Security of Hybrid Key Establishment using Concatenation

Adam Petcher and Matthew Campagna

Amazon Web Services

Abstract. In a hybrid key establishment system, multiple independent key establishment schemes are combined in a manner that also combines their security properties. Such constructions can combine systems that are secure in different settings and achieve the combined security of all systems. For example, classical and post-quantum systems can be combined in order to secure communication against current threats as well as future quantum adversaries. This paper describes machine-checked proofs of security for a commonly-used hybrid key establishment system that concatenates the secrets produced by other key establishment systems. Practical interpretation of these results is also provided in order to guide the use of this system in applications and standards.

Keywords: key encapsulation · post-quantum

1 Introduction

Widespread methods of key establishment, such as elliptic curve Diffie-Hellman (ECDH), are insecure against quantum adversaries that may exist in the future. This quantum threat has motivated the development of key establishment systems that are conjectured to be secure against quantum computers [7]. A pressing concern is that encrypted communication could be recorded today, and then decrypted in the future after quantum computers have been developed, and this concern could be addressed by using quantum-secure key establishment today. Ideally, these novel key establishment systems could be deployed along with traditional systems in a way that ensures the same level of security against classical adversaries.

A hybrid key establishment mechanism (KEM)[6] can be used to establish keys that are secure against future quantum adversaries while retaining security against classical adversaries. In a hybrid KEM, multiple KEMs are combined in an attempt to produce a single KEM that has all of the security properties of all component KEMs. For example, a hybrid KEM could combine a post-quantum KEM with a KEM based on ECDH. A 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 paper examines a hybrid KEM combiner that concatenates the secrets provided by the component KEMs, and then gives this concatenated secret to a key derivation function (KDF). The simplicity of this combiner makes it appealing for use as a key establishment system in many applications, and it is used in multiple significant cryptographic standards and draft standards as described in Section 7. This paper contains two proofs that show the KEM constructed using this combiner is secure in different settings. First, the KEM is proved secure according to indistinguishability under chosen plaintext attack (IND-CPA) in the standard model under the assumptions that an underlying KEM is IND-CPA and the KDF is secure according to a typical KDF security definition. Second, the KEM is proved secure according to indistinguishability under chosen ciphertext attack (IND-CCA) when the KDF is modeled as a random oracle and one underlying KEM is assumed to be one way against chosen ciphertext attack (OW-CCA).

The security proofs in this work ensure that the hybrid KEM retains the security of the underlying KEMs, and that the composition is secure if at least one KEM is secure. All theorems include concrete bounds to provide insight into the security of each hybrid KEM when deployed at scale. Many standards only require IND-CPA security, and the fact that this construction is IND-CPA-secure follows directly from the assumptions on the KDF and underlying KEMs. The IND-CCA proof is slightly more complex, and it depends on additional behavior of the combiner aside from the concatenation of secrets. In order to rule out trivial proof errors, the proofs are mechanically checked by the Coq proof assistant[8] using the Foundational Cryptography Framework[16].

2 Notation

In game definitions, $\leftarrow$ indicates assignment of a value to a variable, and the variant $\overset{\$}{\leftarrow}$ indicates an assignment after executing a probabilistic procedure. For stateful procedures, $(\mathsf{state} \leftarrow \cdot)$ and $(\cdot \leftarrow \mathsf{state})$ indicates saving to and loading from state, respectively. This notation also coerces a set into a procedure that samples uniformly from the set. For example, $x \overset{\$}{\leftarrow} \{0,1\}^n$ samples uniformly from the set of bit sequences of length n and stores the result in x. Some definitions use sequences of values, and array notation is used to describe an element of that sequence. For example, if k is a sequence of keys, then $k[i]$ is the key at position i in the sequence. Sequences are also constructed by assigning to each position in the sequence like $k[i] \leftarrow x$ and then using k to refer to the entire sequence. Some algorithms may fail to produce a valid result, and the $\perp$ function is used in definitions to test the results for this failure. The symbol $\perp$ is also used to describe a constant value x for which $\perp(x)$ is true. The symbol $||$ indicates concatenation of bit strings.

3 Security and Correctness Definitions

3.1 Key Encapsulation Mechanisms

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

– KGen is a probabilistic key generation algorithm that produces a pair (sk, pk), where sk is a secret key and pk is a public key.
– Enc is a probabilistic encapsulation algorithm that takes a public key produced by KGen. Enc produces a pair (k, c), where k is a shared secret and c is a ciphertext encapsulating that key.
– Dec is a deterministic decapsulation algorithm that takes a secret key produced by KGen and a ciphertext produced by Enc. Dec produces the shared secret k

A key encapsulation is carried out in practice between two parties, Alice and Bob. Alice runs KGen to produce (sk, pk) and sends pk to Bob. Bob runs $\mathsf{Enc}(pk)$ to produce (k, c) and sends c to Alice. Alice runs $\mathsf{Dec}(sk, c)$ 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 pk and c must be authenticated.

