Workgroup: CFRG
Internet-Draft:
draft-ounsworth-cfrg-kem-combiners-05
Published: 31 January 2024
Intended Status: Informational
Expires: 3 August 2024
Authors: M. Ounsworth A. Wussler S. Kousidis
Entrust Proton BSI

Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs)

Abstract

The migration to post-quantum cryptography often calls for performing multiple key encapsulations in parallel and then combining their outputs to derive a single shared secret.

This document defines a comprehensible and easy to implement Keccak-based KEM combiner to join an arbitrary number of key shares, that is compatible with NIST SP 800-56Cr2 [SP800-56C] when viewed as a key derivation function. The combiners defined here are practical split-key PRFs and are CCA-secure as long as at least one of the ingredient KEMs is.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ounsworth-cfrg-kem-combiners/.

Discussion of this document takes place on the Crypto Forum Research Group (CFRG) Research Group mailing list (mailto:cfrg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cfrg/. Subscribe at https://www.ietf.org/mailman/listinfo/cfrg/.

Source for this draft and an issue tracker can be found at https://github.com/EntrustCorporation/draft-ounsworth-cfrg-kem-combiners.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 August 2024.

Copyright Notice

Table of Contents

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document is consistent with all terminology defined in [I-D.driscoll-pqt-hybrid-terminology].

1.1. Key Encapsulation Mechanisms

For the purposes of this document, we consider a Key Encapsulation Mechanism (KEM) to be any asymmetric cryptographic scheme comprised of algorithms satisfying the following interfaces [PQCAPI].

```
def kemKeyGen() -> (pk, sk)
def kemEncaps(pk) -> (ct, ss)
def kemDecaps(ct, sk) -> ss
```

where pk is public key, sk is secret key, ct is the ciphertext representing an encapsulated key, and ss is the shared secret.

KEMs are typically used in cases where two parties, hereby referred to as the "encapsulater" and the "decapsulater", wish to establish a shared secret via public key cryptography, where the decapsulater has an asymmetric key pair and has previously shared the public key with the encapsulater.

2. Introduction

The need for a KEM combiner function arises in three different contexts within IETF security protocols:

1. KEM / PSK hybrids where the output of a KEM is combined with a static pre-shared key.

2. Post-quantum / traditional hybrid KEMs where output of a post-quantum KEM is combined with the output of a classical key transport or key exchange algorithm.

3. KEM-based authenticated key exchanges (AKEs) where the output of two or more KEMs performed in different directions are combined.

This document normalizes a mechanism for combining the output of two or more KEMs.

2.1. KEM/PSK hybrids

As a post-quantum stop-gap, several IETF protocols have added extensions to allow for mixing a pre-shared key (PSK) into an (EC)DH based key exchange. Examples include CMS [RFC8696] and IKEv2 [RFC8784].

2.2. PQ/Traditional hybrid KEMs

A post-quantum / traditional hybrid key encapsulation mechanism (hybrid KEM) as defined in [I-D.driscoll-pqt-hybrid-terminology] as

PQ/T Hybrid Key Encapsulation Mechanism: A Key Encapsulation Mechanism (KEM) made up of two or more component KEM algorithms where at least one is a post-quantum algorithm and at least one is a traditional algorithm.

Building a PQ/T hybrid KEM requires a secure function which combines the output of both component KEMs to form a single output. Several IETF protocols are adding PQ/T hybrid KEM mechanisms as part of their overall post-quantum migration strategies, examples include TLS 1.3 [I-D.ietf-tls-hybrid-design], IKEv2 [I-D.ietf-ipsecme-ikev2-multiple-ke], X.509; PKIX; CMS [I-D.ounsworth-pq-composite-kem], OpenPGP [I-D.wussler-openpgp-pqc], JOSE / COSE (CITE once Orie's drafts are up).

The traditional algorithm may in fact be a key transport or key agreement scheme, but since simple transformations exist to turn both of those schemes into KEMs, this document assumes that all cryptograhpic algorithms satisfy the KEM interface described in Section 1.1.

2.3. KEM-based AKE

The need for a KEM-based authenticated key establishment arises, for example, when two communicating parties each have long-term KEM keys (for example in X.509 certificates), and wish to involve both KEM keys in deriving a mutually-authenticated shared secret. In particular this will arise for any protocol that needs to provide post-quantum replacements for static-static (Elliptic Curve) Diffie-Hellman mechanisms. Examples include a KEM replacement for CMP's DHBasedMac [I-D.ietf-lamps-cmp-updates].

