ETSI TS 103 744 V1.2.1 (2025-03)

TECHNICAL SPECIFICATION

CYBER;
Quantum-Safe Cryptography (QSC);
Quantum-safe Hybrid Key Establishment

Reference

RTS/CYBER-QSC-0019

Keywords

key exchange, quantum safe cryptography

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Hybrid Key Establishments are constructions that combine a traditional key establishment method, such as elliptic curve Diffie Hellman [1], with a quantum-safe key encapsulation mechanism, such as Module-Lattice-based Key Encapsulation Mechanism (ML-KEM) [11], into a single key establishment method. Hybrid key establishments are a migration technique to move to quantum-safe technology in advance of establishing full security assurance in the underlying post-quantum cryptographic scheme.

1 Scope

The present document specifies several methods for deriving cryptographic keys from multiple shared secrets. The shared secrets are established using existing traditional key establishment schemes, like Elliptic Curve Diffie-Hellman (ECDH) in NIST SP800-56Ar3 [1], and new quantum-safe Key Encapsulation Mechanisms (KEMs).

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] NIST SP800-56Ar3: "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography".

[2] IETF RFC 2104: "HMAC: Keyed-Hashing for Message Authentication".

[3] IETF RFC 5869: "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)".

[4] FIPS PUB 180-4: "Secure Hash Standard (SHS)".

[5] Void.

[6] NIST SP 800-186: "Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters".

[7] IETF RFC 5639: "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation".

[8] IETF RFC 7748: "Elliptic Curves for Security".

[9] NIST SP800-185: "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash".

[10] NIST SP800-56Cr2: "Recommendation for Key-Derivation Methods in Key-Establishment Schemes".

[11] FIPS 203: "Module-Lattice-Based Key-Encapsulation Mechanism Standard".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin: "Randomness Extraction and Key derivation Using the CBC, Cascade, and HMAC Modes", Crypto 04, LNCS 3152, pp. 494-510. Springer Verlag, 2004.

[i.2] Void.

[i.3] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, D. Stebila: "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange", IACR eprint 2018-903.

[i.4] Void.

[i.5] Simon D. R.: "On the power of quantum computation", SFCS 94 Proceedings of the 35th Annual Symposium on Foundations of Computer Science, November 1994, Pages 116-123.

[i.6] Shor P.W.: "Algorithms for quantum computation: discrete logarithms and factoring", SFCS 94: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, November 1994, Pages 124-134.

[i.7] NIST CAVP SP 800-56A: "ECC CDH Primitive Test Vectors".

[i.8] IETF RFC 8734 (2020): "Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3". RFC Editor.

[i.9] Void.

[i.10] Kwiatkowski K. (2024): "Post Quantum Cryptography KATs".

[i.11] Campagna M., Petcher A.: "Security of Hybrid Key Encapsulation", IACR eprint 2020-1364, November 2020.

[i.12] Campagna M., Petcher A.: "Security of Hybrid Key Establishment using Concatenation", IACR eprint 2023-972, June 2023.

[i.13] Void.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

asymmetric cryptography: cryptographic system that utilizes a pair of keys, a private key known only to one entity, and a public key that can be openly distributed without loss of security

big-endian: octet ordering that signifies "big-end", or most significant octet value is stored to the left, or at the lowest storage location

EXAMPLE: The decimal value 108591, which is 0x0001A82F as a hex encoded 32-bit integer, is encoded as a length 4 octet string as 0001A82F.

cryptographic hash function: function that maps a bit string of arbitrary length to a fixed length bit string (*message digest* or *digest* for short)

NOTE: Hash functions are designed to satisfy the following properties:

1) (One-way) It is computationally infeasible to find any input that maps to any pre-specified output.

2) (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output.

cryptographic key: binary string used as a secret by a cryptographic algorithm

EXAMPLE: AES-256 requires a random 256-bit string as a secret key.

entity: person, device, or system that is executing the steps of a process

NOTE: Steps of one of the processes defined or referenced in the present document.

info: octet string set by the application as additional information

EXAMPLE: An application specific value like an ASCII encoded string, e.g. info = "ETSI_QSHKE_TEST_VECTORS_V_1_2".

key agreement scheme: key establishment procedure in which the resultant secret keying material is a function of contributions of the entities participating, such that no entity can predetermine the value of the secret keying material independently of the other entities' contributions

key derivation: process to derive key material from one or more shared secrets

key encapsulation mechanism: set of methods to establish a shared secret key between two parties

key establishment/exchange method: cryptographic procedure by which cryptographic keys are established between two parties

label: octet string that specifies a separation of use for the instance of the key derivation or exchange, such as a random nonce.

message digest/digest: fixed-length output of a cryptographic hash function over a variable length input

octet string: ordered sequence of octets/bytes consisting of 8-bits each

private key: key in an asymmetric cryptographic scheme that is kept secret

public key: key in an asymmetric cryptographic scheme that can be made public without loss of security

public key cryptography: See asymmetric cryptography.

random oracle: theoretical black box that responds to every unique query with a uniformly random selection from the set of possible responses, with repeated queries receiving the same response

security level: value n for which the best-known attack against breaking the security properties of a cryptographic algorithm requires 2^n operations.

NOTE: Sometimes also referred to as *bit-strength*.

shared secret: secret value that has been computed using a key-establishment scheme

3.2 Symbols

For the purposes of the present document, the following symbols apply:

A || B	The concatenation of binary strings A followed by B
∅	A zero-length octet string
$[x]_n$	An integer value *x* expressed as an *n*-bit integer
$\lceil q \rceil$	The least integer value *x* greater than or equal to *q*
len(A)	The number of octets in an octet string *A*
hash( )	A cryptographic hash function
digest_len	The length in octets of a hash function's digest
block_len	The block length in octets of a hash function's block size
C	A ciphertext value created by a KEM
d	A private key for elliptic curve cryptography
k	A cryptographic secret or key
P/R	A public key for an asymmetric cryptographic scheme
psk	A pre-shared key
Q	A public key for elliptic curve cryptography

sk	A private key for an asymmetric cryptographic scheme

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CDH	Cofactor Diffie-Hellman
CID	Ciphersuite IDentifier
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
HKDF	HMAC-based Key Derivation Function
HMAC	Hash-based Message Authentication Code
IND-CCA	INDistinguishability under Chosen-Ciphertext Attacks
IND-CPA	INDistinguishability under Chosen-Plaintext Attacks
KDF	Key Derivation Function
KEM	Key Encapsulation Mechanism
KMAC	Keccak Message Authentication Code
LNCS	Lecture Notes in Computer Science
MA	Message from entity A
MB	Message from entity B
ML-KEM	Module-Lattice-Based Key Encapsulation Mechanism
NIST	National Institute of Standards and Technology
OW-CCA	One Way Chosen Ciphertext Attack
OW-CPA	One-Way Chosen-Plaintext Attack
PRF	PseudoRandom Function
QKD	Quantum Key Distribution
RSA	Rivest, Shamir and Adelman
SP	Special Publication
SSH	Secure Shell
TLS	Transport Layer Security

4 Purpose of quantum-safe hybrid key establishment

4.1 Status of quantum-safe key encapsulation mechanisms

NIST has initiated a process of analysing and standardizing one or more new quantum-safe key encapsulation mechanisms suitable to replace traditional key establishment schemes. At the time of the present document, there is one FIPS approved standard, FIPS 203 [11].

The present document addresses the following cases:

1) One or more key exchange method establishes a shared secret from which randomness extraction is necessary.

2) One or more key exchange method incorporates a hash-based key derivation function prior to use within the hybrid method defined in the present document.

Quantum-safe hybrid key establishment specified in the present document ensures that the derived key is at least as secure as the maximum security of the key establishment schemes. The resulting hybrid scheme will remain secure if one of the key establishment schemes remains secure.

Quantum Key Distribution (QKD) provides an alternative method of establishing a shared secret between two entities using quantum mechanics. The scope of the present document is limited to elliptic curve Diffie-Hellman and quantum-safe key encapsulation mechanisms.

5 Architecture for quantum-safe hybrid key establishment

5.1 Functional entities

There are two entities defined for quantum-safe hybrid key establishment, an Initiator *A* that initiates a key establishment scheme, and a Responder *B* who responds to the request. The entities communicate over a network medium.

EXAMPLE: Examples of such mediums are: ethernet, wireless and cellular networks.

Figure 1: Communicating entities *A* and *B*

5.2 Information relationships (reference points)

The network media over which the Initiator and Responder communicate will have a packet formatting scheme that allows the encoding and transmission of octet (byte) strings. The Initiator and Responder will exchange messages, where each message is an octet string that can span multiple packets. *MA* denotes a message from *A* to *B*, and *MB* denotes a message sent from *B* to *A*.

A may initiate a hybrid key establishment by the transmission of a message to *B*. *B* responds to this message. The exchange between the entities can consist of a single message or multiple rounds of messages.

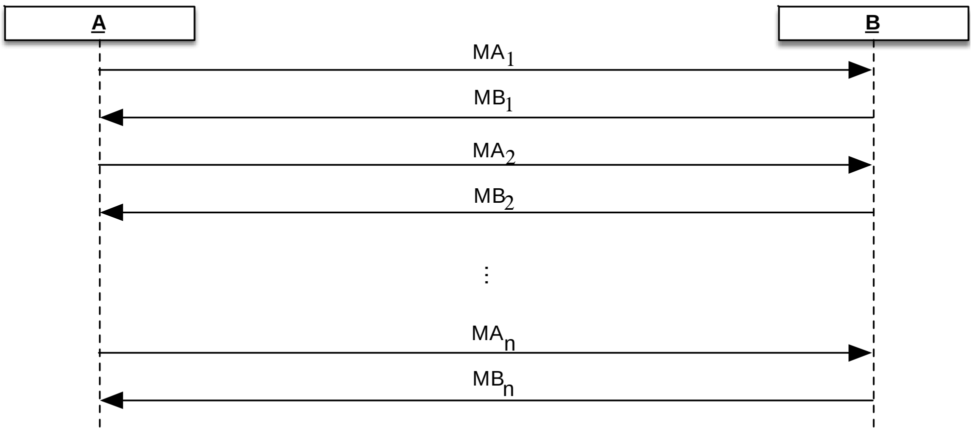

Figure 2: Messages exchanged between entities *A* and *B*

The transcript of the key establishment is the list of all messages exchanged between *A* and *B*, in the sequence order they were sent:

$$transcript = (MA_1, MB_1, MA_2, MB_2, \ldots MA_n, MB_n)$$

In other embodiments, *B* may be in possession of authentic public keys belonging to *A*. The exchange of messages may consist solely of messages from *B* to *A*.

6 Introductory information

6.1 Introduction

Quantum-safe hybrid key establishment combines a traditional key establishment scheme, like ECDH and a quantum-safe Key Encapsulation Mechanism (KEM). Hybrid key establishment schemes specified in the present document use two or more shared secrets to derive cryptographic key material using a key derivation function. The key derivation functions for hybrid key establishment specified in the present document provide both the key expansion property and random extraction as per Crypto 04, LNCS 3152 [i.1].

6.2 Notation

6.2.1 Radix

The prefix "0x" indicates hexadecimal numbers.

6.2.2 Conventions

The assignment operator "=", as used in several programming languages:

<variable> = <expression>

means that <*variable*> assumes the value that <*expression*> had before the assignment took place. For instance:

$$x = x + y + 3$$

means:

(new value of *x*) becomes (old value of *x*) + (old value of *y*) + 3.

6.2.3 Bit/Byte ordering

All data variables are represented with the most significant bit (or byte) on the left-hand side and the least significant bit (or byte) on the right-hand side. Where a variable is broken down into a number of sub-strings, the left most (most significant) sub-string is numbered 0, the next most significant is numbered, 1 and so on, through to the least significant.

EXAMPLE: An n-bit MESSAGE is subdivided into 64-bit substrings M_0, M_1, …, M_i so if the message is:

0x0123456789ABCDEFFEDCBA987654321086545381AB594FC28786404C50A37…

then:

M_0 = 0x0123456789ABCDEF
M_1 = 0xFEDCBA9876543210
M_2 = 0x86545381AB594FC2
M_3 = 0x8786404C50A37…

6.2.4 Integer encoding

Integers are represented in the bit/byte ordering defined in clause 6.2.3. The most significant bit (or byte) on the left-hand side and the least significant bit (or byte) on the right-hand side.

EXAMPLE: A 32-bit integer of the value I = 37 is encoded as:

I = 0x00000025

NOTE: This is big-endian or network byte ordering.

7 Cryptographic primitives

7.1 Hash functions (hash)

A hash function maps an arbitrary length bit string *(input)* to a fixed length (*digest_len*) octet string output *(digest)*:

$$digest = hash(input)$$

Approved hash functions for the purpose of the present document shall be limited to those in the following list:

- SHA-256, SHA-384 as defined in FIPS PUB 180-4 [4].

