# HMAC Based Key Combiners for Multiple Keys

CFRG, IETF 124 @ Montreal

**Guilin Wang**, and Haoyang Wang

# Key Combiners for Multiple Keys

- **Information about this new draft**
  - **Title**: **HMAC Based Hybrid Key Combiners for Multiple Keys**
  - draft-wang-cfrg-key-combiners-00
  - **Date submitted**: 2025-10-20
  - https://datatracker.ietf.org/doc/draft-wang-cfrg-key-combiners/
- **What is the draft about?**
  - Key combiners are building blocks for security protocols (TLS 1.3, IKEv2 …), in particular for PQ migration.
  - This draft specifies two provably secure key combiners for multiple keys, both based on HMAC.
  - HKCv1: designed for simultaneously available input keys using concatenation
  - HKCv2: tailored for scenarios where input keys arrive incrementally
  - Decoupling feature: the multiple input keys can be produced by different algorithms or even different approaches, and without transcripts involved in deriving the pseudorandom master key.

# Key Combiners for Multiple Keys: Constructions

## ❑ HKCv1(SLAT, K1||K2 ||…||Kn, CTX, L)

PRK=HMAC(SALT, K1||K2 ||…||Kn)
K'=HMAC(PRK, CTX) (7)
K=Truncate(K', L)

- **L≤k≤ min{|K1|,⋯ ,|Kn|},** where L is the final output length of HKCv1, and k is the output length of HMAC.
- **SALT** is a salt value, which does not need to be secret, but it should ideally be unpredictable or at least unique for different contexts.
- **CTX**, denoting the context info, includes information like protocol identifiers, algorithm IDs, user IDs, nonces, etc.
- HCKv1 is useful in scenarios like TLS 1.3

## ❑ HKCv2(SLAT, K1||K2 ||…||Kn, CTX, L)

S1=HMAC(SALT,K1)
Si=HMAC(S(i−1), Ki), 1<i≤n (8)
K'=HMAC(Sn, CTX)
K=Truncate(K', L)

- HCKv2 is useful in scenarios like IKEv2

## ❑ CatKDF from ETSI TS 103 744 (2025.3)

```
K=KDF(secret, label, context, length)
 =HMAC(Secret, Data)                                                   (3)
 =HMAC(label, counter||(psk||k1||k2)||f(info, MA, MB))   (repeatable)
```

- secret= psk||k1||k2, k1 and k2 are obtained by running traditional and PQ KEMs.
- Context=f(info, MA, MB), where f is a context formatting function, and MA, MB are the transcripts output by parties A and B, respectively.

**Also CasKDF from ETSI TS 103 744 (2025.3)**

## ❑ Key Combiner from NIST SP800-133 (2020.6)

```
K=T(HMAC-hash(salt, K1||K2||...||Kn||D1|D2|| ...||Dn), kLen)        (5)
```

- data (D1, …, Dm) can also be combined with the Ki to form K that are independence from K1, …, Kn.
- Each Ki shall be generated and/or established independently using algorithm approved by NIST
- Each Ki shall be kept secret and shall not be used for any purpose other than the computation of a specific symmetric key K.

## ❑ draft-ounsworth-cfrg-kem-combiners (2024.8)

```
ss = KDF(counter || k_1 || ... || k_n || fixedInfo, outputBits)      (6)
```

- k_i=ct_i||ss_i or Ki=ct_i||rlen(ct_i)||ss_i||rlen(ss_i)

# Key Combiners for Multiple Keys: Comparison

### ❑ HKCv1(SLAT, K1||K2 ||…||Kn, CTX, L)

PRK=HMAC(SALT, K1||K2 ||…||Kn)
K'=HMAC(PRK, CTX) (7)
K=Truncate(K', L)

- **Efficiency:** decoupled from transcripts of key generation.
- **Security:** HCKv1 is a provably secure randomness extractor, if the underlying hash function of HMAC is δ-AU (δ-Almost Universal) [WWW25], weaker than collision-resistant.

### ❑ HKCv2(SLAT, K1||K2 ||…||Kn, CTX, L)

S1=HMAC(SALT,K1)
Si=HMAC(S(i−1), Ki), 1<i≤n (8)
K'=HMAC(Sn, CTX)
K=Truncate(K', L)

- **Efficiency:** decoupled from transcripts of key generation.
- **Security:** HKCv2 is an IND-CCA secure computational extractor as long as one of the input keys has enough min-entropy, under random oracle model [WWW25].

### ❑ CatKDF from ETSI TS 103 744 (2025.5)

```
K=KDF(secret, label, context, length)
 =HMAC(Secret, Data)                                              (3)
 =HMAC(label, counter||(psk||k1||k2)||f(info, MA, MB))   (repeatable)
```

**Also CasKDF from ETSI TS 103 744 (2025.5)**

- CatKDF and CasKDF are just specified for three input keys.
- The transcripts are involved in context formatting function
- CatKDF is IND-CPA secure in the standard model and IND-CCA secure in the Random Oracle model [PC23].
- CatKDF is IND-CPA secure in the Random Oracle model [CP21].

### ❑ Key Combiner from NIST SP800-133

```
K=T(HMAC-hash(salt, K1||K2||...||Kn||D1|D2|| ...||Dn), kLen)      (5)
```

- One HMAC involved, Di optional, but strong requirements for ki
- To our best knowledge, no formal security proofs for this scheme

### ❑ draft-ounsworth-cfrg-kem-combiners

```
ss = KDF(counter || k_1 || ... || k_n || fixedInfo, outputBits)    (6)
```

- For keys via running KEMs only, and the transcripts involved.
- IND-CCA2 security if KDF behaves like a split-key pseudorandom function [GHP19]

# Key Combiners for Multiple Keys: Comparison

## Further Actions

**Comparison to the following two CFRG Documents on Hybrid KEMs**

- Hybrid PQ/T Key Encapsulation Mechanisms (draft-irtf-cfrg-hybrid-kems-07)
- Concrete Hybrid PQ/T Key Encapsulation Mechanisms (draft-irtf-cfrg-concrete-hybrid-kems-01)

## Your comments are welcome!


- Wang.guilin@Huawei.com
- haoyang.wang@sjtu.edu.cn


*Thanks!*