Security of Hybrid Key Encapsulation

Matthew Campagna and Adam Petcher

Amazon Web Services
campagna@amazon.com, apetcher@amazon.com

Abstract. In a hybrid key encapsulation construction, multiple independent key encapsulation mechanisms are combined in a way that ensures the resulting key is secure according to the strongest mechanism. Such constructions can combine mechanisms that are secure in different settings and achieve the combined security of all mechanisms. For example classical and post-quantum mechanisms can be combined in order to secure communication against current threats as well as future quantum adversaries. This paper contains proofs of security for two hybrid key encapsulation mechanisms along with the relevant security definitions. Practical interpretation of these results is also provided in order to guide the use of these mechanisms in applications and standards.

Keywords: key encapsulation · post quantum

1 Introduction

Widespread methods of session key establishment, such as elliptic curve Diffie-Hellman, are insecure against quantum adversaries that may exist in the future. This quantum threat has motivated the development of numerous key encapsulation mechanisms (KEMs) that are conjectured to be secure against quantum computers. A KEM is a protocol that uses public key cryptography to establish a shared secret between two or more parties. A pressing concern is that encrypted communication could be recorded today and then decrypted in the future after such quantum computers have been developed. This concern could be addressed by using quantum-secure KEMs today, but there is some risk that using these novel cryptosystems could weaken security against existing classical adversaries.

Hybrid KEMs[6][3] can be used to enable security against future quantum adversaries while retaining security against current adversaries. In a hybrid KEM, multiple KEMs are composed such that the security of the composite is determined by the security of the strongest KEM. Moreover, if the component KEMs are secure against different classes of adversary, then the hybrid KEM is secure against all such adversaries. The challenge in developing a hybrid KEM is that one or more component KEM is allowed to be completely insecure against some class of adversary, and it is necessary to ensure that the adversary cannot leverage this insecurity to defeat the hybrid KEM.

This work examines two hybrid KEM combiners that accept the shared secrets produced by multiple KEMs and produce a single shared secret. The first

combiner concatenates the secrets before giving them to a key derivation function (KDF). The second combiner executes the KDF multiple times in a cascade in which one KEM secret is provided in each iteration. Both hybrid KEMs are proved secure when the KDF is modeled as a random oracle and one input KEM is assumed to be one-way against chosen plaintext attack (OW-CPA). The OW-CPA security definition ensures that no efficient adversary that knows the public information produced by the KEM is able to guess the whole shared secret. The concatenation-based KEM is also proved secure under the assumption that the KDF is secure and that at least one KEM is secure according to indistinguishability under chosen plaintext attack (IND-CPA). The KDF security definition states that the KDF outputs are indistinguishable from random as long as the input keying material is drawn from an appropriate source. An IND-CPA KEM is one in which any efficient adversary is unable to distinguish the resulting secret from a random value, even after learning all of the public information produced by the KEM.

The security proofs ensure that these hybrid KEMs retain the security of the underlying KEMs, and that the composition is secure even when one or more KEM is insecure. All theorems include concrete bounds to provide insight into the security of each hybrid KEM when deployed at scale. In order to rule out trivial proof errors, the proofs are mechanically checked by the Coq proof assistant[4] using the Foundational Cryptography Framework[10].

2 Security Definitions

2.1 Key Encapsulation Mechanisms

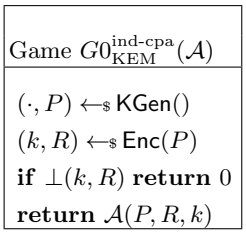
A key encapsulation mechanism is a tuple of probabilistic algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

- KGen is a key generation algorithm that produces a pair (sk, P), where sk is a secret key and P is a public key.
- Enc is an encapsulation algorithm that takes a public key produced by KGen. Enc produces a pair (k, R), where k is a shared secret and R is a public encapsulation.
- Dec is a decapsulation algorithm that takes a secret key produced by KGen and a public encapsulation produced by Enc. Dec produces the shared secret k

The Enc and Dec algorithms may fail to produce a valid result, and the $\perp$ function is used in definitions to test for this failure. The symbol $\perp$ is also used to describe a constant value x for which $\perp(x) = 1$.

A key encapsulation is carried out in practice between two parties, Alice and Bob. Alice runs KGen to produce (sk, P) and sends P to Bob. Bob runs $\mathsf{Enc}(P)$ to produce (k, R) and sends R to Alice. Alice runs $\mathsf{Dec}(sk, R)$ to obtain k. At the conclusion of this protocol, k is a shared secret known to both parties. To prevent man-in-the-middle attacks, the values P and R must be sent over authenticated channels.