KEM Correctness In order for a KEM to be useful, the secret values returned by Enc and Dec must be the same. For some KEMs, these values may be different, but only with small probability. The correctness error is defined as the probability that these secret values differ.

```
Game $G^{\text{correct}}_{\text{KEM}}$
$(sk, pk) \xleftarrow{\$} \mathsf{KGen}()$
$(k, c) \xleftarrow{\$} \mathsf{Enc}(pk)$
**if** $\perp(k)$ **return** $0$
$k' \leftarrow \mathsf{Dec}(sk, c)$
**return** $k \neq k'$
```

Fig. 1: Correctness game

Definition 1 (KEM Correctness Error). *Let KEM be a key encapsulation mechanism, the correctness error of KEM is*

$$\mathsf{Cor}_{KEM} = \Pr[G^{correct}_{KEM} = 1]$$

Indistinguishability under Chosen Plaintext Attack A KEM has no plaintext, and therefore there is no adversarial choice of plaintext. The word "plaintext" only appears in this definition due to the fact that it is derived from similar definitions used for encryption. A KEM achieves indistinguishability under chosen plaintext attack (IND-CPA) against adversary $\mathcal{A}$ if $\mathcal{A}$ cannot distinguish the secret from a random secret, except with acceptably small probability. $\mathcal{A}$ accepts the secret along with all of the public information produced by the KEM, and it returns a bit that may encode the adversary's decision about whether the secret is real or random. In this and similar definitions, the secret and public information given to $\mathcal{A}$ is called the *challenge*. This definition is parameterized by a probability distribution S that is a source of good random secrets.

```
Game $G0^{\text{ind-cpa}}_{\text{KEM}}(\mathcal{A})$
$(\cdot, pk) \xleftarrow{\$} \mathsf{KGen}()$
$(k, c) \xleftarrow{\$} \mathsf{Enc}(pk)$
if $\bot(k)$ return $0$
return $\mathcal{A}(pk, c, k)$
```

```
Game $G1^{\text{ind-cpa}}_{\text{KEM},\mathsf{S}}(\mathcal{A})$
$(\cdot, pk) \xleftarrow{\$} \mathsf{KGen}()$
$(k, c) \xleftarrow{\$} \mathsf{Enc}(pk), k' \xleftarrow{\$} \mathsf{S}()$
if $\bot(k)$ return $0$
return $\mathcal{A}(pk, c, k')$
```

Fig. 2: IND-CPA games

Definition 2 (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*

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

Indistinguishability under Chosen Ciphertext Attack Indistinguishability under chosen ciphertext attack (IND-CCA) is identical to IND-CPA except that the adversary is also given access to a Chosen Ciphertext Attack (CCA) oracle that invokes Dec on any input ciphertexts except for the one associated with the challenge.

```
Game $G0^{\text{ind-cca}}_{\text{KEM}}(\mathcal{A})$
$(sk, pk) \xleftarrow{\$} \mathsf{KGen}()$
$(k, c) \xleftarrow{\$} \mathsf{Enc}(pk)$
if $\bot(k)$ return $0$
return $\mathcal{A}^{\mathcal{D}_{sk,c}}(pk, c, k)$
```

```
Game $G1^{\text{ind-cca}}_{\text{KEM},\mathsf{S}}(\mathcal{A})$
$(sk, pk) \xleftarrow{\$} \mathsf{KGen}()$
$(k, c) \xleftarrow{\$} \mathsf{Enc}(pk), k' \xleftarrow{\$} \mathsf{S}()$
if $\bot(k)$ return $0$
return $\mathcal{A}^{\mathcal{D}_{sk,c}}(pk, c, k')$
```

Fig. 3: IND-CCA games

```
Oracle $\mathcal{D}_{\tilde{sk},\tilde{c}}(c)$

if $c = \tilde{c}$ return $\perp$
return $\mathsf{Dec}(\tilde{sk}, c)$
```

Fig. 4: CCA oracle

```
Game $G^{\text{ow-cca}}_{\text{KEM}}(\mathcal{A})$

$(sk, pk) \xleftarrow{\$} \mathsf{KGen}()$
$(k, c) \xleftarrow{\$} \mathsf{Enc}(pk)$
if $\perp(k)$ return $0$
$s \xleftarrow{\$} \mathcal{A}^{\mathcal{D}_{sk,c}}(pk, c)$
if $k \in s$
  return $1$
else
  return $0$
```

Fig. 5: OW-CCA game

Definition 3 (IND-CCA Advantage). *Let KEM be a key encapsulation mechanism and $\mathcal{A}$ be an algorithm, then the advantage of $\mathcal{A}$ against IND-CCA of KEM w.r.t. source S is*

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

One Way under Chosen Ciphertext Attack A KEM is one way under chosen ciphertext attack (OW-CCA) 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 $\mathcal{A}$ also gets access to the CCA oracle. The adversary produces a set of values and, they win if the shared secret is in the set.