7.2 Context formatting function (*f*)

7.2.1 Context formatting function (*f*) description

The context formatting functions used in the present document take a list of inputs and return an octet string. A generic calling interface to the context function *f* used in the present document is defined in the present clause:

$$context = f(val1, val2, \ldots)$$

where the parameters and output shall be defined as follows.

Input:

$val_1, val_2, \ldots, val_n$ - an ordered sequence of octet strings each of length less than 2^{32} octets, where $n > 0$.

Output:

context - an octet string representing the context value.

7.2.2 Concatenate-based context formatting function

The present clause defines a concatenate-based context formatting function. The concatenate-based context formatting function takes an ordered sequence of octet strings and converts them into a length-delimited single octet string. The concatenate-based context formatting function *f* has the following calling interface:

$$context = cb\_f(val_1, val_2, \ldots)$$

where the parameters, procedure and output shall be as follows.

Input:

$val_1, val_2, \ldots, val_n$ - an ordered sequence of octet strings each of length less than 2^{32} octets, where $n > 0$.

Process:

1) *Set context* = $\varnothing$.

2) For $i = 1, \ldots, n$.

   a) Set $len_i = len(val_i)$, returns the length of val_i in octets.

      i) If $len_i > 2^{32} - 1$, return error.

      ii) $L_i = [len_i]_{32}$ - a 32-bit integer value expressed as 4 octets.

   b) Set $context = context \mid\mid L_i \mid\mid val_i$.

3) Return *context*.

Output:

context - an octet string representing the context value or an error.

The concatenate-based formatting function is designed to be used in conjunction with the KMAC variety of KDFs. HKDF implementations do not perform efficiently when instantiated with a concatenate-based context formatting function. The *context* value is passed as the 'info' input to the HKDF function. See OpenSSL's documentation on length limits for *context* - https://www.openssl.org/docs/man1.1.1/man3/EVP_PKEY_CTX_add1_hkdf_info.html. IETF RFC 5869 [3] does not limit the length of the 'info' input value; however, for efficiency reasons, it is recommended to keep the length relatively short since it is processed through a hash function in every execution of HKDF expand step.

7.2.3 Concatenate-and-hash-based context formatting function

The present clause defines a concatenate-and-hash context formatting function. The concatenate-and-hash context formatting function takes an ordered sequence of octet strings and converts them into a length delimited octet string. The concatenate-and-hash-based context formatting function *f* has the following calling interface:

$$context = cahb\_f(val_1, val_2, \ldots)$$

where the prerequisite, parameters, procedure and output shall be as follows.

Prerequisite:

hash - an approved hash function as per clause 7.1 that produces a *digest_len*-length string of octets (*digest_len* in {32, 48}).

Input:

$val_1, val_2, \ldots, val_n$ - an ordered sequence of octet strings each of length less than 2^{32} octets, where $n > 0$.

Process:

1) Set $buffer = \varnothing$.

2) For $i = 1, \ldots, n$.

   a) Set $len_i = len(val_i)$, returns the length of val_i in octets.

      i) If $len_i > 2^{32} - 1$, return error.

      ii) $L_i = [len_i]_{32}$ - a 32-bit integer value expressed as 4 octets.

   b) Set $\mathrm{buffer} = \mathrm{buffer} \parallel \mathrm{L_i} \parallel \mathrm{val_i}$.

3) Set $context = hash(buffer)$.

4) Return context.

Output:

context - a *digest_len* octet string representing the context value or an error.

NOTE: In practice, the concatenate-and-hash based context format function may save memory allocation, and make use of the *context* value more efficiently if there are cases where it is used iteratively.

7.3 PseudoRandom Function (PRF)

7.3.1 PRF description

A PseudoRandom Function generates output from a random (or secret) seed such that the output is computationally indistinguishable from truly random output. A generic calling interface to the PRF used in the present document is defined in the present clause:

$$output = PRF(secret, val_1, val_2, \ldots)$$

where the parameters and output shall be defined as follows.

Input:

secret - an octet string that constitutes the input for the pseudorandom function.

$val_1, val_2, \ldots, val_n$ - an ordered sequence of octet strings each of length less than 2^{32} octets, where $n > 0$.

Output:

output - the pseudorandom output, length dependent on the primitive used in the PRF.

Approved PRF functions for the purpose of the present document shall be limited to HMAC as defined in IETF RFC 2104 [2] with a SHA2 hash function from clause 7.1, or KMAC128 or KMAC256 as defined in NIST SP800-185 [9].

7.3.2 PRF to HMAC mapping

This clause specifies a mapping from the PRF definition to HMAC with a specified hash function.

HMAC shall be as defined in IETF RFC 2104 [2] with an approved SHA2 hash function from clause 7.1 and has a calling interface:

$$output = HMAC(secret, data)$$

where the prerequisite, parameters, and output shall be defined as follows.

Prerequisite:

cahb_f - a context formatting function as defined in clause 7.2.3.

Input:

secret - an octet string that constitutes the input for the pseudorandom function.

data - an octet string that is included in the HMAC.

Output:

output - pseudorandom output of a fixed length (of the underlying hash function).

The mapping shall be defined as follows:

$$output = PRF\ (secret, val_1, val_2, \ldots) = HMAC(secret, cahb\_f(val_1, val_2, \ldots)).$$

7.3.3 PRF to KMAC mapping

This clause specifies a mapping from the PRF definition to KMAC function, KMAC128 or KMAC256. KMAC shall be as defined in NIST SP800-185 [9] and has a calling interface:

$$output = KMAC(K, X, L, S)$$

where the prerequisite, parameters, and output shall be defined as follows.

Prerequisite:

cb_f - a context formatting function as defined in clause 7.2.2.

Input:

K - a key bit string of any length, at least security strength in length.

X - an input bit string of any length.

L - is an integer representation of the output length in bits.

S - is an optional customization bit string of any length.

Output:

output - pseudorandom output of bit length *L*.

The mapping shall be defined as follows:

$$\text{output} = \text{PRF (secret, } val_1, val_2, \ldots) = \text{KMAC (secret, cb\_f(}val_1, val_2, \ldots\text{), kmac\_outlen, } \varnothing),$$

where *kmac_outlen* is set to 32 or 48 octets for KMAC128 and KMAC256, respectively, corresponding to the minimum security level of the underlying key establishment schemes, and ∅ represents an empty octet string.

If a *secret* value is not specified or is the empty string, then a *secret* value of a zero string of 164 octets shall be used for KMAC128, and a *secret* value of a zero string of 132 octets shall be used for KMAC256. This shall only happen in the case when a *psk* is not used in the cascade combiner, see clause 8.3.3.

7.4 Key Derivation Functions (KDFs)

7.4.1 KDF description

The key derivation functions used in the present document are derived from PRFs. A generic calling interface to the KDFs used in the present document is defined in the present clause:

$$key\_material = KDF(secret, label, context, length)$$

where the parameters and output shall be defined as follows.

Input:

secret - an octet string that constitutes the input for the key derivation function.

label - an optional octet string that specifies a separation of use for the instance of the key derivation or exchange, such as a random nonce.

context - an octet string that constitutes an input for the key derivation function. It may include identity information specific to the entities deriving keys or exchanged messages.

length - the length in octets of the derived keying material.

Output:

key_material - the derived keying material of length *length*.

Approved KDF functions for the purpose of the present document shall be limited to HKDF as defined in IETF RFC 5869 [3] with an approved SHA2 hash function from clause 7.1, or KMAC128 or KMAC256 as defined in NIST SP800-185 [9].

NOTE: The KDFs selected for this specification are based on meeting the random oracle property used in the security proof [i.12], and their availability in existing cryptographic modules.

7.4.2 KDF to HKDF mapping

This clause specifies a mapping from the KDF definition to HKDF with a SHA2 hash function from clause 7.1.

HKDF shall be as defined in IETF RFC 5869 [3] and has a calling interface:

key_material = HKDF(secret, salt, info, length)

where the parameters and output shall be defined as follows.

Input:

secret - an octet string that constitutes the input for the key derivation function. It shall be present.

salt - optional salt value (a non-secret random value); it may be present. If it is not present it shall be set to a *digest_len* octet string of zero values. The *digest_len* is defined by the underlying hash function.

info - an octet string that contains application specific information. It may be a zero-length string.

length - the length in octets of the derived keying material. It shall be present.

Output:

key_material - the derived keying material of length *length*.

The mapping shall set *salt = label*, and info = context, as follows:

key_material = KDF(secret, label, context, length) = HKDF(secret, label, context, length).

7.4.3 KDF to HMAC mapping

This clause specifies a mapping from the KDF definition to an HMAC implementation of the NIST SP800-56Cr2 [10] One step Key Derivation Function using HMAC. HMAC shall be as defined in IETF RFC 2104 [2] with an approved SHA2 hash function from clause 7.1 and has a calling interface:

output = HMAC(secret, data)

where the prerequisite, parameters, and output shall be defined as follows.

Input:

secret - an octet string that constitutes the input for the pseudorandom function.

data - an octet string that is included in the HMAC.

Output:

output - pseudorandom output of a fixed length (of the underlying hash function).

The mapping shall set the *secret = label* and *data = counter* || *secret* || *context*, where *HMAC(label, counter* || *secret* || *context)* is iteratively executed to generate at least *length* output octets, with a 32-bit *counter* value. The streamed output is truncated (if needed) to output *key_material* of exactly *length*-octets. The mapping adheres to the following process.

Process:

1) Let $n = \lceil length/digest\_len \rceil$.

2) If $n > 2^{32} - 1$ return error.

3) If len(*secret* || *context*) > (*block_len* - 4) return error.

4) Set $buffer = \varnothing$.

5) For *counter = 1 to n*.

   a) Set $buffer = buffer \,||\, HMAC(label, [counter]_{32} \,||\, secret \,||\, context)$, $[counter]_{32}$ a 32-bit integer value expressed as 4 octets.

6) Set *key_material* = left most *length*-octets of *buffer*.

7) Return *key_material*.

If a *label* value is not specified, then a *label* value of a zero string of *block_len* octets shall be used.

7.4.4 KDF to KMAC mapping

This clause specifies a mapping from the KDF definition to KMAC implementation of the NIST SP800-56Cr2 [10] One step Key Derivation Function using KMAC128 or KMAC256. KMAC shall be as defined in NIST SP800-185 [9] and has a calling interface:

$$key\_material = KMAC(K, X, L, S)$$

where the parameters and output shall be defined as follows.

Input:

K - a key bit string of any length, at least security strength in length.

X - an input bit string of any length.

L - is an integer representation of the output length in bits.

S - is an optional customization bit string of any length.

Output:

key_material - the derived keying material of bit length *L*.

The mapping shall set $K = label$, $X = counter \,||\, secret \,||\, context$ and $L = length \times 8$, as follows:

$key\_material = KDF(secret, label, context, length) = KMAC(label, counter \,||\, secret \,||context, length \times 8, \text{"KDF"})$,

with big-endian 4-byte unsigned integer counter = 0x00000001.

If a *label* value is not specified, then a *label* value of a zero string of 164 octets shall be used for KMAC128, and a *label* value of a zero string of 132 octets shall be used for KMAC256, as specified in NIST SP800-56Cr2 [10].

NOTE 1: This specification requires the lengths of the bit strings *K*, *X* and *S* to be multiples of 8.

NOTE 2: The requirement that S = "KDF" is part of the NIST SP800-56Cr2 [10] requirement on the use of KMAC as a key derivation function.

7.5 Elliptic Curve Diffie-Hellman (ECDH)

7.5.1 ECDH description

One of the key-establishment schemes shall be Elliptic Curve Diffie-Hellman defined in clause 5.7.1.2 of NIST SP800-56Ar3 [1]. A shared secret *k* is computed between two entities over the same elliptic curve domain parameters. The key-establishment scheme, for a given set of elliptic curve domain parameters, consists of a pair of algorithms:

- *ECKeyGen( )*, a probabilistic algorithm that returns a private and public key pair *(d, Q)*.
- $ECDH(d_A, Q_B)$, a deterministic algorithm that takes d_A, entity A's private key, and Q_B, entity B's public key, as input and computes a shared secret *k*, or ⊥ an error indicator.

Similarly, party B computes the same shared secret by computing $ECDH(d_B, Q_A)$.

7.5.2 Elliptic curve domain parameters

The elliptic curve parameters used shall be one of the following:

- NIST P-384/secp384r1 defined in clause G.1.3 of NIST SP 800-186 [6].
- NIST P-256/secp256r1 defined in clause G.1.2 of NIST SP 800-186 [6].
- brainpoolP384r1 defined in clause 3.6 of IETF RFC 5639 [7].
- brainpoolP256r1 defined in clause 3.4 of IETF RFC 5639 [7].
- Curve448 defined in clause 4.2 of IETF RFC 7748 [8].
- Curve25519 defined in clause 4.1 of IETF RFC 7748 [8].