Indistinguishability under Chosen Plaintext Attack A KEM achieves indistinguishability under chosen plaintext attack (IND-CPA) against adversary $\mathcal{A}$ if $\mathcal{A}$ cannot distinguish the shared secret from a random shared secret, except with acceptably small probability. $\mathcal{A}$ is given all of the public information produced by the KEM. This definition is parameterized by a distribution S that describes a secure source of shared secrets. Note: A KEM has no plaintext, and the word "plaintext" only appears in this definition due to the fact that it is derived from similar definitions used for encryption.



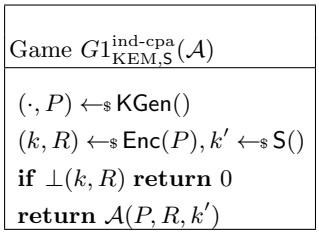


Fig. 1: IND-CPA games

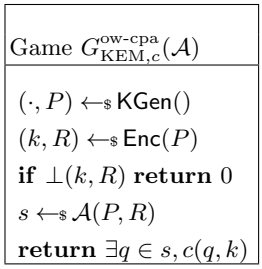


Fig. 2: OW-CPA game

Definition 1 (IND-CPA Advantage). *Let KEM be a key encapsulation mechanism and $\mathcal{A}$ be an algorithm, then the advantage of $\mathcal{A}$ against IND-CPA of KEM w.r.t. source S is*

$$\mathsf{Adv}^{ind\text{-}cpa}_{KEM,\mathsf{S}}(\mathcal{A}) = \left| \Pr\left[G0^{ind\text{-}cpa}_{KEM}(\mathcal{A}) = 1\right] - \Pr\left[G1^{ind\text{-}cpa}_{KEM,\mathsf{S}}(\mathcal{A}) = 1\right] \right|$$

One Way under Chosen Plaintext Attack A KEM is one-way under chosen plaintext attack (OW-CPA) against an adversary $\mathcal{A}$ if $\mathcal{A}$ can only produce the resulting shared secret with acceptably small probability. In this definition, $\mathcal{A}$ is given all of the public information produced by the KEM and produces a list of values. The adversary wins if the shared secret is contained in any element in this list. The definition is parameterized by a function c that determines whether the secret is contained within some value. For example, the adversary may win the game if the list includes a larger string that contains the shared secret as a substring.

Definition 2 (OW-CPA Advantage). *Let KEM be a key encapsulation mechanism and $\mathcal{A}$ be an algorithm, then the advantage of $\mathcal{A}$ against OW-CPA of KEM with function c is*

$$\mathsf{Adv}^{ow\text{-}cpa}_{KEM,c}(\mathcal{A}) = \Pr\left[G^{ow\text{-}cpa}_{KEM,c}(\mathcal{A}) = 1\right]$$

Key Reuse IND-CPA security and OW-CPA security do not imply that the KEM result k is secret when the values produced by KGen are reused in multiple protocol instances. That is, a new sk value must be produced for each protocol instance, and it must be discarded after the shared secret is produced. If key reuse is desired, then the KEM must be secure according to a stronger notion such as IND-CCA, in which the adversary is given access to a Dec oracle for sk.

2.2 Key Derivation Functions

A key derivation function (KDF) is a function on four arguments KDF(s, r, c, ℓ), where s is the keying material, r is salt, c is any additional information information (often associated with the context), and ℓ is the desired output key material length.

Secure Key Derivation Function To prove the security of the hybrid key exchange mechanisms, we can use a weaker form of the secure key derivation function definition from [9], in which the adversary is not given access to a KDF oracle. The definition is also modified to allow the source S to fail to produce a value.

Game $G0^{\text{kdf-weak}}_{\text{KDF}}(\mathcal{A})$
$(s, a) \leftarrow\!\!\$\ \mathsf{S}()$
if $\perp(s, a)$ **return** 0
$r \leftarrow\!\!\$\ \text{Salt}()$
$(c, \ell) \leftarrow\!\!\$\ \mathcal{A}(a, r)$
$o \leftarrow \text{KDF}(s, r, c, \ell)$
return $\mathcal{A}(o)$

Game $G1^{\text{kdf-weak}}_{\text{KDF}}(\mathcal{A})$
$(s, a) \leftarrow\!\!\$\ \mathsf{S}()$
if $\perp(s, a)$ **return** 0
$r \leftarrow\!\!\$\ \text{Salt}()$
$(c, \ell) \leftarrow\!\!\$\ \mathcal{A}(a, r)$
$o \leftarrow\!\!\$\ \{0,1\}^{\ell}$
return $\mathcal{A}(o)$

