral.

References

[BBGS23] Matilda Backendal, Mihir Bellare, Felix Günther, and Matteo Scarlata. When messages are keys: Is HMAC a dual-PRF? In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 661–693. Springer, Cham, August 2023. doi:10.1007/978-3-031-38548-3_22.

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Berlin, Heidelberg, August 2006. doi:10.1007/11818175_36.

[Bel15] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision resistance. *Journal of Cryptology*, 28(4):844–878, October 2015. doi:10.1007/s00145-014-9185-x.

[BFGM23] Colin Boyd, Elsie Mestl Fondevik, Kristian Gjøsteen, and Lise Millerjord. Hybrid group key exchange with application to constrained networks. In Elias Athanasopoulos and Bart Mennink, editors, *Information Security (ISC) 2023*, volume 14411 of *LNCS*, pages 455–472. Springer, 2023. doi:10.1007/978-3-031-49187-0_23.

[BST03] Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In Colin D. Walter, Çetin Kaya Koç, and

Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 166–180. Springer, Berlin, Heidelberg, September 2003. doi:10.1007/978-3-540-45238-6_14.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_15.

[DFGS15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1197–1210. ACM Press, October 2015. doi:10.1145/2810103.2813653.

[DFGS21] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol. *Journal of Cryptology*, 34(4):37, October 2021. doi:10.1007/s00145-021-09384-1.

[EB20] Richard Davis Elaine Barker, Lily Chen. Recommendation for key-derivation methods in key-establishment schemes. Technical Report SP 800-56C Rev. 2, National Institute for Standards in Technology, August 2020. doi:10.6028/NIST.SP.800-56Cr2.

[FB96a] Ned Freed and Dr. Nathaniel S. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples. RFC 2049, November 1996. URL: https://www.rfc-editor.org/info/rfc2049, doi:10.17487/RFC2049.

[FB96b] Ned Freed and Dr. Nathaniel S. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, November 1996. URL: https://www.rfc-editor.org/info/rfc2045, doi:10.17487/RFC2045.

[FB96c] Ned Freed and Dr. Nathaniel S. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046, November 1996. URL: https://www.rfc-editor.org/info/rfc2046, doi:10.17487/RFC2046.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 190–218. Springer, Cham, March 2018. doi:10.1007/978-3-319-76578-5_7.

[HGO24] Russ Housley, John Gray, and Tomofumi Okubo. Using Key Encapsulation Mechanism (KEM) Algorithms in the Cryptographic Message Syntax (CMS). RFC 9629, August 2024. URL: https://www.rfc-editor.org/info/rfc9629, doi:10.17487/RFC9629.

[KF05] Dr. John C. Klensin and Ned Freed. Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures. RFC 4289, December 2005. URL: https://www.rfc-editor.org/info/rfc4289, doi:10.17487/RFC4289.

[MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 448–471. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_16.

[Moo96] Keith Moore. MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. RFC 2047, November 1996. URL: https://www.rfc-editor.org/info/rfc2047, doi:10.17487/RFC2047.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: https://www.rfc-editor.org/info/rfc8446, doi:10.17487/RFC8446.

[SRT19] Jim Schaad, Blake C. Ramsdell, and Sean Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. RFC 8551, April 2019. URL: https://www.rfc-editor.org/info/rfc8551, doi:10.17487/RFC8551.