7.6 Key Encapsulation Mechanisms (KEMs)

7.6.1 KEM description

The present clause specifies the basic properties of a Key Encapsulation Mechanism (KEM) as a tuple of algorithms (*KEMKeyGen*, *Encaps*, *Decaps*) associated with a key space *K*:

- *KEMKeyGen( )*, a probabilistic algorithm that returns a private and public key pair *(sk, P)*, or ⊥ an error indicator.
- *Encaps(P)*, a probabilistic algorithm that takes a public key *P* as input and outputs a cryptographic key from the key space *K* and an associated ciphertext encapsulation using the public key *P*, denoted by *(k, C)*, or ⊥ an error indicator.
- *Decaps(sk, C)*, a deterministic algorithm that takes a private key *sk* and ciphertext *C* as input and outputs a cryptographic key *k* from the key space *K*, or a rejection key *k'*.

All KEMs shall provide OW-CPA security (see annex B).

7.6.2 Post-quantum KEMs

The present clause specifies the basic properties of post-quantum KEMs as a tuple of algorithms. All post-quantum KEMs shall expose a tuple of algorithms matching the KEM properties defined in clause 7.6.1. The current list of approved post-quantum KEMs consists of the FIPS-approved ML-KEM [11].

7.7 Primitive parameter sets

7.7.1 Parameter set description

The present clause specifies a parameter set. Parameter sets consist of tuples of specific choices for a key derivation function, elliptic curve Diffie-Hellman and quantum-safe key encapsulation mechanism instances. Parameters sets of are denoted KDF_CURVE_KEM.

The following clause restricts the parameter sets to enable greater interoperability.

The PRF and context formatting function are defined from the KDF function. When the KDF is instantiated as HKDFwHash or HMACwHash, the PRF shall be HMAC and the context formatting function will be *cahb_f* with the same hash function. When the KDF is KMAC#, the PRF shall be KMAC# and the context formatting function will be *cb_f*.

7.7.2 Parameter sets

The present clause specifies sets of primitives. The parameter sets are defined by a triple, a key derivation mechanism, a first key establishment scheme, and a second key establishment scheme. An implementation shall use one of the following parameter sets.

HKDFwSHA256_P256_ML-KEM-512,

HKDFwSHA256_ X25519_ML-KEM-512,

HKDFwSHA256_PBP256_ML-KEM-512,

HMACwSHA256_P256_ML-KEM-512,

HMACwSHA256_ X25519_ML-KEM-512,

HMACwSHA256_PBP256_ML-KEM-512,

KMAC128_P256_ML-KEM-512,

KMAC128_ X25519_ML-KEM-512,

KMAC128_ PBP256_ML-KEM-512,

HKDFwSHA256_P256_ML-KEM-768,

HKDFwSHA256_ X25519_ML-KEM-768,

HKDFwSHA256_ PBP256_ML-KEM-768,

HMACwSHA256_P256_ML-KEM-768,

HMACwSHA256_ X25519_ML-KEM-768,

HMACwSHA256_ PBP256_ML-KEM-768,

KMAC128_P256_ML-KEM-768,

KMAC128_ X25519_ML-KEM-768,

KMAC128_ PBP256_ML-KEM-768,

HKDFwSHA384_P384_ML-KEM-768,

HKDFwSHA384_ X448_ML-KEM-768,

HKDFwSHA384_ PBP384_ML-KEM-768,

HMACwSHA384_P384_ML-KEM-768,

HMACwSHA384_ X448_ML-KEM-768,

HMACwSHA384_ PBP384_ML-KEM-768,

KMAC256_P384_ML-KEM-768,

KMAC256_ X448_ML-KEM-768,

KMAC256_ PBP384_ML-KEM-768,

HKDFwSHA384_P384_ML-KEM-1024,

HKDFwSHA384_ X448_ML-KEM-1024,

HKDFwSHA384_ PBP384_ML-KEM-1024,

HMACwSHA384_P384_ML-KEM-1024,

HMACwSHA384_ X448_ML-KEM-1024,

HMACwSHA384_ PBP384_ML-KEM-1024,

KMAC256_P384_ML-KEM-1024,

KMAC256_ X448_ML-KEM-1024,

KMAC256_ PBP384_ML-KEM-1024.

8 Hybrid key establishment schemes

8.1 General

8.1.1 Key establishment abstraction

The present clause specifies how to combine two or more key establishment schemes into a hybrid key establishment scheme, under the assumption that all KEM schemes provide at least OW-CPA security.

NOTE: A KEM that provides IND-CPA or IND-CCA also provides OW-CPA. (See annex B for additional details.)

A key establishment mechanism consists of three functions:

- *KeyGen( )*, a key generation function that produces a private key *sk* and public key *P*;
- *ResponseFunc(P)*, a responder function which produces a shared secret *k* and response value *R*, or ⊥ an error indicator;
- *ReceiveFunc(sk, R)*, a receiving function on the private key *sk* and the response value *R* to compute the shared secret *k*, or ⊥ an error indicator.

If *ResponseFunc* returns an error indicator, B shall respond with an error message and terminate the process.

If A receives an error message from B, or if *ReceiveFunc* returns an error indicator, A shall terminate the process.

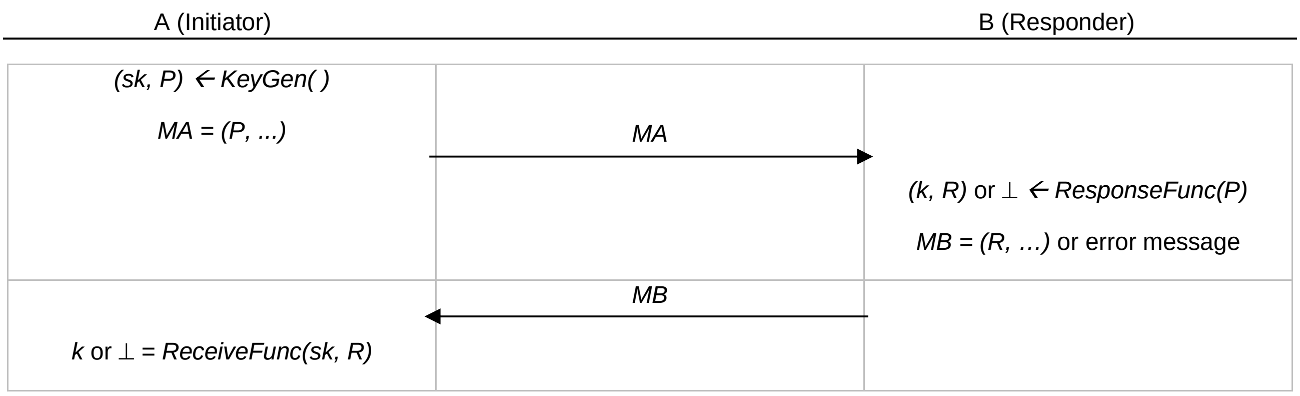

Figure 3: Key establishment abstraction

MA - shall be an octet string containing an encoding of one or more exchanged public keys from Initiator to Responder. *MA* may include session negotiation information.

MB - shall be an octet string containing an encoding of one or more response values. *MB* may include session negotiation information.

Messages shall be protected from unauthorized modification. Messages may be protected using digital signatures from a signing key associated with the sender's identity signed by a trusted third-party certificate authority.

Collectively such an instance of a key establishment is named a *key establishment transaction*.

This is designed to support the major use case of ECDHE with a quantum-safe KEM. Functional mappings between the naming convention and the calling abstractions are provided in clauses 8.1.2 and 8.1.3.

8.1.2 Key establishment abstraction to ECDHE

This clause specifies the mapping of the key establishment abstraction interface to ECDHE:

$$(sk, P) = KeyGen(\ ) = ECKeyGen(\ ).$$

The *ResponseFunc* shall be a combination of the *ECKeyGen( )* function and the *ECDH( )* function as follows.

ResponseFunc(P):

1) $(sk', R) = ECKeyGen(\ )$.

2) k or $\perp = ECDH(sk', P)$.

3) Return (k, R) or $\perp$.

The final *ReceiveFunc* shall be the *ECDH( )* function, k or $\perp = ReceiveFunc(sk, R) = ECDH(sk, R)$, where R is the public key (point) of the Responder sent in the message *MB*.

8.1.3 Key establishment abstraction to KEM

This clause specifies the mapping of the key establishment abstraction to a KEM:

$$(sk, P) = KeyGen(\ ) = KEMKeyGen(\ ).$$

The *ResponseFunc* shall be the *Encaps( )* function, (k, C) or $\perp = ResponseFunc(P) = Encaps(P)$.

The *ReceiveFunc* shall be the *Decaps( )* function, k or $\perp = ReceiveFunc(sk, C) = Decaps(sk, C)$, where C is the ciphertext value, generated by the Responder and sent in the message *MB*.

8.2 Concatenate hybrid key establishment scheme

8.2.1 Concatenate hybrid key establishment scheme - ephemeral

This clause specifies the concatenate hybrid key establishment scheme using ephemeral keys. The key establishment description of Figure 3 is extended to exchange a pair of public keys in a single message and multiple response values in a single message. The concatenate hybrid key establishment scheme is constructed as depicted in Figure 4.

A (Initiator)		B (Responder)
$(sk_1, P_1) = KeyGen_1(\,)$ $(sk_2, P_2) = KeyGen_2(\,)$ $MA = (P_1, P_2, ...)$	*MA* →	(k_1, R_1) or $\perp = ResponseFunc_1(P_1)$ (k_2, R_2) or $\perp = ResponseFunc_2(P_2)$ $MB = (R_1, R_2, \ldots)$ or error message
k_1 or $\perp = ReceiveFunc_1(sk_1, R_1)$ k_2 or $\perp = ReceiveFunc_2(sk_2, R_2)$	← *MB*	
$key\_material = CatKDF(\ \ldots\ )$		$key\_material = CatKDF(\ \ldots\ )$

Figure 4: Concatenate hybrid key establishment - ephemeral

If any $ResponseFunc_i$ returns an error indicator, B shall respond with an error message and terminate the process.

If *A* receives an error message, *A* shall terminate the process. If any $ReceiveFunc_i$ returns an error indicator, *A* shall terminate the process.

MA - shall be an octet string containing an encoding of exchanged public keys P_i from Initiator to Responder. *MA* may include session negotiation information, such as label contribution values.

MB - if *MB* is not an error message, it shall be an octet string containing an encoding of the response values R_i. *MB* may include session negotiation information, such as label contribution values.

Messages shall be protected from unauthorized modification. Messages may be protected using digital signatures from a signing key associated with the sender's identity signed by a trusted third-party certificate authority.

8.2.2 Concatenate hybrid key establishment scheme - static

This clause specifies the concatenate hybrid key establishment scheme using static keys. The key establishment description of Figure 3 is modified to obtain a pair of static public keys in a trusted manner in a single message and a pair of response values in a single message. The roles of Initiator and Responder are replaced with Recipient and Sender. A hybrid key establishment scheme using static keys is constructed as depicted in Figure 5.

A (Recipient)		B (Sender)
$(sk_1, P_1), (sk_2, P_2), \ldots$	$MA = (P1, P2, \ldots)$ →	*Obtain authentic static public keys* $MA = (P1, P2, \ldots)$ (k_1, R_1) or $\perp \leftarrow ResponseFunc_1(P_1)$ (k_2, R_2) or $\perp \leftarrow ResponseFunc_2(P_2)$ $MB = (R_1, R_2, \ldots)$ or error message
k_1 or $\perp = ReceiveFunc_1(sk_1, R_1)$ k_2 or $\perp = ReceiveFunc_2(sk_2, R_2)$	← MB	
$key\_material = CatKDF(\ldots)$		$key\_material = CatKDF(\ldots)$

Figure 5: Concatenate hybrid key establishment - static

If any $ResponseFunc_i$ returns an error indicator, B shall terminate the process.

If any $ReceiveFunc_i$ returns an error indicator, *A* shall terminate the process.

MA - Prior to or during the key-establishment, the Sender (B) receives Recipient's (A's) static public keys and additional values such as labels in a trusted manner.

MB - if *MB* is not an error message, it shall be an octet string containing an encoding of the response values R_i. *MB* may include session negotiation information. In the case where more than two key establishment schemes are being used, *MB* shall contain all of the corresponding public keys and ciphertexts.

Messages shall be protected from unauthorized modification. Messages may be protected using digital signatures from a signing key associated with the sender's identity signed by a trusted third-party certificate authority.

8.2.3 Concatenate hybrid key combiner - CatKDF

This clause specifies the concatenate hybrid key combiner using the concatenate KDF CatKDF, as denoted in Figure 4 and Figure 5.

The CatKDF( ) mode shall be defined as follows:

Fixed values:

KDF - The key derivation function being used from clause 7.4.

f - A context formatting function being used from clause 7.2. When KDF is instantiated as HKDF with hash, the underlying hash function is applied to the concatenate-and-hash context formatting function.

k_len - an integer value denoting the length of keys such as *psk*; *k_len* is *digest_len* when CatKDF is instantiated with HKDF or HMAC and *kmac_outlen* when CatKDF is instantiated as KMAC. *kmac_outlen* is set to 32 or 48 octets for KMAC128 and KMAC256, respectively.