Fig. 3: KDF weak security games

Definition 3 (Weakly Secure KDF Advantage). *Let KDF be a function,* S *be a source of keying material, Salt be a source of salt, and* $\mathcal{A}$ *be an algorithm, then the advantage of* $\mathcal{A}$ *against weak security of KDF when extracting from* S *using Salt is*

$$\mathsf{Adv}^{kdf\text{-}weak}_{KDF,\mathsf{S},Salt}(\mathcal{A}) = \left| \Pr\left[G0^{kdf\text{-}weak}_{KDF,\mathsf{S},Salt}(\mathcal{A}) = 1\right] - \Pr\left[G1^{kdf\text{-}weak}_{KDF,\mathsf{S},Salt}(\mathcal{A}) = 1\right] \right|$$

Key Derivation Functions as Random Oracles In some of the proofs in this paper, the KDF is modeled as a random oracle. More precisely, the KDF

is modeled as a family of distinct random oracles for each length value ℓ, where each random oracle takes a query tuple (s, r, c) and returns a random bit string of length ℓ if the query is not entirely equal to a previous query value.

3 Constructions

A hybrid key exchange scheme is composed of key exchange mechanisms KEM_i for $i \in 1 \dots n$. The components of KEM_i are $(\mathsf{KGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$. The scheme also includes a combiner $\mathcal{C}$ that takes the information produced from each KEM along with some context and label information and an optional pre-shared key. The combiner produces a secure shared secret of the desired length ℓ. If any Enc_i fails to produce a value, then the hybrid scheme fails. The process used by the responder to produce a shared secret is shown in Figure 4.

Hybrid KEM(P, context, label, ℓ, psk)

for $i = 1 \dots n$ **do**
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** $\perp$
return $\mathcal{C}(\text{context}, \text{label}, \ell, \text{psk}, k, P, R)$

Fig. 4: Hybrid KEM

3.1 CtKDF

The concatenation KDF (CtKDF) combiner produces an intermediate secret by concatenating the shared secrets produced by the KEMs. This intermediate secret is given to the KDF along with a context that is derived from all of the public information produced by the KEMs using a formatting function f. The output of the KDF is the shared secret produced by the hybrid KEM.

CtKDF(context, label, ℓ, psk, k, P, R)

secret $\leftarrow \text{psk} \| k_1 \| k_2 \| \dots \| k_n$
context' $\leftarrow f(\text{context}, P, R)$
return $\text{KDF}(\text{secret}, \text{label}, \text{context'}, \ell)$

Fig. 5: CtKDF combiner

CasKDF(context, label, ℓ, psk, k, P, R)

$c_0 = \text{psk}$
for $i = 1 \dots n$ **do**
  secret $\leftarrow (c_{i-1}, k_i)$
  context' $\leftarrow (\text{context}_i, P_i, R_i)$
  $c_i \| r_i \leftarrow \text{KDF}(\text{secret}, \text{label}_i, \text{context'}, d + \ell)$
return r

Fig. 6: CasKDF combiner

3.2 CasKDF

The cascade KDF (CasKDF) combiner produces a shared secret using a cascade that accepts a single secret in each iteration. The combiner accepts a value ℓ that determines the length of the output key, and there is an additional length value d that determines the length of an intermediate secret used in the cascade. The context and label values are lists of length n, where context_i and label_i are used in iteration i of the cascade. The cascade produces n values, where only the last value is a secret, and the first $n-1$ values are computationally independent of the secret.

4 Security Proofs

4.1 CtKDF

Security proofs for CtKDF are provided in both the random oracle model and in the standard model.

Random Oracle Model In the random oracle model, CtKDF is IND-CPA secure as long as at least one KEM is OW-CPA secure. In this setting, the KDF is modeled as a random oracle as described in Section 2.2. The component KEM schemes may not query the random oracle.

Theorem 1 (CtKDF Security in the Random Oracle Model). *For any $m \leq n$ and any algorithm $\mathcal{A}$ the advantage of $\mathcal{A}$ against IND-CPA of CtKDF is*

$$\mathsf{Adv}^{ind\text{-}cpa}_{CtKDF,\{0,1\}^\ell}(\mathcal{A}) \leq \mathsf{Adv}^{ow\text{-}cpa}_{KEM_m,c}(\mathcal{B}(\mathcal{A},m))$$

where c is a function that takes a query and a secret, and returns true iff the whole secret appears anywhere in the query, and $\mathcal{B}$ is defined in Figure 7.