3. KEM Combiner construction

A KEM combiner is a function that takes in two or more shared secrets ss_i and returns a combined shared secret ss.

```
ss = kemCombiner(ss_1, ss_2, ..., ss_n)
```

This document assumes that shared secrets are the output of a KEM, but without loss of generality they MAY also be any other source of cryptographic key material, such as pre-shared keys (PSKs), with PQ/PSK being a quantum-safe migration strategy being made available by some protocols, see for example IKEv2 in [RFC8784].

In general it is desirable to use a split-key PRF as a KEM combiner, meaning that the combiner has the properties of a PRF when keyed by any of its single inputs. The following simple yet generic construction can be used in all IETF protocols that need to combine the output of two or more KEMs:

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

Figure 1: general KEM combiner construction

where:

- KDF represents a suitable choice of a cryptographic key derivation function,
- k_i represent the constant-length input keys and is discussed in Section 3.2,
- fixedInfo is some protocol-specific KDF binding,
- counter parameter is instantiation-specific and is discussed in Section 4.
- outputBits determines the length of the output keying material,
- || represents concatenation.

In Section 4 several possible practical instantiations are listed that are in compliance with NIST SP-800 56Cr2 [SP800-56C]. The shared secret ss MAY be used directly as a symmetric key, for example as a MAC key or as a Key Encryption Key (KEK).

The values k_i can be processed individually, without requiring to store intermediate values except for the hash state and the protocol binding information required for fixedInfo.

3.1. Length encoding

All variable length string inputs s MUST be suffixed with the length, right-encoded using the rlen function, having the following construction:

Validity Conditions: 0 <= len(s) < 2^{2040}
1. Let x = len(s)
1. Let n be the smallest positive integer for which 2^{8n} > x
2. Let x_1, x_2, ..., x_n be the base-256 encoding of x satisfying:
    x = sum 28(n-i)x i, for i = 1 to n
3. Let O_i = uint8(x_i), for i = 1 to n
4. Let O_{n+1} = uint8(n)
5. rlen(s) = O_1 || O_2 || ... || O_n || O_{n+1}

This is compatible with the right_encode construction presented in [SP800-185], and encodes the length of the string s as a byte string in a way that can be unambiguously parsed from the end.

Right encoding is preferred to left encoding, since it provides the same security guarantees but allows encoding ciphertext where length is a priori unknown.

3.2. k_i construction

Following the guidance of Giacon et al. [GHP18], we wish for a KEM combiner that is CCA-secure as long as at least one of the ingredient KEMs is. In order to protect against chosen ciphertext attacks, it is necessary to include both the shared secret ss_i and its corresponding ciphertext ct_i. If both the secret share ss_i and the ciphertext ct_i are constant length, then k_i MAY be constructed concatenating the two values:

k_i = ct_i || ss_i

If ss_i or ct_i are not guaranteed to have constant length, it is REQUIRED to append the rlen encoded length when concatenating, prior to inclusion in the overall construction described in Figure 1:

k_i = ct_i || rlen(ct_i) || ss_i || rlen(ss_i)

Any protocols making use of this construction MUST either right-encode the length of all inputs ss_i and ct_i, or justify that any inputs will always be fixed length. In the case of a PSK the associated ciphertext is the empty string.

Including the ciphertext guarantees that the combined kem is IND-CCA2 secure as long as one of the ingredient KEMs is, as stated by [GHP18].

The ciphertext precedes the secret share, as memory-constrained devices can write c_i into the hash state and no further caching is required when streaming.

3.3. Note on the order of parameters

For a two-KEM instantiation, the construction is

KDF(counter || ct_1 || rlen(ct_1) || ss_1 || rlen(ss_1) || ct_2 || rlen(ct_2) || ss_2 || rlen(ss_2) || fixedInfo, outputBits)

The order of parameters is chosen intentionally to facilitate streaming implementations on devices that lack sufficient memory to hold the entirety of ct_1 or ct_2.

This construction aims to have two streaming-friendly properties. First, ct_i can be written to KDF's update interface as it is received and does not need to be stored, finally adding its corresponding ss_i once it is available. And second, the first KEM can be processed in its entirety and written to KDF's update interface before beginning to process the second KEM.

3.4. Protocol binding

The fixedInfo parameter is a fixed-format string containing some context-specific information. This serves to prevent cross-context attacks by making this key derivation unique to its protocol context.