Input:

psk - a pre-shared secret key of length *k_len*. It may be present. If not present, this value shall be the empty octet string, ∅.

(k_1, k_2) - *a pair* of octet strings containing shared secrets k_i, derived through a hybrid key establishment, see Figure 4 and Figure 5.

MA, *MB* - octet string of a pair of exchanged messages in establishment of the shared secrets k_i.

info - an optional octet string set by the application as additional information.

label - an optional octet string that specifies a separation of use for the instance of the key establishment. The label values should be constructed from exchanged label contribution values in messages *MA* and *MB*. If a label value is used it shall be of a fixed agreed length. Any labels used in the key derivation function should not be provided as an argument to the same hash function for another purpose in the application.

length - the length in octets of the derived key material *key_material*.

Process:

1) Form $secret = psk \mid\mid k_1 \mid\mid k_2$.

2) Set *context = f(info, MA, MB),* where *f* is a context formatting function.

3) *key_material = KDF(secret, label, context, length).*

4) Return *key_material*.

Output:

key_material - derived key material.

An implementation shall fix the order of the shared secrets k_i according to the order of the key establishment schemes defined in clause 7.7.2.

A pre-shared key, *psk*, for this method may be established using key material from a previous session or an alternative key-establishment method like QKD.

8.3 Cascade hybrid key establishment scheme

8.3.1 Cascade hybrid key establishment scheme - ephemeral

This clause specifies the cascade hybrid key establishment scheme using ephemeral keys. The key establishment description of Figure 3 is extended to exchange multiple public keys in different messages and multiple response values in different messages. The cascade hybrid key establishment scheme is constructed as depicted in Figure 6.

A (Initiator)		B (Responder)
$(sk_1, P_1) = KeyGen_1()$ $MA_1 = (P_1,...)$	MA_1 →	(k_1, R_1) or $\perp = ResponseFunc_1(P_1)$ $MB_1 = (R_1, \ldots)$ or error message
k_1 or $\perp = ReceiveFunc_1(sk_1, R_1)$	← MB_1	
$(sk_2, P_2) = KeyGen_2()$ $MA_2 = (P_2, ...)$	MA_2 →	(k_2, R_2) or $\perp = ResponseFunc_2(P_2)$ $MB_2 = (R_2, \ldots)$ or error message
k_2 or $\perp = ReceiveFunc_2(sk_2, R_2)$	← MB_2	
key_material = CasKDF(…)		*key_material = CasKDF(…)*

Figure 6: Cascade hybrid key establishment - ephemeral

If any $ResponseFunc_i$ returns an error indicator, *B* shall return an error message and terminate the process.

If *A* receives an error message from *B*, *A* shall terminate the process. If any $ReceiveFunc_i$ returns an error indicator, *A* shall terminate the process.

MA_i - shall be an octet string containing an encoding of exchanged public key P_i from Initiator to Responder. MA_i may include session negotiation information, such as label contribution values.

MB_i - shall be an octet string containing and encoding of the response value R_i. MB_i may include session negotiation information, such as label contribution values.

Messages shall be protected from unauthorized modification. Messages may be protected using digital signatures from a signing key associated with the sender's identity signed by a trusted third-party certificate authority.

8.3.2 Cascade hybrid key establishment scheme - static

This clause specifies the cascade hybrid key establishment scheme using static keys. A hybrid key establishment scheme using static keys is constructed as depicted in Figure 7.

A (Initiator)		B (Responder)
$(sk_1, P_1), (sk_2, P_2), \ldots$	$MA_1 = (P_1, \ldots), MA_2 = (P_2, \ldots)$ →	
		(k_1, R_1) or $\perp \leftarrow ResponseFunc_1(P_1)$ $MB_1 = (R_1, \ldots)$ or error message (k_2, R_2) or $\perp \leftarrow ResponseFunc_2(P_2)$ $MB_2 = (R_2, \ldots)$ or error message
k_1 or $\perp = ReceiveFunc_1(sk_1, R_1)$ k_2 or $\perp = ReceiveFunc_2(sk_2, R_2)$	← MB_1, MB_2	
$key\_material = CasKDF(\ldots)$		$key\_material = CasKDF(\ldots)$

Figure 7: Cascade hybrid key establishment - static

If any $ResponseFunc_i$ returns an error indicator, B shall terminate the process.

If any $ReceiveFunc_i$ returns an error indicator, *A* shall terminate the process.

MA_i - Prior to or during the key-establishment, the Sender (B) receives Recipient's (A's) static public keys and additional values such as labels in a trusted manner.

MB_i - if MB_i is not an error message, it shall be an octet string containing an encoding of the response values R_i. MB_i may include session negotiation information such as label contribution values.

Messages shall be protected from unauthorized modification. Messages may be protected using digital signatures from a signing key associated with the sender's identity signed by a trusted third-party certificate authority.

8.3.3 Cascade hybrid key combiner - CasKDF

This clause specifies the cascade hybrid key combiner using the cascade KDF CasKDF(), as denoted in Figure 6 and Figure 7.

The CasKDF() mode shall be defined as follows.

Fixed values:

KDF - the key derivation function being used from clause 7.4.

PRF - the PRF function being used from clause 7.3. When KDF is instantiated as HKDF or HMAC with hash, the underlying hash function is applied to the concatenate-and-hash context formatting function.

k_len - an integer value denoting the length of intermediate keys such as *psk*, $chain\_secret_i$, and $round\_secret_i$; *k_len* is *digest_len* when CasKDF is instantiated with HKDF or HMAC and *kmac_outlen* when CasKDF is instantiated as KMAC. *kmac_outlen* is set to 32 or 48 octets for KMAC128 and KMAC256, respectively.

Input:

psk - a pre-shared secret key of length *k_len*. It may be present. If not present this value shall be the empty octet string, ∅.

(k_1, k_2) - a pair of octet strings containing shared secrets k_i, derived through key establishment mechanisms, see Figure 6 and Figure 7.

MA_i, MB_i - octet strings of exchanged messages in establishment of the shared secret k_i, for each $i = 1,2$.

$info_i$ - an optional octet string set by the application as additional information to establish secret i.

$label_i$ - an optional octet string that specifies a separation of the instance of the key establishment. The label value should be constructed from exchanged label contribution values in messages MA_i and MB_i. If a label value is used it shall be of a fixed agreed length. Any labels used in the key derivation function should not be provided as an argument to the same hash function for another purpose in the application.

$length_i$ - a length value for intermediate and final key material for $i = 1, 2$.

Process:

1) $chain\_secret_0 = psk$, a k_len value or the empty string, $\varnothing$.

2) For $i = 1, 2$.

   a) $round\_secret_i = PRF(chain\_secret_{i-1}, k_i, MA_i, MB_i)$.

   b) $chain\_secret_i \,||\, key\_material_i = KDF(round\_secret_i, label_i, info_i, k\_len + length_i)$.

3) Return chain_secret_1, chain_secret_2, and key_material_1, key_material_2.

Output:

$key\_material_1$ and $key\_material_2$ - a pair of intermediate and final keys of length $length_1$ and $length_2$.

$chain\_secret_1$ and $chain\_secret_2$ - a pair of chain secrets of length k_len.

The KDF is run twice, with each time injecting the shared secret from the next key establishment.

The intermediate key material, $key\_material_1$ may be used in the exchange of messages MA_2 *and* MB_2.

Implementations shall fix the order of key establishment schemes in the exchanged messages in MA_i and MB_i according to the order of the key establishment schemes defined in clause 7.7.2.

A pre-shared key, *psk*, for this method may be established using a key material from a previous session or an alternative key-establishment method like QKD.

Annex A (informative): Background

A.1 Quantum computing threats to traditional key exchange protocols

Quantum computing, first conjectured by Richard Feynman in 1982, has progressed to small-scale limited quantum computing. In contrast to the classical computing paradigm, where the basic computational unit is the "bit" - a two-level system which can hold either a value of '0' or '1', quantum computing represents a new paradigm of computation which harnesses the fundamental laws of quantum mechanics to perform computations on basic units called "qubits" - two-level *quantum mechanical* systems.

The laws of quantum mechanics are strikingly different from the familiar laws of Newtonian mechanics. For example, qubits are allowed to be in linear "superpositions" of the 0 and 1 states, in which the qubit can be seen as "holding both the value at '0' and '1' at the same time". Such an analogy is not precisely correct, as coherent superpositions cannot be really interpreted as a quantum system having two states at once; quantum superpositions are an intrinsic feature of quantum mechanics with no direct analog in classical physics. Note also that quantum superpositions are fundamentally different from classical fuzzy states in which a bit can be in the state '0' with probability p and state '1' with probability 1-p. Qubits can also "interfere" with each other, in a way that is similar to how waves interfere. Moreover, two or more qubits can be "entangled" with each other, i.e. correlated stronger than any classical systems can ever be. The quantum state of n qubits is described by 2^ncomplex numbers (amplitudes). Therefore, classically simulating n qubits requires in general an exponential amount of storage (and time) and is not feasible for large n. Quantum computers take advantage of quantum mechanical laws and features of superposition, interference and entanglement to manipulate the state of n qubits "at once", allowing for novel *quantum algorithms* that can, in some cases, provide significant speedups when compared to their classical counterparts.

In 1994, Peter Shor showed how to utilize Simon's Algorithm [i.5] for the hidden subgroup problem, to factor large semiprimes and solve the discrete log problem [i.6]. Shor's algorithm effectively breaks finite field and elliptic curve discrete-log-based, and integer-factorization-based cryptosystems, like ECC, and RSA. However, a large-scale, fault-tolerant quantum computer would be required to use Shor's algorithm to break cryptographically relevant instances of ECC and RSA. There are a number of challenges in building one. While progress is routinely made on these challenges, it is far from certain that a large-scale fault-tolerant quantum computer will be constructed.

Cryptographic engineering is fundamentally based on understanding the probability of a break in a cryptographic primitive and weighing that against the value of the information that primitive is used to protect. For instance, the probability that an adversary can guess a 128-bit key is $1/2^{128}$, and that information might protect the confidentiality of an individual's credit card information during a financial transaction.

Many encrypted communications are negotiated using asymmetric key establishment agreement schemes like ECC and RSA, e.g. TLS, IPSec, SSH. When viewed through the lens of traditional cryptographic engineering the probability of a large-scale quantum computer being available during the confidentiality lifespan requirement of these sessions exceed the probability that many organizations have historically been willing to tolerate.

Similar to the transition from RSA to ECDHE that provided perfect forward secrecy, quantum-safe KEMs promise to deliver *quantum forward secrecy* - the property of a key establishment protocol that gives assurances the session keys will not be compromised even if the adversary has a large-scale fault tolerant quantum computer.

A.2 Rationale for quantum-safe hybrid key establishment

Today, the existing key exchanges are at risk from a future adversary with a quantum computer. Assurances that the best-known attacks on the proposed quantum-safe KEMs are not sufficient to warrant sole reliance on for confidential communication channels. A critical factor in evaluating the security of a cryptographic primitive is not solely the complexity of the best-known attack and the probability of its success, it is also the assurance that no other attack will be discovered. The cycle of analysis, publish, disseminate required to build assurance for new cryptographic schemes cannot be done in parallel and subsequently takes time for this cycle to repeat.

The proposed post-quantum KEMs have not reached a level of assurance or community agreement that warrants a final selection for standardization and reliance as a single technique to provide confidentiality. Quantum-safe hybrid key establishment schemes pair a high-assurance but quantum-vulnerable key exchange with a potentially quantum-safe key encapsulation mechanism to obtain the best possible key exchange. In addition to providing potential quantum-safety this approach allows for the design of new cryptographic protocols and applications that are tuned to the new bandwidth and computational requirements of quantum-safe key establishment.

Annex B (informative): Security consideration

B.1 Security definitions

A key exchange scheme is secure according to Indistinguishability under Chosen-Plaintext Attack (IND-CPA) if an adversary that is given all of the public information from an exchange (e.g. MA, MB) is unable to distinguish the resulting key from a key selected independently at random. A key exchange scheme is One Way under Chosen-Plaintext Attack (OW-CPA) if an adversary that is given all of the public information from an exchange (e.g. MA, MB) is unable to produce the resulting key. These definitions apply to computationally bounded adversaries, and the adversary is allowed to succeed with some small probability.

The CatKDF and CasKDF schemes are IND-CPA secure in the random oracle model as long as at least one underlying scheme is OW-CPA. However, for use cases that require a static key a stronger security property, Indistinguishability under Chose-Cipher Attacks (IND-CCA), is required.