In Figure 7, $\mathcal{A}^{\mathcal{O}}$ denotes the interaction between $\mathcal{A}$ and the random oracle $\mathcal{O}$. The procedure returns the list of distinct queries that were produced during this interaction.

Proof. The only difference between $G0$ and $G1$ in $\mathsf{Adv}^{\text{ind-cpa}}_{\text{CtKDF},\{0,1\}^\ell}(\mathcal{A})$ is that $G1$ gives an independent random value to $\mathcal{A}$ instead of the value produced by CtKDF. This

$\mathcal{B}(\mathcal{A},\ m,\ \hat{P},\ \hat{R})$

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i(), (\cdot, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i(), (\cdot, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
$P_m \leftarrow \hat{P}, R_m \leftarrow \hat{R}, k \leftarrow\!\!\$\ \{0,1\}^\ell$
$\mathcal{A}^{\mathcal{O}}(P, R, k)$
return queries($\mathcal{A}^{\mathcal{O}}$)

Fig. 7: Constructed adversary against OW-CPA in CtKDF random oracle model security proof

value has the same distribution as the one produced by the combiner, and the adversary can only distinguish these values by querying the random oracle on the appropriate value. This oracle query includes the shared secret produced by KEM_m as a substring, so $\mathcal{A}$ will only produce this query value if $\mathcal{B}$ wins the OW-CPA game for KEM_m. □

Standard Model In the standard model, CtKDF is IND-CPA secure as long as at least one KEM is IND-CPA secure and the KDF is a weakly secure key derivation function for the appropriate source of key material.

Theorem 2 (CtKDF Security in the Standard Model). *For any $m \leq n$ and any algorithm $\mathcal{A}$ the advantage of $\mathcal{A}$ against IND-CPA of CtKDF is*

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\textit{CtKDF},\{0,1\}^\ell}(\mathcal{A}) \leq \mathsf{Adv}^{\textit{ind-cpa}}_{\textit{KEM}_m,S_m}(\mathcal{B}(\mathcal{A},m)) + \mathsf{Adv}^{\textit{kdf-weak}}_{\textit{KDF},S^+(S_m,m),\textit{label}}(\mathcal{C}(\mathcal{A}))$$

where $\mathcal{B}$ is defined in Figure 8 and $\mathcal{C}$ is defined in Figure 9. S_m is the secure distribution of secrets associated with $\textit{KEM}_m$, and S^+ is the derived source of keying material defined in Figure 10.

$\mathcal{B}(\mathcal{A},\ m,\ \hat{P},\ \hat{R},\ \hat{k})$

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$P_m \leftarrow \hat{P}, R_m \leftarrow \hat{R}$
$k_m \leftarrow \hat{k}$
$k' \leftarrow \text{CtKDF}(\text{context}, \text{label}, \ell, \text{psk}, k, P, R)$
return $\mathcal{A}(P, R, k')$

Fig. 8: Constructed adversary against IND-CPA in CtKDF standard model security proof

$\mathcal{C}(\mathcal{A}, (\text{P},\ \text{R}),\ \text{r})$

save(P, R)
return $(f(\text{context}, P, R), \ell)$

$\mathcal{C}(\mathcal{A},\ \text{o})$

load(P, R)
return $\mathcal{A}(P, R, o))$

Fig. 9: Constructed adversary against KDF weak security in CtKDF standard model security proof

Proof. Follows from Theorem 3 and Theorem 4. □

$S^+(S, m)$

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** $\perp$
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** $\perp$
$(\cdot, P_m) \leftarrow\!\!\$\ \mathsf{KGen}_m$
$(\cdot, R_m) \leftarrow\!\!\$\ \mathsf{Enc}_m(P_m)$
$k_m \leftarrow\!\!\$\ S$
return $(\text{psk} \| k, (P, R))$

Fig. 10: Derived keying material source in CtKDF standard model security proof

CtKDF_a($\mathcal{A}$, m)

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$(\cdot, P_m) \leftarrow\!\!\$\ \mathsf{KGen}_m()$
$(\hat{k}, R_m) \leftarrow\!\!\$\ \mathsf{Enc}_m(P_m)$
if $\perp(\hat{k}, R_m)$ **return** 0
$k_m \leftarrow\!\!\$\ \{0,1\}^\ell$
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$k' \leftarrow \text{CtKDF}(\text{context}, \text{label}, \ell, \text{psk}, k, P, R)$
return $\mathcal{A}(P, R, k')$