Definition 4 (OW-CCA Advantage). *Let KEM be a key encapsulation mechanism and $\mathcal{A}$ be an algorithm, then the advantage of $\mathcal{A}$ against OW-CCA of KEM is*

$$\mathsf{Adv}^{\textit{ow-cca}}_{\textit{KEM}}(\mathcal{A}) = \Pr[G^{\textit{ow-cca}}_{\textit{KEM}}(\mathcal{A}) = 1]$$

3.2 Key Derivation Functions

A key derivation function (KDF) is a function on four arguments (s, r, c, ℓ), where s is the input key material, r is salt, c is arbitrary information (a.k.a. “info”) associated with the output key material, and ℓ is the desired output key material length.

Secure Key Derivation Function To prove IND-CPA security of the hybrid KEM, we can use a weaker form of the secure key derivation function definition from [13], in which the adversary is not given access to a KDF oracle. In this definition, S is a source of input keying material that also produces auxiliary public information, and Salt is a source of extractor salt. The definition is also modified to allow the source S to fail to produce a value.

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

```
Game $G1^{\text{kdf-weak}}_{\text{KDF,Salt},\mathsf{S}}(\mathcal{A})$
$(s, a) \xleftarrow{\$} \mathsf{S}()$
if $\bot(s)$ return 0
$r \xleftarrow{\$} \text{Salt}()$
$(c, \ell) \xleftarrow{\$} \mathcal{A}(a, r)$
$o \xleftarrow{\$} \{0, 1\}^{\ell}$
return $\mathcal{A}'(o)$
```

Fig. 6: KDF weak security games

Definition 5 (Weakly Secure KDF Advantage). *Let KDF be a function, S be a source of input key material, Salt be a source of salt, and $\mathcal{A}$ and $\mathcal{A}'$ be a pair of procedures comprising an adversary, then the advantage of this adversary against weak security of KDF when extracting from S using Salt is*

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

Key Derivation Functions as Random Oracles In one 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.

4 Hybrid KEMs and the Concatenation KDF Combiner

A hybrid key encapsulation mechanism is composed of key encapsulation 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 construction uses a combiner $\mathcal{C}$ that takes the information produced from each KEM along with some context v and label l. The combiner produces a shared secret of the desired length ℓ. If any routine from any KEM fails to produce a value, then the hybrid KEM fails. This definition of a hybrid KEM is only used to illustrate how a hybrid KEM is organized in practice, and proofs use a more general definition that allows the adversary to have full control over some of the KEMs.

```
$\mathsf{KGen}()$
for $i = 1 \ldots n$ do
    $(sk[i], pk[i]) \xleftarrow{\$} \mathsf{KGen}_i()$
return $(sk, pk)$
```

Fig. 7: Notional Hybrid KGen

```
$\mathsf{Enc}_{v,l,\ell}(pk)$
for $i = 1 \ldots n$ do
    $(k[i], c[i]) \xleftarrow{\$} \mathsf{Enc}_i(pk[i])$
    if $\perp(k[i])$ return $\perp$
return $(\mathcal{C}(v, l, \ell, k, pk, c), (pk, c))$
```

Fig. 8: Notional Hybrid Enc

```
$\mathsf{Dec}_{v,l,\ell}(sk, (pk, c))$
for $i = 1 \ldots n$ do
    $k[i] \leftarrow \mathsf{Dec}_i(pk[i])$
    if $\perp(k[i])$ return $\perp$
return $\mathcal{C}(v, l, \ell, k, pk, c)$
```

Fig. 9: Notional Hybrid Dec

```
$\mathrm{CtKDF}_{\mathrm{f}}(v, l, \ell, k, pk, c)$
$\mathrm{secret} \leftarrow k[1] \| k[2] \| \ldots \| k[n]$
$v' \leftarrow \mathrm{f}(v, pk, c)$
return $\mathrm{KDF}(\mathrm{secret}, l, v', \ell)$
```

Fig. 10: CtKDF combiner

The concatenation KDF combiner (CtKDF) produces an intermediate secret by concatenating the shared secrets produced by the KEMs. This intermediate secret is given to the KDF along with an info value that is produced by applying the function f to all of the public information produced by the KEMs. The function f is used to format the information so that it can be used as a KDF info value, and the function will have different required properties in different proofs. The output of the KDF is the shared secret produced by the hybrid KEM.