A key exchange scheme is IND-CCA if the scheme is IND-CPA and that the adversary is given access to a decryption oracle on any ciphertexts except one associated with the challenge ciphertext and is unable to distinguish the resulting key from a key selected independently at random. Similarly, a scheme is One Way under Chosen Ciphertext Attack (OW-CCA) if the adversary has access to the decryption oracle on any ciphertext except one associated with the challenge ciphertext and is unable to produce the resulting key.

The CatKDF schemes are IND-CCA secure in the random oracle model as long as at least one of the underlying schemes is One Way under Chosen-Ciphertext Attacks (OW-CCA).

There are additional security properties that may be desirable for a key exchange scheme, for instance, key compromise impersonation security or unknown key share security. The present document makes no security assertions beyond the IND-CPA and IND-CCA security in the random oracle model. In this regards, a prudent interpretation of the hybrid key exchanges defined in the present document is to consider them as hybrid KEMs.

For detailed security definitions, and proofs, see [i.11] and [i.12]. Alternate constructions suitable for TLS with security proofs can be found in [i.3].

Annex C (informative): Message Encoding for Test Vector Generation

C.1 Message Formatting Function for Test Vector Generation

This clause defines a message formatting function. It is the formatting used to generate the test vectors in Annex D. The message formatting function takes a ciphersuite identifier (cid), and an ordered sequence of octet strings, and converts them into a delimited octet string. The message construction function has the following calling interface:

$$message = m\_format(cid, val_1, val_2, \ldots)$$

where the parameters and output are defined as follows.

Input:

cid - ciphersuite identifier of length 16 bits consisting of 4-bit substrings as specified in clause 6.2.3; the most significant 4 bits $cid_{0,\ldots,3}$ denote the underlying KDF from clause 7.4, $cid_{4,\ldots,7}$ denote the underlying classical ECDH key establishment instance from clause 7.5.2 , and $cid_{8,\ldots,11}$ denote the underlying post-quantum ML-KEM instance from clause 7.6.2. $cid_{12,\ldots,15}$ denotes concatenate CatKDF from clause 8.2 or cascade CasKDF from clause 8.3 operation mode.

A 16-bit *cid* is subdivided into 4-bit substrings CID_0, CID_1, CID_2, CID_3 so if the value is:

$$CID = 0x0123 = (0000000100100011)_{bin}$$

then:

$CID_0 = (0000)_{bin}$, denotes the KDF instance as from clause 7.4
$CID_1 = (0001)_{bin}$, denotes elliptic curve ECDH from clause 7.5.2
$CID_2 = (0010)_{bin}$, denotes post quantum KEM from clause 7.6.2
$CID_3 = (0011)_{bin}$, denotes concatenate CatKDF from clause 8.2 or cascade CasKDF from clause 8.3

Our test vectors use the following CID values:

enum CID_0 = {HKDFwSHA256 = 1, HKDFwSHA384 = 2, HMACwSHA256 = 4, HMACwSHA384 = 5, KMAC128 = 7, KMAC256 = 8};

enum CID_1 = {P256 = 1, P384 = 2, PBP256 = 4, PBP384 = 5, X25519 = 7, X448 = 8};

enum CID_2 = {ML-KEM-512 = 1, ML-KEM-768 = 2, ML-KEM-1024 = 3};

enum CID_3 = {CatKDF = 1, CasKDF = 2};

$val_1, val_2, \ldots$ - an ordered sequence of octet strings each of length less than 2^{32} octets, where $n > 0$.

Process:

1) Set $buffer = cid$.

2) For $i = 1, \ldots, n$.

   a) Set $len_i = len(val_i)$, returns the length of val_i in octets.

      i) If $len_i > 2^{32} - 1$, return error.

      ii) $L_i = [len_i]_{32}$ - a 32-bit integer value expressed as 4 octets.

   b) Set $buffer = buffer \| L_i \| val_i$.

3) Set $message = buffer$.

4) Return $message$.

Output:

message - an octet string representing the message value or an error.

The message formatting function is designed to be used in conjunction with concatenate and cascade hybrid key establishment scheme combiners.

In CatKDF, the values $labelA$, P_1, P_2 are mapped to val_1, val_2, and val_3 in a single *m_foramt* invocation as follows:

$$MA = m\_format(cid, labelA, P_1, P_2).$$

Similarly, MB is formed as:

$$MB = m\_format(cid, labelB, R_1, R_2).$$

In CasKDF, the values $labelA_i$ and P_i are mapped to val_1 and val_2 in separate *m_foramt* invocations, for each $i = 1, 2$, as follows:

$$MA_i = m\_format(cid, labelA_i, P_i).$$

Similarly, MB_i is formed as:

$$MB_i = m\_format(cid, labelB_i, R_i).$$

NOTE 1: In the static hybrid key establishment scheme, prior to or during the key-establishment, the Sender (B) receives Recipient's (A's) static public keys in a trusted manner.

NOTE 2: Any application of this formatting function should use a fixed number and fixed order of arguments to avoid potential parsing issues.

NOTE 3: The test vector application uses the label contribution values $label_A$ and $label_B$ as *k_len* random nonces, and constructs a label for the key derivation function by bytewise XORing these values together $label = label_A \wedge label_B$.

Annex D (informative): Test Vectors

D.1 Introduction

Portions of the test vectors are taken from NIST CAVP SP 800-56A ECC CDH Primitive Test Vectors [i.7], IETF RFC 8734 Test Vectors [i.8], IETF RFC 7748 Test Vectors [8], and ML-KEM Test Vectors [i.10].

These test vectors are generated by the C reference implementation for Quantum-safe Hybrid Key Exchanges that can be found at: https://forge.etsi.org/rep/cyber/103744_QHKEX.

This code is provided as an informative implementation of the Quantum-safe Hybrid Key Exchanges for the Concatenate KDF (CatKDF) and Cascade KDF (CasKDF).

The code is not intended for production use. It is intended to be a reference implementation for test vectors for the specification. The implementation has a dependency on OpenSSL version 3.2.4-dev libcrypto, LibOQS version 0.11.0-release, and OQS-Provider 0.7.0-release.

To build the code at the command line execute:

```
gcc -Wall -o etsi-hkex-test main.c -crypto.c qshkex.c -lcrypto -loqs
```

To run the code at the command line execute:

```
./etsi-hkex-test
```

D.2 Test vectors for CatKDF

D.2.1 KDF as HKDF with SHA256, ECDH with NIST P-256, and ML-KEM768

```
CID 1121
LA = 101020304050607080900A0B0C0D0E0F101020304050607080900A0B0C0D0E0F14
PA1 =
EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA7002523ACC85A429CB0
6EE6648325389F59EDFCE1405141
PA2 =
533145987CB33CD2C8F5AC8A73C03A97FB31D2B89D02B1576CFC77BC97656FC040FCC7569750459EE753EBD69839692B375A
0F53165B191CA8C6185B85626282C0A4AD63B36B6A289F661D505850832248D6F39084695C028B979835848931874E037EDF
33CD9F1521E2B478BF0C37D859572AB08C20F7687749032949AB7457B66C1206C08A78C58633BA008E8770224700B84758B2
2B290E9ED01F750439B00B97729960940817D3BA0CE7F10EB6A0C8EBD975C85A6299A6CE486132292189F9DCC13F27591870
4192A417E7B305834323C354213195BAD0A5AA30A3980B028A1E116E97E74ED8D053A2FA51E00068BCD69D325246E93A30AD
493D28099AF143016BE454413256A5DA144C9985C70664EE12013DF167E228B921EA3BAC494B5D5140581A4D7E7080B4C284
A6DAA955366EAE3217541A0B1109313F7A1AE9305D55B45CB6A5B2F3321A10DB137E92306A3B2A25FC1999495F7A732D7CD8
3BE0923050A5562A036699F92427CB71C0CA201D958F7A3171740AB477188DB276A52E576703493E289C14BF896D5A827586
492D23B605CDC6932A4ABF902A219F88B12D0479C91B4EE6C586A4A2061FC22D66264772359B569456111574D84606CBA282
92C37A40B81F5A113BB9E72871E14184270FE2A87C33BA7D01E95A578950D098C48A584428BAC781BA938F13BB80F4790EC3
6F1A394FEC97AB80054F60A90DA86ABE7D84799A604233A5A495294D13805B6CF3505DB833E9F223D102AAA9E666580B088E
1CBABC4CB52725CB768B5E990B89411AA652E06A2BDB937B185EB924C59AF6BA9BE2BA09409DB5C262D751731461A89AAC05
ADACC320B6AFC5C4A5FDBB2F56B2483D11779C7117707B121D0B36362B2BF8D4B28F6948F5EB1D5E41B1C8A5053D581019C3
6AC87988803CB934027909DC4C5BD1AE2EF451D9F7945209C0E9E60AF43C50DC98646F3BB13BE6491E91930D9AA1AA315FB3
B1A067579565D170C4E888E9042751106B72A60470A69992508E8CE3101A602A22DA7FB1A744F09488E704481F649DAB6158
E5306F573009EC63ACF11B549D280033343FC9750BB927B931B94F9C5475A2B803AF36CC3CC30521EA2A86040E03E7CF90B3
6D2E370856285AE4523367482E68473CA500200F75C8C423C8AB93C7BFE74DF253AF97114A3FEAC1DDD34E06747C6B386990
428100A99351BA14A3C19B1A7A6DA5A715F6A698E9FC330C13B98D61A2A85AC0E0896CC8032490B93FEB284703F21E6D1630
7D7886F948752FE55CEE7291C06AA7BBB79146C644AEA7C2765C5EFA535B729158EB51BC2D4C389B970B30086966D858D51B
78EF3829F9290DAF241C6D370425D83DDD61153DD2AC0FD206120434AD6792D60B2F084A2950AAC250B4001F309EA45B09D9
DB440D6A2C9BB3B140569C51C985A2B0BE03ABA918C2311B11069FC9A0C762CCA59C1BDAE47C0406AF0CC7BBB1821C0F8733
563ABBE896525E1156F7D593F55C39922B6B79E462CE587189247FA045CB19FA7256B960D0B25B7B364ECE250D43E19313C7
98CE424634DC358A0914FAF33EA2B06EAA748947C6C150E54D7BD3A4C4873B117266145833FF04CAA96424AB678B49BA94D6
619F01BBC62961850CF25765C7627910EF8E79F9B3749086AABF75407D7AC42748BC
LB = 020203040506070809 0A0B0C0D0E0F14202030405060708090A0B0C0D0E0F14B
```

```
PB1 =
700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DDB20BA5C51DCC5948D
46FBF640DFE0441782CAB85FA4AC
PB2 =
CC1160B24F3AB56D7E3EEF460A2AD73D0E5B50BC5FD6E06D74C80DF6295A5E7FED8664C9B819ABB90BF1B2481CA3958EDE01
9E5A8D215E3C3B2FC662E6A6E5F4D6CD36C30C747745FCD8F85560645EC010987B87A7B42E619F388FDBB9DF0B916A01AD06
40A9B761A9353DE373033EC5E2D16B59DB508BA33259B9791691FE35B2F08AFF381BFC9A74E380DFAC915B6165ED8C276DC7
112FD36B7C21710AC6A2201179642BACA216C3D4C39EA74E75A370B9ADE85A2AD8B00EADCC4AE2B3819374FDCE01F4AF0253
5194B88933ECE5A9B90C1886EFBCF40B1505D4D2603F7FED815F3B101664A90B239FE598E90C09069E102C500912C71F5339
6EA1C9035682C9ECB1627193FDEE803078E4CAB436B6CACE8A861AD95B5F53E6C9757EEDD7FC9EA369089E82D2D32C952094
60BD880C7D856DD8BD589E131C991E1148B5E4139E7F4EE32B25DA676FCB279ED94D9B28B6C4843F07A93FADC1314869D2E2
71D90D993E4F5B654157BDBF45215D691AFAECC498E9ADB3A798029C2D68665878634E02B34006BA953DB444203C88C4523B
EDCA14358E0B129BC712EAB69D1C253C490DD4E5F751FBA1B38BA200A3E73E78299306855237C6F72B15598B2EAB30224A22
C2C7883C6BEDE636C4D1C06ED0B6793E4609969CF8A442614C8C547598C26CC848BC4593E24591FC85F624548A1D01EA5BE0
4892C5C12EC07F70C8CA51C4CCD1C9F061EE1BFE07112C2577552966EB25759E77882F1B5B327964E6269E36C1D5AC9C94B0
1D891E1B3AC2660C5D99E32364AB31015986352ABB20A228C358A548D1B413FF534550F2EE0C80E0741B7F90DC22E8D701A4
4514810820A5A1D5EE62173283F91DAB585615E36A520FD7540FF7E7B9D6F2FB616251F1D2B6A034AF6427E66D559D7E04F1
C7C22FEFA5F3F7B256D7225D5F683FE29238754D66E10687290675202CD15A32A7F71B5750A95BFCC183D2C26BF31D41225C
04FBC123683FA911B994C5B9AE5172EE9847356C67302E7C405625543B7FA82D9BE737467F7C000F97FFBB647500DE9EE36A
266F97AE83AB58D239533C8A2FB440E649ACDF6EFC8D98BAF10A1BFA0F7681EC9708BAB9D00A78FFF74AA4C7099D6DA69FBE
E2B7C5EA7CEB04527D01CA2CCA8416181BA450C2519A8F383881FF439DEDCE45BF0117CAC0A612E52AD1A3B2F2D1C9917264
DD131813F3D9F6D2D574D6CE9AA0943AA7E84C9BDEC515B6827921366794FE6B9058769ADCB3A14593CA49A866DC7AF5BCAF
2A48BB06271B3948CC64572AE3A9D46A1209D3E70A02D50AC3CB0AA1B820F8EA25DA127931406AAFC9B00C0FD90C75973215
86694E35EA291A88665A2FFBAE5C9BE45938E60027C6C6DEB384F873457B5B5C7107651AE0273DF8A23FDFF74A2A6BD67703
854FA5E115712E84D6726F27370FDE6390B362142FCD93647BD2D7A3927DE5F5040F60E17C39D1248F6662CF441EE4B2470C
D412050CE09EC31075001286F401A58FE6217000D6A9C8BA753C4CE9D1D8CF8B908614DF0A81
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
k2 = D71BD5A07C158C130283EF854516D290A46ADE09A63831C7B83B8FD0724C8FB0
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label = LA ^ LB = 1212233445566778899AABBCCDDEEFE5212233445566778899AABBCCDDEEFE5F
MA = cid || len(LA)|| LA || len(PA1) || PA1 || len(PA2) || PA2
MB = cid || len(LB)|| LB || len(PB1) || PB1 || len(PB2) || PB2

key material = 99B5DC7F166C3158043BC626DD0C4498
```