Fig. 11: Intermediate game in CtKDF standard model security proof

Theorem 3. *For $m \leq n$ and any algorithm $\mathcal{A}$,*

$$\left| \Pr\left[G0^{ind\text{-}cpa}_{KEM}(\mathcal{A}) = 1\right] - \Pr[\,CtKDF\_a(\mathcal{A}, m) = 1]\,\right| \leq \mathsf{Adv}^{ind\text{-}cpa}_{KEM_m, S_m}(\mathcal{B})$$

Proof. After splitting the loop at location m, the IND-CPA assumption for KEM_m can be applied to replace the shared secret with an independent random value. The remaining code in CtKDF_a is the constructed adversary $\mathcal{B}$. □

Theorem 4. *For $m \leq n$ and any algorithm $\mathcal{A}$,*

$$\left| \Pr[\,CtKDF\_a(\mathcal{A}, m) = 1] - \Pr\left[G1^{ind\text{-}cpa}_{KEM}(\mathcal{A}) = 1\right]\right| \leq \mathsf{Adv}^{kdf\text{-}weak}_{KDF, S^+(S_m, m), label}(\mathcal{C}(\mathcal{A}))$$

Proof. CtKDF_a is identical to $G0^{\text{kdf-weak}}_{\text{KDF}, S^+(S_m, m), \text{label}}(\mathcal{C}(\mathcal{A}))$, and the weakly-secure KDF assumption is applied to replace the output of the KDF with an independent random value. After this transformation, the game is equivalent to $G1^{\text{ind-cpa}}_{\text{KEM}}(\mathcal{A})$. □

4.2 CasKDF

CasKDF is proved secure in the random oracle model only, though it may be possible to prove CasKDF secure in the standard model by assuming KDF is

a secure key derivation function similar to what is described in Section 4.1. In the IND-CPA security game for CasKDF, the first $n-1$ values produced by the cascade are given to the adversary as auxiliary information. This arrangement models an expected case in which the earlier values are used as part of a protocol while the cascade is in progress. The proof ensures that the final value produced by the cascade is secret even if the earlier values are used in any way.

Random Oracle Model In the random oracle model, CasKDF is IND-CPA secure as long as at least one KEM is OW-CPA secure. In this setting, the KDF is modeled as a distinct random oracle for each label and length $d+\ell$ as described in Section 2.2. The component KEM schemes may not query the random oracle.

Theorem 5 (CasKDF Security in the Random Oracle Model). *For any $m \leq n$ and any algorithm $\mathcal{A}$ that queries the random oracle at most q times, the advantage of $\mathcal{A}$ against IND-CPA of CasKDF is*

$$\mathsf{Adv}^{ind\text{-}cpa}_{CasKDF,\{0,1\}^\ell}(\mathcal{A}) \leq \mathsf{Adv}^{ow\text{-}cpa}_{KEM_m,c}(\mathcal{B}(\mathcal{A},m)) + 2n^2/2^d + q * n/2^d$$

where c is a function that takes a query and a secret, and returns true iff the whole secret appears in the appropriate location in the query, and $\mathcal{B}$ is defined in Figure 12.

Proof. Follows from Theorem 6 and Theorem 7. □

In Figure 12, $\text{CasKDF}^{\mathcal{O}^\$}_{j\ldots k}$ denotes iterations j though k of the CasKDF combiner that returns the next chain secret and all output values as a list. This combiner accesses a modified random oracle $\mathcal{O}^\$$ that always returns a new random value for each query, and it updates the state of $\mathcal{O}$ so that this value will be returned when $\mathcal{O}$ is queried on the same input.

The second term $(2n^2/2^d)$ in Theorem 5 results from the fact that secret values produced in the cascade may collide with secrets produced by earlier iterations. Such a collision would cause secrets to be reused in the cascade if the corresponding context information is also identical. If the label/context used in each iteration of the cascade is distinct, then this term can be removed from the bound.

$\mathcal{B}(\mathcal{A},\ m,\ \hat{P},\ \hat{R})$

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
$(\cdot, v_{1\ldots(m-1)}) \leftarrow \text{CasKDF}^{\mathcal{O}^\$}_{1\ldots(m-1)}$
  $(\text{context}, \text{label}, \ell, \text{psk}, k, P, R)$
$v_m \leftarrow\!\!\$\ \{0,1\}^\ell, x \leftarrow\!\!\$\ \{0,1\}^d$
$P_m \leftarrow \hat{P}, R_m \leftarrow \hat{R}$
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(\cdot, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
$(\cdot, v_{(m+1)\ldots n}) \leftarrow \text{CasKDF}^{\mathcal{O}^\$}_{(m+1)\ldots n}$
  $(\text{context}, \text{label}, \ell, x, k, P, R)$
$\hat{k} \leftarrow\!\!\$\ \{0,1\}^\ell, \mathcal{A}^{\mathcal{O}}(P, (v_{1\ldots(m-1)}, R), \hat{k})$
return $\text{queries}(\mathcal{A}^{\mathcal{O}})$