5 Security Proofs

The hybrid KEM using the CtKDF combiner is proved IND-CPA-secure in the standard model and IND-CCA-secure in the random oracle model (ROM). These proofs use the more general hybrid KEM definition shown in Figures 11, 13, and 12. In this definition, there is one KEM at position s in the sequence of KEMs that is "strong" in the sense that we only make security/correctness assumptions about this KEM. This KEM is composed of (KGen_s, Enc_s, Dec_s), and it executes first. Then the adversary procedure $\hat{\mathcal{A}}$ gets the public outputs from this KEM and produces all of the values from the other KEMs. $\hat{\mathcal{A}}$ also chooses the context value that is given to the combiner, and $\hat{\mathcal{A}}$ is allowed to communicate freely with $\mathcal{A}$. The total number of KEMs, the position of the values from the strong KEM in the sequences/concatenation, and the length of the secret produced by each KEM are arbitrary but must be chosen before the game.

The hybrid KEM ciphertext includes the public key and ciphertext of the strong KEM, as well as all secrets, public keys, ciphertext, and context information chosen by $\hat{\mathcal{A}}$. In the return statement of Figure 13, pk and c are the combined sequences containing the strong values as well as values supplied by $\hat{\mathcal{A}}$. The hybrid ciphertext does not include the strong secret, so only the value $\hat{k}$ is added to the ciphertext.

In the hybrid Dec construction in Figure 12, only the strong decapsulation procedure Dec_s is used. The adversary chooses the other shared secret values by including them in the hybrid ciphertext supplied to this procedure. This organization is used to model a scenario in which the adversary has enough control over the other Dec procedures that it can freely choose the ciphertext values and the resulting shared secret values.

```
$\mathsf{KGen}()$

$(sk_s, pk_s) \xleftarrow{\$} \mathsf{KGen}_s()$
return $(sk_s, pk_s)$
```

Fig. 11: Hybrid KGen

```
$\mathsf{Dec}_{v,l,\ell}(sk, (pk, c, k, v))$

$k[s] \leftarrow \mathsf{Dec}_s(sk, c[s])$
if $\perp(k)$ return $\perp$
return $\mathcal{C}(v, l, \ell, k, pk, c)$
```

Fig. 12: Hybrid Dec

```
$\mathsf{Enc}_{v,l,\ell}(pk_s)$

$(k_s, c_s) \xleftarrow{\$} \mathsf{Enc}_s(pk_s)$
$(\hat{k}, \hat{pk}, \hat{c}, v) \xleftarrow{\$} \hat{\mathcal{A}}(pk_s, c_s)$
for $i = 1 \dots n$ do
  if $i = s$
    $(k[i], pk[i], c[i]) \leftarrow (k_s, pk_s, c_s)$
  else
    $(k[i], pk[i], c[i]) \leftarrow (\hat{k}[i], \hat{pk}[i], \hat{c}[i])$
if $\perp(k)$ return $\perp$
return $(\mathcal{C}(v, l, \ell, k, pk, c), (pk, c, \hat{k}, v))$
```

Fig. 13: Hybrid Enc

5.1 CtKDF is IND-CCA in the Random Oracle Model

In the random oracle model, CtKDF is IND-CCA secure as long as at least one KEM is correct and OW-CCA secure. In this setting, the KDF is modeled as a random oracle $\mathcal{O}$ as described in Section 3.2. The secure KEM may not access the random oracle, but both adversary procedures $\hat{\mathcal{A}}$ and $\mathcal{A}$ may access the random oracle. This proof relies on the fact that the function f is an injection, so the combiner includes the ciphertext value in the info value provided to the KDF. If this value was not provided to the KDF, then the adversary could get past the equality check in the CCA oracle by choosing a different ciphertext value, but a set of secret values that match the ones it chose at the beginning of the game.

Theorem 1 (CtKDF IND-CCA Security in the Random Oracle Model). *For any injective function f and any adversary comprising procedures $\hat{\mathcal{A}}$ and $\mathcal{A}$, the advantage of this adversary against IND-CCA of CtKDF is*

$$\mathsf{Adv}^{\textit{ind-cca}}_{\textit{CtKDF}_f,\{0,1\}^\ell}(\hat{\mathcal{A}}^{\mathcal{O}}, \mathcal{A}^{\mathcal{O},\mathcal{D}}) \leq \mathsf{Adv}^{\textit{ow-cca}}_{\textit{KEM}_s}(\mathcal{B}^{\mathcal{O},\mathcal{D},\hat{\mathcal{A}},\mathcal{A}}) + \mathsf{Cor}_{\textit{KEM}_s}$$