D.2.2 KDF as HMAC with SHA256, ECDH with NIST P-256, and ML-KEM768

```
CID 4121
LA = 1010203040506070809 0A0B0C0D0E0F1010203040506070809 0A0B0C0D0E0F14
PA1 =
EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA7002523ACC85A429CB0
6EE6648325389F59EDFCE1405141
PA2 =
533145987CB33CD2C8F5AC8A73C03A97FB31D2B89D02B1576CFC77BC97656FC040FCC7569750459EE753EBD69839692B375A
0F53165B191CA8C6185B85626282C0A4AD63B36B6A289F661D505850832248D6F39084695C028B979835848931874E037EDF
33CD9F1521E2B478BF0C37D859572AB08C20F7687749032949AB7457B66C1206C08A78C58633BA008E8770224700B84758B2
2B290E9ED01F750439B00B97729960940817D3BA0CE7F10EB6A0C8EBD975C85A6299A6CE486132292189F9DCC13F27591870
4192A417E7B305834323C354213195BAD0A5AA30A3980B028A1E116E97E74ED8D053A2FA51E00068BCD69D325246E93A30AD
493D28099AF143016BE454413256A5DA144C9985C70664EE12013DF167E228B921EA3BAC494B5D5140581A4D7E7080B4C284
A6DAA955366EAE3217541A0B1109313F7A1AE9305D55B45CB6A5B2F3321A10DB137E92306A3B2A25FC1999495F7A732D7CD8
3BE0923050A5562A036699F92427CB71C0CA201D958F7A3171740AB477188DB276A52E576703493E289C14BF896D5A827586
492D23B605CDC6932A4ABF902A219F88B12D0479C91B4EE6C586A4A2061FC22D66264772359B569456111574D84606CBA282
92C37A40B81F5A113BB9E72871E14184270FE2A87C33BA7D01E95A578950D098C48A584428BAC781BA938F13BB80F4790EC3
6F1A394FEC97AB80054F60A90DA86ABE7D84799A604233A5A495294D13805B6CF3505DB833E9F223D102AAA9E666580B088E
1CBABC4CB52725CB768B5E990B89411AA652E06A2BDB937B185EB924C59AF6BA9BE2BA09409DB5C262D751731461A89AAC05
ADACC320B6AFC5C4A5FDBB2F56B2483D11779C7117707B121D0B36362B2BF8D4B28F6948F5EB1D5E41B1C8A5053D581019C3
6AC87988803CB934027909DC4C5BD1AE2EF451D9F7945209C0E9E60AF43C50DC98646F3BB13BE6491E91930D9AA1AA315FB3
B1A067579565D170C4E888E9042751106B72A60470A69992508E8CE3101A602A22DA7FB1A744F09488E704481F649DAB6158
E5306F573009EC63ACF11B549D280033343FC9750BB927B931B94F9C5475A2B803AF36CC3CC30521EA2A86040E03E7CF90B3
6D2E370856285AE4523367482E68473CA500200F75C8C423C8AB93C7BFE74DF253AF97114A3FEAC1DDD34E06747C6B386990
428100A99351BA14A3C19B1A7A6DA5A715F6A698E9FC330C13B98D61A2A85AC0E0896CC8032490B93FEB284703F21E6D1630
7D7886F948752FE55CEE7291C06AA7BBB79146C644AEA7C2765C5EFA535B729158EB51BC2D4C389B970B30086966D858D51B
78EF3829F9290DAF241C6D370425D83DDD61153DD2AC0FD206120434AD6792D60B2F084A2950AAC250B4001F309EA45B09D9
DB440D6A2C9BB3B140569C51C985A2B0BE03ABA918C2311B11069FC9A0C762CCA59C1BDAE47C0406AF0CC7BBB1821C0F8733
563ABBE896525E1156F7D593F55C39922B6B79E462CE587189247FA045CB19FA7256B960D0B25B7B364ECE250D43E19313C7
98CE424634DC358A0914FAF33EA2B06EAA748947C6C150E54D7BD3A4C4873B117266145833FF04CAA96424AB678B49BA94D6
619F01BBC62961850CF25765C7627910EF8E79F9B3749086AABF75407D7AC42748BC
LB = 0202030405060708090A0B0C0D0E0F1420203040506070809 0A0B0C0D0E0F14B
PB1 =
700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DDB20BA5C51DCC5948D
46FBF640DFE0441782CAB85FA4AC
PB2 =
CC1160B24F3AB56D7E3EEF460A2AD73D0E5B50BC5FD6E06D74C80DF6295A5E7FED8664C9B819ABB90BF1B2481CA3958EDE01
9E5A8D215E3C3B2FC662E6A6E5F4D6CD36C30C747745FCD8F85560645EC010987B87A7B42E619F388FDBB9DF0B916A01AD06
```

```
40A9B761A9353DE373033EC5E2D16B59DB508BA33259B9791691FE35B2F08AFF381BFC9A74E380DFAC915B6165ED8C276DC7
112FD36B7C21710AC6A2201179642BACA216C3D4C39EA74E75A370B9ADE85A2AD8B00EADCC4AE2B3819374FDCE01F4AF0253
5194B88933ECE5A9B90C1886EFBCF40B1505D4D2603F7FED815F3B101664A90B239FE598E90C09069E102C500912C71F5339
6EA1C9035682C9ECB1627193FDEE803078E4CAB436B6CACE8A861AD95B5F53E6C9757EEDD7FC9EA369089E82D2D32C952094
60BD880C7D856DD8BD589E131C991E1148B5E4139E7F4EE32B25DA676FCB279ED94D9B28B6C4843F07A93FADC1314869D2E2
71D90D993E4F5B654157BDBF45215D691AFAECC498E9ADB3A798029C2D68665878634E02B34006BA953DB444203C88C4523B
EDCA14358E0B129BC712EAB69D1C253C490DD4E5F751FBA1B38BA200A3E73E78299306855237C6F72B15598B2EAB30224A22
C2C7883C6BEDE636C4D1C06ED0B6793E4609969CF8A442614C8C547598C26CC848BC4593E24591FC85F624548A1D01EA5BE0
4892C5C12EC07F70C8CA51C4CCD1C9F061EE1BFE07112C2577552966EB25759E77882F1B5B327964E6269E36C1D5AC9C94B0
1D891E1B3AC2660C5D99E32364AB31015986352ABB20A228C358A548D1B413FF534550F2EE0C80E0741B7F90DC22E8D701A4
4514810820A5A1D5EE62173283F91DAB585615E36A520FD7540FF7E7B9D6F2FB616251F1D2B6A034AF6427E66D559D7E04F1
C7C22FEFA5F3F7B256D7225D5F683FE29238754D66E10687290675202CD15A32A7F71B5750A95BFCC183D2C26BF31D41225C
04FBC123683FA911B994C5B9AE5172EE9847356C67302E7C405625543B7FA82D9BE737467F7C000F97FFBB647500DE9EE36A
266F97AE83AB58D239533C8A2FB440E649ACDF6EFC8D98BAF10A1BFA0F7681EC9708BAB9D00A78FFF74AA4C7099D6DA69FBE
E2B7C5EA7CEB04527D01CA2CCA8416181BA450C2519A8F383881FF439DEDCE45BF0117CAC0A612E52AD1A3B2F2D1C9917264
DD131813F3D9F6D2D574D6CE9AA0943AA7E84C9BDEC515B6827921366794FE6B9058769ADCB3A14593CA49A866DC7AF5BCAF
2A48BB06271B3948CC64572AE3A9D46A1209D3E70A02D50AC3CB0AA1B820F8EA25DA127931406AAFC9B00C0FD90C75973215
86694E35EA291A88665A2FFBAE5C9BE45938E60027C6C6DEB384F873457B5B5C7107651AE0273DF8A23FDFF74A2A6BD67703
854FA5E115712E84D6726F27370FDE6390B362142FCD93647BD2D7A3927DE5F5040F60E17C39D1248F6662CF441EE4B2470C
D412050CE09EC31075001286F401A58FE6217000D6A9C8BA753C4CE9D1D8CF8B908614DF0A81
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
k2 = D71BD5A07C158C130283EF854516D290A46ADE09A63831C7B83B8FD0724C8FB0
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label = LA ^ LB = 121223344556677889 9AABBCCDDEEFE5212233445566778899AABBCCDDEEFE5F
MA = cid || len(LA)|| LA || len(PA1) || PA1 || len(PA2) || PA2
MB = cid || len(LB)|| LB || len(PB1) || PB1 || len(PB2) || PB2

key material = 3F0EC466248B91B18FA82A557C12E0E4
```