The fixedInfo string MUST have a definite structure depending on the protocol where all parts strictly defined by the protocol specification.

```
fixedInfo = fixedInfo || s
```

Each fixed-length input string f MAY be directly used as input:

```
s = f ; f is guaranteed to have fixed length
```

Each variable-length input string v MUST be suffixed with a right-encoding of the length:

```
s = v || rlen(v) ; v may have variable length
```

fixedInfo MUST NOT include the shared secrets and ciphertexts, as they are already represented in the KDF input.

The parameter fixedInfo MAY contain any of the following information:

- Public information about the communication parties, such as their identifiers.

*The public keys or certificates contributed by each party to the key-agreement transaction.

*Other public information shared between communication parties before or during the transaction, such as nonces.

*An indication of the protocol or application employing the key-derivation method.

*Protocol-related information, such as a label or session identifier.

*An indication of the key-agreement scheme and/or key-derivation method used.

*An indication of the domain parameters associated with the asymmetric key pairs employed for key establishment.

*An indication of other parameter or primitive choices.

*An indication of how the derived keying material should be parsed, including an indication of which algorithm(s) will use the (parsed) keying material.

This is a non-comprehensive list, further information can be found in paragraph 5.8.2 of NIST SP800-56Ar3 [SP800-56A].

4. Practical instantiations

The KDF must be instantiated with cryptographically-secure choices for KDF. The following are RECOMMENDED Keccak-based instatiations, but other choices MAY be made for example to allow for future cryptographic agility. A protocol using a different instantiation MUST justify that it will behave as a split-key PRF, as required in [GHP18].

Each instance defines a function to be used as KDF, a hashSize to determine parameter size, and optionally a counter:

1. KDF = KMAC128, with hashSize = 128 bit.

2. KDF = KMAC256, with hashSize = 256 bit.

3. KDF = SHA3-256, with hashSize = 256 bit.

4. KDF = SHA3-512, with hashSize = 512 bit.

As justified in the security considerations, we recommend only Keccak-based instantiations because assuming there are no weaknesses found in the Keccak permutation, it behaves as a split-key PRF that

can be keyed by any input k_i. SHAKE is also not included in the list as it is not allowed by [SP800-56A] section 7, and does not provide any implementation advantage over KMAC.

KMAC constructions are RECOMMENDED over SHA-3, as KMAC offers a simple cSHAKE-based construction, with the advantage of returning an unrelated output when requesting a different outputBits KEK length.

4.1. KMAC based construction

Options 1 and 2 are KMAC-based, as specified in NIST SP 800-185 [SP800-185]. To instantiate the function:

* The context S MUST be the utf-8 string "KDF".

* The key K MUST be a context-specific string of at least hashSize bits, and it MAY be used as an additional option to perform context separation, in scenarios where fixedInfo is not sufficient.

* The parameter counter MUST be the fixed value 0x00000001, corresponding to a 32-bit binary big-endian representation of the number zero.

  - This aligns with [SP800-56C] where counter is initialized to 0x00000000 is Section 4.1 Step 3, but is first incremented in 6.1 before being used in a hash in 6.2.

To derive a shared secret ss of desired length, KMAC is called a single time with the input string X defined in Section 3 and length L being outputBits. This is compatible with the one-step KDF definition given in NIST SP800-56Cr2 [SP800-56C], Section 4.

4.2. Hash-and-counter based construction

Options 3 and 4 instantiate the KDF using SHA3, specified in NIST FIPS 202 [FIPS202]. To generate an outputBits long secret share ss:

* the counter MUST be initialized with the value 0x00000001.

* The hash is performed over the string defined in Section 3, and repeated ceil(outputBits/hashSize) times. For each iteration the counter MUST be increased by 0x01.

* The strings are concatenated ordered by counter.

* The leftmost outputBits are returned as ss.

An implementation MUST NOT overflow and reuse the counter and an error MUST be returned when producing more than 2^32 consecutive hashes.

5. IANA Considerations

None.

6. Security Considerations

The proposed instantiations in Section 4 are practical split-key PRFs since this specification limits to the use of Keccak-based constructions. The sponge construction was proven to be indifferentiable from a random oracle [BDPA08]. More precisely, for a given capacity c the indifferentiability proof shows that assuming there are no weaknesses found in the Keccak permutation, an attacker has to make an expected number of 2^(c/2) calls to the permutation to tell Keccak from a random oracle. For a random oracle, a difference in only a single bit gives an unrelated, uniformly random output. Hence, to be able to distinguish a key k, derived from shared keys k_i from a random bit string, an adversary has to correctly guess all key shares k_i entirely.