where $\mathcal{B}$ is defined in Figure 14.

```
$\mathcal{B}^{\mathcal{O},\mathcal{D},\hat{\mathcal{A}},\mathcal{A}}(pk_s, c_s)$

$(\hat{k}, \hat{pk}, \hat{c}, v) \xleftarrow{\$} \hat{\mathcal{A}}^{\mathcal{O}}(pk_s, c_s)$
**for** $i = 1 \ldots n$ **do**
    **if** $i = s$
        $(pk[i], c[i]) \leftarrow (pk_s, c_s)$
    **else**
        $(pk[i], c[i]) \leftarrow (\hat{pk}[i], \hat{c}[i])$
**let** $\mathcal{D}' = \hat{\mathcal{D}}^{\mathcal{D}}_{pk,c,\hat{k},v}$ **in**
$okm \xleftarrow{\$} \{0,1\}^\ell$
$\mathcal{A}^{\mathcal{O},\mathcal{D}'}(pk, (pk, c, \hat{k}, v), okm)$
**return** $\mathcal{S}(\mathcal{Q}(\mathcal{O}))$
```

Fig. 14: Constructed adversary against OW-CCA in CtKDF ROM security proof

```
Oracle $\hat{\mathcal{D}}^{\mathcal{D}}_{pk',c',k',v'}(pk, c, k, v)$

**if** $((pk, c, k, v) = (pk', c', k', v'))$ **return** $\perp$
$k[s] \leftarrow$ **if** $c[s] = c'[s]$ **then** $\omega$ **else** $\mathcal{D}(c[s])$
**if** $\perp(k)$ **return** $\perp$
**return** $\mathcal{O}(k[1] \| k[2] \| \ldots \| k[n], l, \mathrm{f}(v, pk, c))$
```

Fig. 15: Constructed CCA Oracle in CtKDF ROM security proof

In Figure 14, $\mathcal{Q}(\mathcal{O})$ returns the sequence of queries that were made to the random oracle over the course of the game. The $\mathcal{S}$ function takes this sequence and removes all information except for the strong secret values. Also in this figure, $\hat{\mathcal{D}}$ is the CCA oracle constructed using $\mathcal{D}$, and $\mathcal{D}'$ is an instantiation of this oracle on specific parameters produced by this game.

Figure 15 shows the oracle that is constructed from the CCA oracle and given to this adversary. In this figure, ω is a special value that is used in place of the secret produced by the strong KEM. The domain of the random oracle is modified in the proof to allow the secret in position s to be either a bit string or this special value ω. Also in this figure, l is a constant value defined by the construction that is typically used as a label. The construction also defines an output length value ℓ that does not appear in these figures due to the way the random oracle is modeled.

The proof steps are summarized here, and the complete mechanized proof [15] provides additional details.

1. The two IND-CCA games produce identical probability distributions unless one of the adversary procedures queries the random oracle on the KDF input tuple that is used to produce the strong secret output. The IND-CCA advantage of the adversary against CtKDF is at most the probability of this event b. Due to the equality test in the CCA oracle, this event can only occur due to direct queries to the random oracle by the adversary. The remaining proof steps are used to produce a bound on the probability of this event.
2. In the game that defines the probability of event b, the random oracle is queried on a tuple associated with the challenge. This query can be removed so that this tuple does not appear in the history of queries to the random oracle. This game transformation does not change the probability distribution.
3. Continue transforming the game associated with event b and remove all calls to Dec_s on the ciphertext value associated with the challenge c_s. Instead of calling Dec_s to produce the required value, use the secret value k_s produced by Enc_s when the challenge values were created. The distance between this transformed game and the previous game is the correctness error of KEM_s.
4. Transform the game so that it defines a new event b', in which the random oracle is queried on any value containing secret produced by Enc_s. This event can occur when the adversary queries the random oracle directly, or when it is produced by a call to Dec_s on some value other than the challenge ciphertext c_s. The probability of event b' is greater than or equal to the probability of event b in the previous game.
5. In the CCA oracle, use the value ω in place of k_s when queried on the challenge ciphertext c_s. The probability of event b' in this game is equivalent to the probability of event b' in the previous game.
6. After the last transformation, the game does not call Dec_s on the challenge ciphertext c_s, and it only uses the challenge secret k_s to check its presence in the history of random oracle queries. This game is equivalent to game defining OW-CCA for KEM_s against the constructed adversary described previously.

The game transformations in steps 3 and 5 above are the only ones that do not produce identical probability distributions, so the terms associated with the correctness and OW-CCA security of KEM$_s$ are the only terms in the bound.

5.2 CtKDF is IND-CPA in the 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. This proof does not require the ciphertext to be given to the KDF as info, so f can be any function.