D.2.3 KDF as KAMC128, ECDH with NIST P-256, and ML-KEM768

```
CID 7121
LA = 101020304050607080 90A0B0C0D0E0F101020304050607080 90A0B0C0D0E0F14
PA1 =
EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA7002523ACC85A429CB0
6EE6648325389F59EDFCE1405141
PA2 =
533145987CB33CD2C8F5AC8A73C03A97FB31D2B89D02B1576CFC77BC97656FC040FCC7569750459EE753EBD69839692B375A
0F53165B191CA8C6185B85626282C0A4AD63B36B6A289F661D505850832248D6F39084695C028B979835848931874E037EDF
33CD9F1521E2B478BF0C37D859572AB08C20F7687749032949AB7457B66C1206C08A78C58633BA008E8770224700B84758B2
2B290E9ED01F750439B00B97729960940817D3BA0CE7F10EB6A0C8EBD975C85A6299A6CE486132292189F9DCC13F27591870
4192A417E7B305834323C354213195BAD0A5AA30A3980B028A1E116E97E74ED8D053A2FA51E00068BCD69D325246E93A30AD
493D28099AF143016BE454413256A5DA144C9985C70664EE12013DF167E228B921EA3BAC494B5D5140581A4D7E7080B4C284
A6DAA955366EAE3217541A0B1109313F7A1AE9305D55B45CB6A5B2F3321A10DB137E92306A3B2A25FC1999495F7A732D7CD8
3BE0923050A5562A036699F92427CB71C0CA201D958F7A3171740AB477188DB276A52E576703493E289C14BF896D5A827586
492D23B605CDC6932A4ABF902A219F88B12D0479C91B4EE6C586A4A2061FC22D66264772359B569456111574D84606CBA282
92C37A40B81F5A113BB9E72871E14184270FE2A87C33BA7D01E95A578950D098C48A584428BAC781BA938F13BB80F4790EC3
6F1A394FEC97AB80054F60A90DA86ABE7D84799A604233A5A495294D13805B6CF3505DB833E9F223D102AAA9E666580B088E
1CBABC4CB52725CB768B5E990B89411AA652E06A2BDB937B185EB924C59AF6BA9BE2BA09409DB5C262D751731461A89AAC05
ADACC320B6AFC5C4A5FDBB2F56B2483D11779C7117707B121D0B36362B2BF8D4B28F6948F5EB1D5E41B1C8A5053D581019C3
6AC87988803CB934027909DC4C5BD1AE2EF451D9F7945209C0E9E60AF43C50DC98646F3BB13BE6491E91930D9AA1AA315FB3
B1A067579565D170C4E888E9042751106B72A60470A69992508E8CE3101A602A22DA7FB1A744F09488E704481F649DAB6158
E5306F573009EC63ACF11B549D280033343FC9750BB927B931B94F9C5475A2B803AF36CC3CC30521EA2A86040E03E7CF90B3
6D2E370856285AE4523367482E68473CA500200F75C8C423C8AB93C7BFE74DF253AF97114A3FEAC1DDD34E06747C6B386990
428100A99351BA14A3C19B1A7A6DA5A715F6A698E9FC330C13B98D61A2A85AC0E0896CC8032490B93FEB284703F21E6D1630
7D7886F948752FE55CEE7291C06AA7BBB79146C644AEA7C2765C5EFA535B729158EB51BC2D4C389B970B30086966D858D51B
78EF3829F9290DAF241C6D370425D83DDD61153DD2AC0FD206120434AD6792D60B2F084A2950AAC250B4001F309EA45B09D9
DB440D6A2C9BB3B140569C51C985A2B0BE03ABA918C2311B11069FC9A0C762CCA59C1BDAE47C0406AF0CC7BBB1821C0F8733
563ABBE896525E1156F7D593F55C39922B6B79E462CE587189247FA045CB19FA7256B960D0B25B7B364ECE250D43E19313C7
98CE424634DC358A0914FAF33EA2B06EAA748947C6C150E54D7BD3A4C4873B117266145833FF04CAA96424AB678B49BA94D6
619F01BBC62961850CF25765C7627910EF8E79F9B3749086AABF75407D7AC42748BC
LB = 020203040506070809 0A0B0C0D0E0F1420203040506070809 0A0B0C0D0E0F14B
PB1 =
700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DDB20BA5C51DCC5948D
46FBF640DFE0441782CAB85FA4AC
PB2 =
CC1160B24F3AB56D7E3EEF460A2AD73D0E5B50BC5FD6E06D74C80DF6295A5E7FED8664C9B819ABB90BF1B2481CA3958EDE01
9E5A8D215E3C3B2FC662E6A6E5F4D6CD36C30C747745FCD8F85560645EC010987B87A7B42E619F388FDBB9DF0B916A01AD06
40A9B761A9353DE373033EC5E2D16B59DB508BA33259B9791691FE35B2F08AFF381BFC9A74E380DFAC915B6165ED8C276DC7
112FD36B7C21710AC6A2201179642BACA216C3D4C39EA74E75A370B9ADE85A2AD8B00EADCC4AE2B3819374FDCE01F4AF0253
5194B88933ECE5A9B90C1886EFBCF40B1505D4D2603F7FED815F3B101664A90B239FE598E90C09069E102C500912C71F5339
6EA1C9035682C9ECB1627193FDEE803078E4CAB436B6CACE8A861AD95B5F53E6C9757EEDD7FC9EA369089E82D2D32C952094
60BD880C7D856DD8BD589E131C991E1148B5E4139E7F4EE32B25DA676FCB279ED94D9B28B6C4843F07A93FADC1314869D2E2
71D90D993E4F5B654157BDBF45215D691AFAECC498E9ADB3A798029C2D68665878634E02B34006BA953DB444203C88C4523B
```

```
EDCA14358E0B129BC712EAB69D1C253C490DD4E5F751FBA1B38BA200A3E73E78299306855237C6F72B15598B2EAB30224A22
C2C7883C6BEDE636C4D1C06ED0B6793E4609969CF8A442614C8C547598C26CC848BC4593E24591FC85F624548A1D01EA5BE0
4892C5C12EC07F70C8CA51C4CCD1C9F061EE1BFE07112C2577552966EB25759E77882F1B5B327964E6269E36C1D5AC9C94B0
1D891E1B3AC2660C5D99E32364AB31015986352ABB20A228C358A548D1B413FF534550F2EE0C80E0741B7F90DC22E8D701A4
4514810820A5A1D5EE62173283F91DAB585615E36A520FD7540FF7E7B9D6F2FB616251F1D2B6A034AF6427E66D559D7E04F1
C7C22FEFA5F3F7B256D7225D5F683FE29238754D66E10687290675202CD15A32A7F71B5750A95BFCC183D2C26BF31D41225C
04FBC123683FA911B994C5B9AE5172EE9847356C67302E7C405625543B7FA82D9BE737467F7C000F97FFBB647500DE9EE36A
266F97AE83AB58D239533C8A2FB440E649ACDF6EFC8D98BAF10A1BFA0F7681EC9708BAB9D00A78FFF74AA4C7099D6DA69FBE
E2B7C5EA7CEB04527D01CA2CCA8416181BA450C2519A8F383881FF439DEDCE45BF0117CAC0A612E52AD1A3B2F2D1C9917264
DD131813F3D9F6D2D574D6CE9AA0943AA7E84C9BDEC515B6827921366794FE6B9058769ADCB3A14593CA49A866DC7AF5BCAF
2A48BB06271B3948CC64572AE3A9D46A1209D3E70A02D50AC3CB0AA1B820F8EA25DA127931406AAFC9B00C0FD90C75973215
86694E35EA291A88665A2FFBAE5C9BE45938E60027C6C6DEB384F873457B5B5C7107651AE0273DF8A23FDFF74A2A6BD67703
854FA5E115712E84D6726F27370FDE6390B362142FCD93647BD2D7A3927DE5F5040F60E17C39D1248F6662CF441EE4B2470C
D412050CE09EC31075001286F401A58FE6217000D6A9C8BA753C4CE9D1D8CF8B908614DF0A81
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
k2 = D71BD5A07C158C130283EF854516D290A46ADE09A63831C7B83B8FD0724C8FB0
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label = LA ^ LB
MA = cid || len(LA)|| LA || len(PA1) || PA1 || len(PA2) || PA2
MB = cid || len(LB)|| LB || len(PB1) || PB1 || len(PB1) || PB2

key material = 1154D484AAB6231EE566F303C68B1EE1
```

D.2.4 KDF as HKDF with SHA256, ECDH with Curve25519, and ML-KEM768

```
CID 1721
LA = 0102030405060708090A0B0C0D0E0F1101020304050607081020304050607085
PA1 = 8520F0098930A754748B7DDCB43EF75A0DBF3A0D26381AF4EBA4A98EAA9B4E6A
PA2 =
06F51174F03C2FE07CC31B0659B87EBF124237811B88B980BC277F8C050B265CBEE1A4AE7145CB9253281BBC291F2808E065
855FD91C7D14122C589FB21B45FD3B7FBB6562FD53006B565189EC983C7A0D4C21596EF3795AE22ECAD71F75395558B55220
662BB66C389ED9AB6B784476E1788F34B28B59A4C6F71F9F6A49DA80C4D3107DB2C306968B5600E18285F8BFA293677F699F
D2D2820DE490F530818D6667686A765FCC0009484D71629C1F148104B462325AC55EF34DD5C7098441A06757CA35C8ADFDAB
6D36333310241F9970B1B123478DCBB75AE6CA39BB84FFCB23A593C24BC54E74B4086165BB4B10AE39F45447AA4EA5F55762
386B11051B84B0A688C12C73336FC87164B7FBC7F0B07D2A51BB05560606D5558666CC322963AD3CA2BC1293260A94707770
C030B2BC9C00477A8D5FF6737961480D48109F66075DA9C8AB516F70A78DA9E444AB65798A2565965A68AD9146FD9267D2A8
0C4352A0A988157522046620A63D9BA7BBCCAFD50A61384453F8545845B25E7A05745C4B26EF7211200A916241047FDABB66
936BB7351C04F5CD06A10AE3C459B74BB0CD85044BF8B6BF2B33294014B9D68517CB1A9813230BA08813807A2DFCC6A739B6
AC79575F19BAB96A7CDED404D0E52508214AD43763A0A048FC36785B2C718F75B3432769C1D2A6240336A82B39D865739AF7
8E94A92BD7B4703210297FB82E0BE57B97384DB5B24FFA5486C982AF48642D967222A28C6F83E5AD6A59CA486733808BA3BD
86A780E59888011965A351CA949B0D6C72FB3BA0F867834D166819C252D9600207C88997318B2D204AC3619247A01829484F
12BCA2E8F16AAC677C9E8AC209F9BCCF019BBDA66E8F073DFBD934E08CA15BE77576029654E049C1C20B1B123E5F0B501356
28CD2C96A5D0B1EEF48FBE08AC4685247721B128F71DC748452B94A458F39B337BC2305CCC505BB18BF6ABE3EC2C825799B9
5C13A1F6411CD3B031F3CEC3D99AFC1A28F2019C1BD129A2EA35F06415C8B717337957CAA964E794881257073FF5C12C3C62
034861C343B7F70B9D475CCC10EA5889910E04CA7EAF0B762DB56962EC40CA47CEB4D228600595CF22421148CF02B4CFCE76
AD27701F925C01C71C5405534ED54B0100C280DD861D89E967BED4829C568BB76CA500342803835F7983179CDA15899033F7
97951031138A83669AB4CCB4630058088AA0845FAF7B0761B405AE866C426711EB152889D175E7B20DA03126B7419A766416
A68AAC0FBC355DA07D6E6339BC0539CEB55711F633B8C7392627497821CB9D7174C4E655B5928BF722C84B8233A0993B6DE0
2D7364443EA4C73A754B8190243DF936184819E1B8AB6F0B0BFBAC6C47E66DDF11757BE55E0787ADC112C63E644D1B904877
764D856ABE78B583600CBB2119CB916C689D4543F4D553CAC33AD19B245B4948EF0BA3E7D1151179B9FB2941F3674E5F8771
677737C435B2383119A3E75434114439506E1B9145FF16B1458879A1039962434151619B77915D033A3FAB7385FE861E4A25
B125A205C3F1B16972470B2791534B5FB8731B20C03684CCBEB97A1FA4417655BB485C347519710522C357D50C43FC264409
8533FE549C825F5DC8CA82A0DD908891BB4F92EAAB51CBE036CD6FB59A749269A0A3
LB = 202030405060708090A0B0C0D0E0F100202030405060708E020304050607080C
PB1 = DE9EDB7D7B7DC1B4D35B61C2ECE435373F8343C85B78674DADFC7E146F882B4F
PB2 =
DB3D4F38A49CC5C1C7766573873F3926184923F570FCDBA6D7086BEA39601FADFEBABB2F9ED0EBD86A6468BE03D3B05713DE
50B5E5D58FFDB5500F2AF7419682BC7857870D1E624C2B48A7E0D30A91F141F0764014FBED12A9F72C958EB53A7ED4A16825
78B4348A05C183C790FF56D66B23BD601FB372C771F52045E13CA2D6DAB80A165A2159B8F990B5BBF125883DEC0A2AF4A54C
3F386DE4DF6EE96923E9BB2551753DDC6D1F00D7EE910835082C6260974D11FBA49951DB2E27B47362EC4EBF804327AD9682
01EDB71BDC8C32887214FB2AAA842AABBE0A4E03FDC695B19610377B9BEA9D371016674E977E9E45382D236365EFEDF56665
9F9936D9D14CA8595DDF121AE7ACC6056DD6FE1B58DC0F0B23CF5366D2488C4D6361B09875242B3A39374159CF3654ABD25F
8B080BAE117A5F77D15031AD8B350C6922FFD426EC9F9F80A1918FE722CB7A4B537210EF8A5538A52048BADCE3CAF5C46454
EEAF413819D57EFBA834D7017F9343B302679A0A28F532E93FE8C5B6AEC3C42C160A3FAD20FA2ABADD29C2564DB59FA25A11
CA502D11421F7F080A616F9656942932E11BFDCC47B1D8EFFA81D724FFABF4613830631737EB7E52DA78A6B026DB1C4FB80B
BF2D8F9393875330B2F9C1A124436028BF2624C1E64797ABE2E526967DD99BC21B555A5833A31ACE7969E3ECC64C1697DEC0
B074108F6B17EFA2A517923B435D8446043FB1A96B2EF3B3D3952F08657997D1D9169FFD14316185EC30457E0FAF7EBA2B5A
02A01993EC530BB9C44C091D2E9D9BC82951E577D76DC2757F2E04E822B6A29F840887A297202E7389A0999F9E69255DE0A3
3AEB8CD0B21B7F3B5AA7BFD864CF8FF4FFE78FFF166CAF1B805464D5E4B60EB0277E3BA4F3F3390640F5F16E0B784CC8A679
40977B2F0202D37A74472161589A7B9E7721CC532904A17AFF8FA86ECDC46A410BD6F9309B2CC310D7941A02A5720CD64241
B5C13C78A86010805AC74C3B81D83348C63C410451BD2052CE764DA83CF80B2654CB1D033F259DA36039E4DF1A5C4CE3F861
3E28CFC2B25EAE2A21B425B44D0C998977B3ADF5156BFCA1E357EC3F6B5805B2CDC4D46A4C83154AEF0F50D493BAB0BBCA13
0E92CA21AF8BDF817F7616B55C4C24C066EBF896F39A7BF50A241561E578BDD931CB07EADD193986C455F8798AB97B8D732D
37C796E09C1C7ABB978BF666CED961373198607AC355FCF6903B4F3047A1D97C344A2D8B73DB310769CCFB4AE3DA6C4E82DE
```

```
B83BF4060722D9E5A9AFBD3E03B4517E3B7D563F31821D5D8EF7F06F0D92A9F401EA9BEFD52CA8DB331360E603C14A340B80
6155DE690CD2D92EE6A8EE53BCEED7EEB308D375518C851C109A03EE61960D6CD7CD1FDEE5CC2A401039A15EE4F97A734008
01A090FBF09BE42119662D06C3FF8217FCFC2C70CDE6E1C2F9A396A2E08A3C22AE3CE90A5C97C9F69EBC1546898A047C9DCA
D128003A90B29B62F7B093CC90C9CD38104C1A3CE62BEA6A3646EF7E3B9FB7C4C44E78FF650C
k1 = 4A5D9D5BA4CE2DE1728E3BF480350F25E07E21C947D19E3376F09B3C1E161742
k2 = E3E4F4E9D4F5C9AE03836BB9266C50B033285ACE9BC56F73817CE19679D1429A
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label = LA ^ LB = 2122334455667788 99AABBCCDDEEFE1121223344556677861223344556677889
MA = cid || len(LA)|| LA || len(PA1) || PA1 || len(PA2) || PA2
MB = cid || len(LB)|| LB || len(PB1) || PB1 || len(PB2) || PB2

key material = 701675524E4986A391AFEFBB604AA63D
```