The proposed construction in Section 3 with the instantiations in Section 4 preserves IND-CCA2 of any of its ingredient KEMs, i.e. the newly formed combined KEM is IND-CCA2 secure as long as at least one of the ingredient KEMs is. Indeed, the above stated indifferentiability from a random oracle qualifies Keccak as a split-key pseudorandom function as defined in [GHP18]. That is, Keccak behaves like a random function if at least one input shared secret ss_i is picked uniformly at random. Our construction can thus be seen as an instantiation of the IND-CCA2 preserving Example 3 in Figure 1 of [GHP18], up to some reordering of input shared secrets ss_i and ciphertexts ct_i and their potential compression H(ss_i || ct_i) by a cryptographic hash function.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[ADKPRY22] Aviram, N., Dowling, B., Komargodski, I., Paterson, K. G., Ronen, E., and E. Yogev, "Practical (Post-Quantum)

Key Combiners from One-Wayness and Applications to TLS.", n.d., <https://eprint.iacr.org/2022/065>.

[BDPA08] Bertoni, G., Daemen, J., Peters, M., and G. Assche, "On the Indifferentiability of the Sponge Construction", 2008, <https://doi.org/10.1007/978-3-540-78967-3_11>.

[ETSI-QHKE] "Quantum-safe Hybrid Key Exchanges", ETSI TS 103 744 V1.1.1 , December 2020, <https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf>.

[FIPS202] "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", Federal information Processing Standards Publication (FIPS) 202 , 2015, <https://doi.org/10.6028/NIST.FIPS.202>.

[GHP18] Giacon, F., Heuer, F., and B. Poettering, "KEM Combiners", 2018, <https://doi.org/10.1007/978-3-319-76578-5_7>.

[I-D.driscoll-pqt-hybrid-terminology] D, F., "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-driscoll-pqt-hybrid-terminology-02, 7 March 2023, <https://datatracker.ietf.org/doc/html/draft-driscoll-pqt-hybrid-terminology-02>.

[I-D.ietf-ipsecme-ikev2-multiple-ke] Tjhai, C., Tomlinson, M., Bartlett, G., Fluhrer, S., Van Geest, D., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in IKEv2", Work in Progress, Internet-Draft, draft-ietf-ipsecme-ikev2-multiple-ke-12, 1 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-multiple-ke-12>.

[I-D.ietf-lamps-cmp-updates] Brockhaus, H., von Oheimb, D., and J. Gray, "Certificate Management Protocol (CMP) Updates", Work in Progress, Internet-Draft, draft-ietf-lamps-cmp-updates-23, 29 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cmp-updates-23>.

[I-D.ietf-tls-hybrid-design] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-09, 7 September 2023, <https://

datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-09>.

[I-D.ounsworth-pq-composite-kem] Ounsworth, M. and J. Gray, "Composite KEM For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-kem-02, 29 May 2023, <https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-kem-02>.

[I-D.wussler-openpgp-pqc] Kousidis, S., Roth, J., Strenzke, F., and A. Wussler, "Post-Quantum Cryptography in OpenPGP", Work in Progress, Internet-Draft, draft-wussler-openpgp-pqc-04, 30 January 2024, <https://datatracker.ietf.org/doc/html/draft-wussler-openpgp-pqc-04>.

[PQCAPI] Project, N. P.-Q. C., "PQC - API notes", November 2022, <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/example-files/api-notes.pdf>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[RFC8696] Housley, R., "Using Pre-Shared Key (PSK) in the Cryptographic Message Syntax (CMS)", RFC 8696, DOI 10.17487/RFC8696, December 2019, <https://www.rfc-editor.org/info/rfc8696>.

[RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/info/rfc8784>.

[SP800-185] Kelsey, J., Chan, S., and R. Perln, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash", NIST Special Publication 800-185 , 2016, <https://doi.org/10.6028/NIST.SP.800-185>.

[SP800-56A] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A , April 2018, <https://doi.org/10.6028/NIST.SP.800-56Ar3>.

[SP800-56C]
Barker, E., Chen, L., and R. Davis, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", NIST Special Publication 800-56C , August 2020, <https://doi.org/10.6028/NIST.SP.800-56Cr2>.

Acknowledgements

This document incorporates contributions and comments from a large group of experts. The authors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of electronic mail and VOIP traffic over the past years in pursuit of this document:

Douglas Stebila, Nimrod Aviram, and Andreas Huelsing.

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road – Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: mike.ounsworth@entrust.com

Aron Wussler
Proton AG
Switzerland

Email: aron@wussler.it

Stavros Kousidis
BSI
Germany

Email: kousidis.ietf@gmail.com