Fig. 12: Constructed adversary against OW-CPA in CasKDF random oracle model security proof

CasKDF_0a($\mathcal{A}$, m)

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$(c, v_{1\ldots(m-1)}) \leftarrow \mathrm{CasKDF}^{\mathcal{O}^\$}_{1\ldots m-1}$
  $(\mathrm{context}, \mathrm{label}, \ell, psk, k, P, R)$
$(\cdot, P_m) \leftarrow\!\!\$\ \mathsf{KGen}_m()$
$(k_m, R_m) \leftarrow\!\!\$\ \mathsf{Enc}_m(P_m)$
if $\perp(k_m, R_m)$ **return** 0
$c' \| v_m \leftarrow\!\!\$\ \mathcal{O}^\$$
  $((P_m, R_m, \mathrm{context}_m), (k_m, c))$
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$(\cdot, v_{(m+1)\ldots n}) \leftarrow \mathrm{CasKDF}^{\mathcal{O}^\$}_{m+1\ldots n}$
  $(\mathrm{context}, \mathrm{label}, \ell, c', k, P, R)$
return $\mathcal{A}^{\mathcal{O}}(P, (v_{1\ldots(m-1)}, R), v_n)$

Fig. 13: Intermediate game 0a in CasKDF random oracle model security proof

CasKDF_1a($\mathcal{A}$, m)

for $i = 1 \ldots (m-1)$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$(c, v_{1\ldots(m-1)}) \leftarrow \mathrm{CasKDF}^{\mathcal{O}^\$}_{1\ldots m-1}$
  $(\mathrm{context}, \mathrm{label}, \ell, psk, k, P, R)$
$(\cdot, P_m) \leftarrow\!\!\$\ \mathsf{KGen}_m()$
$(k_m, R_m) \leftarrow\!\!\$\ \mathsf{Enc}_m(P_m)$
if $\perp(k_m, R_m)$ **return** 0
$c' \| v_m \leftarrow\!\!\$\ \mathcal{O}^\$$
  $((P_m, R_m, \mathrm{context}_m), (k_m, c))$
for $i = (m+1) \ldots n$ **do**
  $(\cdot, P_i) \leftarrow\!\!\$\ \mathsf{KGen}_i()$
  $(k_i, R_i) \leftarrow\!\!\$\ \mathsf{Enc}_i(P_i)$
  if $\perp(k_i, R_i)$ **return** 0
$(\cdot, v_{(m+1)\ldots n}) \leftarrow \mathrm{CasKDF}^{\mathcal{O}^\$}_{m+1\ldots n}$
  $(\mathrm{context}, \mathrm{label}, \ell, c', k, P, R)$
$r \leftarrow\!\!\$\ \{0,1\}^\ell$
return $\mathcal{A}^{\mathcal{O}}(P, (v_{1\ldots(m-1)}, R), r)$

Fig. 14: Intermediate game 1a in CasKDF random oracle model security proof

Theorem 6. *For $m \leq n$ and any algorithm $\mathcal{A}$,*

$$\left| \Pr\left[G0_{KEM}^{ind\text{-}cpa}(\mathcal{A}) = 1\right] - \Pr[\mathit{CasKDF\_0a}(\mathcal{A}, m) = 1] \right| \leq n^2/2^d$$
$$\textit{and}$$
$$\left| \Pr\left[G1_{KEM}^{ind\text{-}cpa}(\mathcal{A}) = 1\right] - \Pr[\mathit{CasKDF\_1a}(\mathcal{A}, m) = 1] \right| \leq n^2/2^d$$

Proof. After splitting the loops at location m, the only remaining transformation is the replacement of the random oracle $\mathcal{O}$ with the modified oracle $\mathcal{O}^{\$}$ which returns a new random value for all queries. This modification produces identical results unless the oracle is called twice on the same value during the cascade, which can only happen if there is a collision in the random c values. This collision happens with probability $n^2/2^d$. □

Theorem 7. *For $m \leq n$ and any algorithm $\mathcal{A}$ that queries the random oracle at most q times,*

$$| \Pr[\mathit{CasKDF\_0a}(\mathcal{A}, m) = 1] - \Pr[\mathit{CasKDF\_1a}(\mathcal{A}, m) = 1] | \leq \mathsf{Adv}_{KEM_m,c}^{ow\text{-}cpa}(\mathcal{B}(\mathcal{A}, m)) + q * n/2^d$$