Theorem 2 (CtKDF Security in the Standard Model). *For any function f and any adversary comprising procedures $\hat{\mathcal{A}}$ and $\mathcal{A}$, the advantage of $\mathcal{A}$ against IND-CPA of CtKDF is*

$$\mathsf{Adv}^{\textit{ind-cpa}}_{CtKDF_f,\{0,1\}^\ell}(\hat{\mathcal{A}},\mathcal{A}) \le \mathsf{Adv}^{\textit{ind-cpa}}_{KEM_s,K}(\mathcal{B}^{\hat{\mathcal{A}},\mathcal{A}}) + \mathsf{Adv}^{\textit{kdf-weak}}_{KDF,K^+(K)}(\mathcal{C}^{\mathcal{A}},\mathcal{C}'^{\mathcal{A}})$$

where $\mathcal{B}$ is defined in Figure 16, and $\mathcal{C}$ and $\mathcal{C}'$ are defined in Figure 17. K is the secure distribution of secrets associated with KEM$_s$, and K^+ is the derived source of input keying material defined in Figure 18.

```
$\mathcal{B}^{\hat{\mathcal{A}},\mathcal{A}}(pk_s, c_s, k_s)$

$(\hat{k}, \hat{pk}, \hat{c}, v) \xleftarrow{\$} \hat{\mathcal{A}}(pk_s, c_s)$
for $i = 1 \ldots n$ do
    if $i = s$
        $(k[i], pk[i], c[i]) \leftarrow (k_s, pk_s, c_s)$
    else
        $(k[i], pk[i], c[i]) \leftarrow (\hat{k}[i], \hat{pk}[i], \hat{c}[i])$
$okm \leftarrow \mathrm{CtKDF}_{\mathrm{f}}(v, l, \ell, k, pk, c)$
return $\mathcal{A}(pk, c, okm)$
```

Fig. 16: Constructed adversary against IND-CPA in CtKDF standard model security proof

```
$\mathcal{C}^{\mathcal{A}}((pk, c, v), r)$

$\mathsf{state} \leftarrow (pk, c)$
return $(\mathrm{f}(v, pk, c), \ell)$
```

```
$\mathcal{C}'^{\mathcal{A}}(okm)$

$(pk, c) \leftarrow \mathsf{state}$
return $\mathcal{A}(pk, c, okm)$
```

Fig. 17: Constructed adversary against KDF weak security in CtKDF standard model security proof

The constructed adversary accepts the salt value r from the KDF security definition, but it does not examine it. In all definitions used in this proof, the value ℓ is a global constant that determines the output key length.

The simple proof is summarized below.

```
$K^+(K)$

$(\cdot, pk_s) \xleftarrow{\$} \mathsf{KGen}_s()$
$(\cdot, c_s) \xleftarrow{\$} \mathsf{Enc}_s(pk_s)$
$k \xleftarrow{\$} K$
$(\hat{k}, \hat{pk}, \hat{c}, v) \xleftarrow{\$} \hat{\mathcal{A}}(pk_s, c_s)$
for $i = 1 \ldots n$ do
  if $i = s$
    $(k[i], pk[i], c[i]) \leftarrow (k, pk_s, c_s)$
  else
    $(k[i], pk[i], c[i]) \leftarrow (\hat{k}[i], \hat{pk}[i], \hat{c}[i])$
return $(k[1] \| k[2] \| \ldots \| k[n], (pk, c, v))$
```

Fig. 18: Derived keying material source in CtKDF standard model security proof

1. Beginning with $G0^{\text{ind-cpa}}$, replace the secret produced by Enc_s with a random value sampled from K. The distance between the transformed game and the previous game is the IND-CPA advantage of constructed adversary $\mathcal{B}$ against IND-CPA of KEM$_s$.
2. Replace the output secret with a random value sampled from S. The distance between this game and the previous game is the weakly-secure KDF advantage of constructed adversary $\mathcal{C}$ against KDF when extracting from source $K^+(K)$.

Proof Mechanization The proofs in this paper are mechanized and checked using the Foundational Cryptography Framework(FCF) [16], 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. [12]
- All arguments and transformations are valid and are applied correctly. [10]

6 Interpretation and Caveats

Inclusion of the Ciphertext in the KDF Info The IND-CCA proof relies on the fact that the ciphertext is given to the KDF in the info parameter. One could imagine a simpler construction in which only the concatenated secrets are given to the KDF. This simpler construction would not be IND-CCA, because the adversary could use its control over the other KEMs to produce a different ciphertext that results in the same concatenated secrets. This ciphertext would bypass the equality check in the CCA oracle, and then the adversary would learn the output secret. This observation was also made for similar constructions in [11] and [6].

Constructed Adversaries and Complexity Classes The results in Section 5 accept an arbitrary adversary comprising procedures $\mathcal{A}$ and $\hat{\mathcal{A}}$, and the adversaries that are constructed from these procedures 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.

Salting the Key Derivation Function The construction in this paper uses 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 can 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 would allow the provable security of this construction to benefit from existing results related to salted functions. In particular, the KDF could be assumed to be a generic extractor, which is provably true in the case of HKDF [13].