Proof. CasKDF_1a gives the adversary a random value with the same distribution as the value produced by the cascade in CasKDF_0a, except that the value produced in CasKDF_1a is independent of the random oracle. The adversary cannot distinguish these values unless it queries the oracle on the input that was used to produce it in the last iteration of the cascade. The probability of this event is less than the probability that the adversary queries on the input produced in iteration m or in any iteration after m. This is a sum of the probabilities of two events, the first of which is that $\mathcal{B}$ wins the OW-CPA game for KEM_m. In the second event, one of the q adversary queries is identical to one of the cascade queries, and each cascade query includes a uniformly random bit string of length d. The probability of such a collision is at most $q * n/2^d$. □

4.3 Proof Mechanization

The proofs in this paper are mechanized and checked using the Foundational Cryptography Framework[10], a computational cryptography library for the Coq proof assistant. FCF includes a simple probabilistic programming language along with a probability theory and program logic that enables reasoning on programs in this language. The library also includes reusable cryptographic definitions and arguments that were used in these proofs.

The proof is checked by Coq to ensure that it contains no errors. In particular, the mechanized proof rules out some classes of error that have troubled cryptographic proofs in the past:

- All cryptographic assumptions are applied correctly. [7]
- All numeric bounds resulting from the probability of collision are correct. [8]
- All arguments and transformations are valid and are applied correctly. [5]

The mechanized definitions and results are identical to the ones presented in this paper, except that the constructions in the mechanized proof operate on octet strings instead of bit strings. The mechanized proof artifacts are available at `https://github.com/aws-samples/hybrid-kem-fcf`.

5 Interpretation and Caveats

Constructed Adversaries and Complexity Classes The results in Section 4 accept an arbitrary adversary $\mathcal{A}$, and the adversaries that are constructed from $\mathcal{A}$ in the reduction appear in the theorem statements. These constructed adverary procedures define the class of adversary against which security assumptions hold. For example, instead of assuming that a KEM is IND-CPA secure against all probabilistic polynomial time (PPT) adversaries in Theorem 2, we can inspect the constructed adversary $\mathcal{B}$ and see that it is PPT if $\mathcal{A}$ is PPT.

All definitions and reductions in this paper are classical, which implies that the constructions are secure against a computationally-bounded classical adversary. The results also imply security against a quantum adversary that only has classical access to the random oracle.

Salting the Key Derivation Function The constructions in this paper use a fixed label to salt the KDF. In practice, this label must be distinct from any other label used with the KDF on the same secrets. Otherwise, an attacker might be able to leverage this other use of the KDF to obtain knowledge of the secrets.

The two parties may produce a random label through an authenticated exchange that occurs before the exchanges related to the KEM. By doing so, they can ensure (with overwhelming probability) that the label is distinct from other labels used with the KDF. Further, using a random label to salt the KDF allows the provable security of these constructions to benefit from existing results related to salted functions. In particular, the KDF can be assumed to be a generic extractor, which is provably true in the case of HKDF.

Without salt, the standard model CtKDF security result of Theorem 2 relies on the assumption that the KDF is a deterministic extractor for a particular source in which a random bit string is combined with bits that are chosen arbitrarily and independent of the random string. This assumption deserves some scrutiny due to well-known limitations on deterministic extraction. [11]

IND-CCA Security We can transform the IND-CPA security definition (Definition 3) into IND-CCA by giving the adversary access to a decapsulation oracle on the secret key. This definition is useful for demonstrating the security of a KEM when secret keys are reused across multiple sessions, or when an attacker may have some access to the secret key while a session is in progress.

Additional investigation and proof is necessary to determine whether these constructions are IND-CCA secure under the appropriate assumptions on the underlying KEMs. The inclusion of the complete transcript of public information in the KDF context prevents the sort of malleability that leads to obvious

attacks against IND-CCA. In the random oracle model, any allowed query to the decapsulation oracle will result in a random value that is independent of the shared secret. In the standard model, the decapsulation oracle allows the attacker to mount related-key attacks against the KDF, so it is necessary to make a stronger assumption on the KDF to ensure it is secure against such attacks.

Component KEM Random Oracle Access In the security proofs in the random oracle model in Section 4, the component KEMs used in the hybrid constructions are not allowed to query the random oracle. In practice, these KEMs may use the same function that is modeled as a random oracle, and the proof will still apply as long as the KEMs are given distinct clones of the random oracle. For example, each component KEM could use a distinct label with the KDF.