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 limitations [17] on deterministic extraction.

Strong KEM Random Oracle Access In the security proofs in the random oracle model in Section 5, the strong KEM used in the hybrid constructions is not allowed to query the random oracle. In practice, this KEM may use the same function that is modeled as a random oracle, and the proof will still apply as long as the KEM is given a distinct clone of the random oracle. For example, the strong KEM could invoke the KDF using a distinct label.

Classical Oracle Access In the IND-CCA proof, the reduction and all definitions are classical. This result implies security against a classical adversary as well as security against a quantum adversary that only has classical access to the CCA oracle and the random oracle. In practice, when this hybrid KEM is implemented on a classical computer, it is reasonable to assume that a quantum adversary only has classical access to the CCA oracle. A quantum adversary is expected to have quantum access to the hash function, so we can only use this result to conclude security against a quantum adversary if we assume that the (classical) ROM is a reasonable model for a hash function implemented on a quantum computer. The general lifting theorem of [18] does not apply because the adversary can force the game to make a non-constant number of queries to the random oracle.

Key Reuse IND-CPA security does 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. The IND-CCA security proof implies that private keys produced by KGen can be saved and reused across multiple sessions. Though there are two important caveats associated with this implication:

- This proof requires the KDF to be modeled as a random oracle. In contrast, the IND-CPA proof uses a relatively weak assumption about the KDF.
- Issues related to forward secrecy and compromise of the reused key are outside the scope of this paper. In practice, key reuse should be avoided in order to provide forward secrecy.

7 Standards

The construction in this paper is modeled after similar constructions in international standards, and the security of these standards can be derived from the results in Section 5 and additional assumptions.

7.1 ETSI

The CtKDF construction in ETSI TS 103 744 [1] is a specialization of the one defined in Section 4. 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 particular source of key material in Theorem 2.

The IND-CPA theorem requires no additional assumptions on the behavior of the formatting function f. ETSI TS 103 744 describes a formatting function that concatenates the values including their lengths, and then hashes the result to produce the info value for the KDF. The part leading up to (but not including) the hash function invocation is injective and can be viewed as the function f in the IND-CCA proof. We can view the hash function invocation as part of the random oracle.

The hybrid KEM in this standard also accepts a pre-shared key which is included in the concatenation along with the other secrets. The definition of the hybrid Enc procedure in Figure 13 allows the adversary procedure $\hat{\mathcal{A}}$ to produce values that will be used as these pre-shared keys. So the proofs in this paper also apply to this standardized hybrid KEM when arbitrarily bad pre-shared keys are used along with at least one good KEM. The proofs in Section 5 do not imply that the hybrid KEM is secure when only the pre-shared key is strong, but facts of this sort could be proved using arguments that are similar to the ones in this section.

7.2 NIST

NIST SP 800-56C Rev. 2 [4] allows hybrid shared secrets that are produced by contatenating a "standard" shared secret Z with an arbitrary secret T to produce $Z' = Z\|T$. The"standard" secret is placed first in the concatentation, and the rest of the concatenation can contain an arbitrary number of additional secrets. The CtKDF combiner described in this paper complies with this NIST standard if all of the following are true:

– 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 as specified in NIST SP 800-56C.

NIST SP 800-56C Rev. 2 does not place strict requirements on the information that is provided as info to the KDF. This information may be produced using an injective function f that binds the resulting secret to the public keys and ciphertexts, in which case the IND-CCA proof applies. The variant of CtKDF used in this standard is IND-CPA regardless of the properties of the formatting function f.

7.3 IETF

Internet draft draft-stebila-tls-hybrid-design [2] describes how to incorporate post-quantum KEMs in TLS 1.3 via a hybrid construction that uses concatenation. This construction is used to produce the TLS handshake secret. It is a special case of CtKDF, so it is IND-CPA according to the proof in Section 5. TLS 1.3 invokes HKDF multiple times using different info values to produce different secrets. The info provided to HKDF includes a hash of all of the handshake messages that were exchanged, and these handshake messages include all of the KEM public keys and ciphertexts. If we view this hash function invocation as part of the random oracle, this construction is also IND-CCA according to the proof in Section 5. In order to apply this result to TLS 1.3, choose one of the secrets (e.g. the traffic secret) as the challenge value that is produced by the game, and the CCA oracle can be used to produce the other secrets by using different info values. The adversary cannot distinguish the chosen secret from a random value, even with knowledge of the other secrets and anything else that can be learned from the CCA oracle. Since the choice of secret is arbitrary, this security result applies to all secrets produced using HKDF in TLS 1.3.