6 Standards

The constructions in this paper are modeled after similar constructions in international standards, and the security of these standards can be derived from the results in Section 4 and additional assumptions.

6.1 ETSI

The CtKDF and CasKDF constructions in Draft ETSI TS 103 744 [1] are specializations of the ones defined in Section 3.

CtKDF The ETSI CtKDF uses HKDF, which is modeled as a random oracle in Theorem 1 and assumed to be a weakly secure KDF for a class of key source in Theorem 2. Both of these theorems require no additional assumptions on the behavior of the formatting function f. For practical purposes, and to support stronger security properties, f should be collision resistant.

CasKDF The KDF in ETSI CasKDF is implemented using HMAC, HKDF, and a formatting function f as shown in Figure 15. In Theorem 5, KDF is modeled as a random oracle on all arguments. The Draft ETSI TS 103 744 CasKDF KDF function is indistinguishable from a random oracle on all arguments if HKDF is modeled as a random oracle and f and HMAC are collision resistant.

Diffie-Hellman Draft ETSI TS 103 744 describes a key exchange abstraction which can be implemented using a KEM or with a construction based on ephemeral Diffie-Hellman. This key exchange abstraction is identical (up to naming) to the KEM definition used in this paper, and the Diffie-Hellman key exchange mapping described in the standard can be viewed as a particular Diffie-Hellman-based KEM. So the results in this paper apply to both KEM and Diffie-Hellman key exchange schemes allowed by the standard.

KDF(secret, label, context, ℓ, psk)

$(s, k) \leftarrow$ secret
$(c, P, R) \leftarrow$ context
return HKDF(HMAC($s, f(k, P, R)$), label, c, ℓ)

Fig. 15: Draft ETSI TS 103 744 CasKDF KDF

6.2 NIST

NIST SP 800-56A Rev. 2[2] allows hybrid shared secrets that are produced by concatenating a "standard" shared secret Z with an arbitrary secret T to produce $Z' = Z \| T$. Note that this standard requires the "standard" secret to be placed first in the concatenation. The rest of the concatenation can contain an arbitrary number of additional secrets. The CtKDF combiner described in this paper complies with this NIST recommendation if all of the following are true:

- The pre-shared key (PSK) is the empty string.
- The first shared secret in the concatenation is produced by an approved key establishment scheme as specified in NIST SP 800-56A and SP 800-56B.
- The KDF is an approved "Two-Step" KDF (such as HKDF) as specified in NIST SP 800-56C.

7 Acknowledgments

We would like to thank Ernie Cohen, Eric Crockett, Scott Fluhrer, Jake Massimo, and Bertram Poettering for conversations that contributed to this work.

References

1. Draft ETSI TS 103 744: Quantum-safe hybrid key exchanges, available to authorized ETSI members during development
2. Sp 800-56C rev. 2: Recommendations for key-derivation methods in key-establishment, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
3. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. Cryptology ePrint Archive, Report 2018/903 (2018), https://eprint.iacr.org/2018/903
4. Coq Development Team: The Coq Reference Manual, https://coq.inria.fr/distrib/current/refman/
5. Gabizon, A.: On the security of the BCTV Pinocchio zk-SNARK variant. Cryptology ePrint Archive, Report 2019/119 (2019), https://eprint.iacr.org/2019/119

6. Giacon, F., Heuer, F., Poettering, B.: KEM combiners. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10769, pp. 190–218. Springer (2018). https://doi.org/10.1007/978-3-319-76578-5_7, https://doi.org/10.1007/978-3-319-76578-5_7
7. Inoue, A., Iwata, T., Minematsu, K., Poettering, B.: Cryptanalysis of OCB2: Attacks on authenticity and confidentiality. Cryptology ePrint Archive, Report 2019/311 (2019), https://eprint.iacr.org/2019/311
8. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. Cryptology ePrint Archive, Report 2012/438 (2012), https://eprint.iacr.org/2012/438
9. Krawczyk, H.: Cryptographic extraction and key derivation: The HKDF scheme. Cryptology ePrint Archive, Report 2010/264 (2010), https://eprint.iacr.org/2010/264
10. Petcher, A., Morrisett, G.: The foundational cryptography framework (2014), http://arxiv.org/abs/1410.3735
11. Santha, M., Vazirani, U.V.: Generating quasi-random sequences from slightly-random sources. In: 25th Annual Symposium onFoundations of Computer Science, 1984. pp. 434–440 (1984)