Internet draft draft-kampanakis-curdle-ssh-pq-ke [3] uses concatenation of secrets to incorporate a post-quantum KEM into SSH. SSH produces keys and IVs by hashing a concatenation of values: `HASH(K || H || X || session_id)`, where `X` varies according to the purpose of the derived material. When a hyrbid KEM is desired, `K` is the concatenation of two secrets produced by two KEMs, and the public keys and ciphertexts are incorporated into the hash `H`. Applying the proofs in this paper to this draft standard is less direct due to the use of the bare hash function, and due to the lack of intermediate steps in the construction that produce secrets. In order to apply the IND-CPA result, we can view the entire set of calls to the hash function as a single invocation of a secure KDF that produces multiple secrets. To apply the IND-CCA result, we can view each each secret independently, and allow other secrets for the same sessions to be produced using the CCA oracle. The hash function is modeled as a random oracle, and the construction is IND-CCA secure as long as the Dec oracle only accepts values of `H`, `X`, and `session_id` of a fixed length that is chosen at the beginning of the game. In practice, the lengths of these values must be fixed for the lifetime of the secrets, and the implementation must reject any value of the wrong length.

8 Related Work

Hybrid KEM combiners have been studied previously. The standards described in this paper could also be modeled using the dualPRF combiner of [6] or the split-key PRF combiner of [11]. Compared to those models, the model used in this paper is slightly less abstract, as it explicitly describes the concatenation operation and the related requirements on the concatenated values. This paper also contributes machine-checked proofs and an explanation of how these proofs apply to various (draft) standards.

Machine-checked proofs of security have been previously developed in order to ensure proofs are valid and to increase trust in the security of cryptographic constructions. For example, SHA-3 was proved to be indifferentiable from a random oracle using EasyCrypt [5], the TLS record layer was proved correct using F* [9], and the WireGuard protocol was proved secure using CryptoVerif [14].

9 Acknowledgments

We would like to thank Ernie Cohen, Eric Crockett, Scott Fluhrer, Torben Hansen, Panos Kampanakis, Yan Peng, and Bertram Poettering for their contributions to this paper.

References

1. ETSI TS 103 744: Quantum-safe hybrid key exchanges, https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf
2. Hybrid key exchange in tls 1.3, https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-06.html
3. Post-quantum hybrid key exchange in ssh, https://www.ietf.org/archive/id/draft-kampanakis-curdle-ssh-pq-ke-00.html
4. SP 800-56C rev. 2: Recommendations for key-derivation methods in key-establishment, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
5. Almeida, J.B., Baritel-Ruet, C., Barbosa, M., Barthe, G., Dupressoir, F., Grégoire, B., Laporte, V., Oliveira, T., Stoughton, A., Strub, P.Y.: Machine-checked proofs for cryptographic standards: Indifferentiability of sponge and secure high-assurance implementations of sha-3. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 1607–1622. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363211, https://doi.org/10.1145/3319535.3363211
6. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 206–226. Springer International Publishing, Cham (2019)
7. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: Crystals - kyber: A cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroSP). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
8. Coq Development Team: The Coq Reference Manual, https://coq.inria.fr/distrib/current/refman/
9. Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Protzenko, J., Rastogi, A., Swamy, N., Zanella-Béguelin, S., , K.B., , J.P., , J.K.Z.: Implementing and proving the tls 1.3 record layer. In: SP '17 38th IEEE Symposium on Security and Privacy (May 2017), https://www.microsoft.com/en-us/research/publication/implementing-proving-tls-1-3-record-layer/
10. 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
11. Giacon, F., Heuer, F., Poettering, B.: Kem combiners. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography – PKC 2018. pp. 190–218. Springer International Publishing, Cham (2018)
12. 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
13. Krawczyk, H.: Cryptographic extraction and key derivation: The HKDF scheme. Cryptology ePrint Archive, Report 2010/264 (2010), https://eprint.iacr.org/2010/264
14. Lipp, B., Blanchet, B., Bhargavan, K.: A mechanised cryptographic proof of the WireGuard virtual private network protocol. In: IEEE European Symposium on Security and Privacy (EuroS&P'19). pp. 231–246. IEEE Computer Society, Stockholm, Sweden (Jun 2019)

15. Petcher, A.: Hybrid KEM Proofs in FCF (Apr 2023), `https://github.com/aws-samples/hybrid-kem-fcf`
16. Petcher, A., Morrisett, G.: The foundational cryptography framework. CoRR **abs/1410.3735** (2014), `http://arxiv.org/abs/1410.3735`
17. Santha, M., Vazirani, U.V.: Generating quasi-random sequences from slightly-random sources. In: 25th Annual Symposium on Foundations of Computer Science, 1984. pp. 434–440 (1984)
18. Yamakawa, T., Zhandry, M.: Classical vs quantum random oracles. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 568–597. Springer International Publishing, Cham (2021)