D.2.5 KDF as HMAC with SHA256, ECDH with Curve25519, and ML-KEM768

```
CID 4721
LA = 0102030405060708090A0B0C0D0E0F11010203040506070810203040506070 85
PA1 = 8520F0098930A754748B7DDCB43EF75A0DBF3A0D26381AF4EBA4A98EAA9B4E6A
PA2 =
06F51174F03C2FE07CC31B0659B87EBF124237811B88B980BC277F8C050B265CBEE1A4AE7145CB9253281BBC291F2808E065
855FD91C7D14122C589FB21B45FD3B7FBB6562FD53006B565189EC983C7A0D4C21596EF3795AE22ECAD71F75395558B55220
662BB66C389ED9AB6B784476E1788F34B28B59A4C6F71F9F6A49DA80C4D3107DB2C306968B5600E18285F8BFA293677F699F
D2D2820DE490F530818D6667686A765FCC0009484D71629C1F148104B462325AC55EF34DD5C7098441A06757CA35C8ADFDAB
6D36333310241F9970B1B123478DCBB75AE6CA39BB84FFCB23A593C24BC54E74B4086165BB4B10AE39F45447AA4EA5F55762
386B11051B84B0A688C12C73336FC87164B7FBC7F0B07D2A51BB05560606D5558666CC322963AD3CA2BC1293260A94707770
C030B2BC9C00477A8D5FF6737961480D48109F66075DA9C8AB516F70A78DA9E444AB65798A2565965A68AD9146FD9267D2A8
0C4352A0A988157522046620A63D9BA7BBCCAFD50A61384453F8545845B25E7A05745C4B26EF7211200A916241047FDABB66
936BB7351C04F5CD06A10AE3C459B74BB0CD85044BF8B6BF2B33294014B9D68517CB1A9813230BA08813807A2DFCC6A739B6
AC79575F19BAB96A7CDED404D0E52508214AD43763A0A048FC36785B2C718F75B3432769C1D2A6240336A82B39D865739AF7
8E94A92BD7B4703210297FB82E0BE57B97384DB5B24FFA5486C982AF48642D967222A28C6F83E5AD6A59CA486733808BA3BD
86A780E59888011965A351CA949B0D6C72FB3BA0F867834D166819C252D9600207C88997318B2D204AC3619247A01829484F
12BCA2E8F16AAC677C9E8AC209F9BCCF019BBDA66E8F073DFBD934E08CA15BE77576029654E049C1C20B1B123E5F0B501356
28CD2C96A5D0B1EEF48FBE08AC4685247721B128F71DC748452B94A458F39B337BC2305CCC505BB18BF6ABE3EC2C825799B9
5C13A1F6411CD3B031F3CEC3D99AFC1A28F2019C1BD129A2EA35F06415C8B717337957CAA964E794881257073FF5C12C3C62
034861C343B7F70B9D475CCC10EA5889910E04CA7EAF0B762DB56962EC40CA47CEB4D228600595CF22421148CF02B4CFCE76
AD27701F925C01C71C5405534ED54B0100C280DD861D89E967BED4829C568BB76CA500342803835F7983179CDA15899033F7
97951031138A83669AB4CCB4630058088AA0845FAF7B0761B405AE866C426711EB152889D175E7B20DA03126B7419A766416
A68AAC0FBC355DA07D6E6339BC0539CEB55711F633B8C7392627497821CB9D7174C4E655B5928BF722C84B8233A0993B6DE0
2D7364443EA4C73A754B8190243DF936184819E1B8AB6F0B0BFBAC6C47E66DDF11757BE55E0787ADC112C63E644D1B904877
764D856ABE78B583600CBB2119CB916C689D4543F4D553CAC33AD19B245B4948EF0BA3E7D1151179B9FB2941F3674E5F8771
677737C435B2383119A3E75434114439506E1B9145FF16B1458879A1039962434151619B77915D033A3FAB7385FE861E4A25
B125A205C3F1B16972470B2791534B5FB8731B20C03684CCBEB97A1FA4417655BB485C347519710522C357D50C43FC264409
8533FE549C825F5DC8CA82A0DD908891BB4F92EAAB51CBE036CD6FB59A749269A0A3
LB = 202030405060708090A0B0C0D0E0F100202030405060708E020304050607080C
PB1 = DE9EDB7D7B7DC1B4D35B61C2ECE435373F8343C85B78674DADFC7E146F882B4F
PB2 =
DB3D4F38A49CC5C1C7766573873F3926184923F570FCDBA6D7086BEA39601FADFEBABB2F9ED0EBD86A6468BE03D3B05713DE
50B5E5D58FFDB5500F2AF7419682BC7857870D1E624C2B48A7E0D30A91F141F0764014FBED12A9F72C958EB53A7ED4A16825
78B4348A05C183C790FF56D66B23BD601FB372C771F52045E13CA2D6DAB80A165A2159B8F990B5BBF125883DEC0A2AF4A54C
3F386DE4DF6EE96923E9BB2551753DDC6D1F00D7EE910835082C6260974D11FBA49951DB2E27B47362EC4EBF804327AD9682
01EDB71BDC8C32887214FB2AAA842AABBE0A4E03FDC695B19610377B9BEA9D371016674E977E9E45382D236365EFEDF56665
9F9936D9D14CA8595DDF121AE7ACC6056DD6FE1B58DC0F0B23CF5366D2488C4D6361B09875242B3A39374159CF3654ABD25F
8B080BAE117A5F77D15031AD8B350C6922FFD426EC9F9F80A1918FE722CB7A4B537210EF8A5538A52048BADCE3CAF5C46454
EEAF413819D57EFBA834D7017F9343B302679A0A28F532E93FE8C5B6AEC3C42C160A3FAD20FA2ABADD29C2564DB59FA25A11
CA502D11421F7F080A616F9656942932E11BFDCC47B1D8EFFA81D724FFABF4613830631737EB7E52DA78A6B026DB1C4FB80B
BF2D8F9393875330B2F9C1A124436028BF2624C1E64797ABE2E526967DD99BC21B555A5833A31ACE7969E3ECC64C1697DEC0
B074108F6B17EFA2A517923B435D8446043FB1A96B2EF3B3D3952F08657997D1D9169FFD14316185EC30457E0FAF7EBA2B5A
02A01993EC530BB9C44C091D2E9D9BC82951E577D76DC2757F2E04E822B6A29F840887A297202E7389A0999F9E69255DE0A3
3AEB8CD0B21B7F3B5AA7BFD864CF8FF4FFE78FFF166CAF1B805464D5E4B60EB0277E3BA4F3F3390640F5F16E0B784CC8A679
40977B2F0202D37A74472161589A7B9E7721CC532904A17AFF8FA86ECDC46A410BD6F9309B2CC310D7941A02A5720CD64241
B5C13C78A86010805AC74C3B81D83348C63C410451BD2052CE764DA83CF80B2654CB1D033F259DA36039E4DF1A5C4CE3F861
3E28CFC2B25EAE2A21B425B44D0C998977B3ADF5156BFCA1E357EC3F6B5805B2CDC4D46A4C83154AEF0F50D493BAB0BBCA13
0E92CA21AF8BDF817F7616B55C4C24C066EBF896F39A7BF50A241561E578BDD931CB07EADD193986C455F8798AB97B8D732D
37C796E09C1C7ABB978BF666CED961373198607AC355FCF6903B4F3047A1D97C344A2D8B73DB310769CCFB4AE3DA6C4E82DE
B83BF4060722D9E5A9AFBD3E03B4517E3B7D563F31821D5D8EF7F06F0D92A9F401EA9BEFD52CA8DB331360E603C14A340B80
6155DE690CD2D92EE6A8EE53BCEED7EEB308D375518C851C109A03EE61960D6CD7CD1FDEE5CC2A401039A15EE4F97A734008
01A090FBF09BE42119662D06C3FF8217FCFC2C70CDE6E1C2F9A396A2E08A3C22AE3CE90A5C97C9F69EBC1546898A047C9DCA
D128003A90B29B62F7B093CC90C9CD38104C1A3CE62BEA6A3646EF7E3B9FB7C4C44E78FF650C
k1 = 4A5D9D5BA4CE2DE1728E3BF480350F25E07E21C947D19E3376F09B3C1E161742
k2 = E3E4F4E9D4F5C9AE03836BB9266C50B033285ACE9BC56F73817CE19679D1429A
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label = LA ^ LB = 212233445566778899AABBCCDDEEFE1121223344556677861223344556677889
MA = cid || len(LA)|| LA || len(PA1) || PA1 || len(PA2) || PA2
MB = cid || len(LB)|| LB || len(PB1) || PB1 || len(PB2) || PB2
```

```
key material = 56617E0EC39843C6E41935F02A5D7FFD
```

D.2.6 KDF as KAMC128, ECDH with Curve25519, and ML-KEM768

```
CID 7721
LA = 0102030405060708090A0B0C0D0E0F1101020304050607081020304050607085
PA1 = 8520F0098930A754748B7DDCB43EF75A0DBF3A0D26381AF4EBA4A98EAA9B4E6A
PA2 =
06F51174F03C2FE07CC31B0659B87EBF124237811B88B980BC277F8C050B265CBEE1A4AE7145CB9253281BBC291F2808E065
855FD91C7D14122C589FB21B45FD3B7FBB6562FD53006B565189EC983C7A0D4C21596EF3795AE22ECAD71F75395558B55220
662BB66C389ED9AB6B784476E1788F34B28B59A4C6F71F9F6A49DA80C4D3107DB2C306968B5600E18285F8BFA293677F699F
D2D2820DE490F530818D6667686A765FCC0009484D71629C1F148104B462325AC55EF34DD5C7098441A06757CA35C8ADFDAB
6D36333310241F9970B1B123478DCBB75AE6CA39BB84FFCB23A593C24BC54E74B4086165BB4B10AE39F45447AA4EA5F55762
386B11051B84B0A688C12C73336FC87164B7FBC7F0B07D2A51BB05560606D5558666CC322963AD3CA2BC1293260A94707770
C030B2BC9C00477A8D5FF6737961480D48109F66075DA9C8AB516F70A78DA9E444AB65798A2565965A68AD9146FD9267D2A8
0C4352A0A988157522046620A63D9BA7BBCCAFD50A61384453F8545845B25E7A05745C4B26EF7211200A916241047FDABB66
936BB7351C04F5CD06A10AE3C459B74BB0CD85044BF8B6BF2B33294014B9D68517CB1A9813230BA08813807A2DFCC6A739B6
AC79575F19BAB96A7CDED404D0E52508214AD43763A0A048FC36785B2C718F75B3432769C1D2A6240336A82B39D865739AF7
8E94A92BD7B4703210297FB82E0BE57B97384DB5B24FFA5486C982AF48642D967222A28C6F83E5AD6A59CA486733808BA3BD
86A780E59888011965A351CA949B0D6C72FB3BA0F867834D166819C252D9600207C88997318B2D204AC3619247A01829484F
12BCA2E8F16AAC677C9E8AC209F9BCCF019BBDA66E8F073DFBD934E08CA15BE77576029654E049C1C20B1B123E5F0B501356
28CD2C96A5D0B1EEF48FBE08AC4685247721B128F71DC748452B94A458F39B337BC2305CCC505BB18BF6ABE3EC2C825799B9
5C13A1F6411CD3B031F3CEC3D99AFC1A28F2019C1BD129A2EA35F06415C8B717337957CAA964E794881257073FF5C12C3C62
034861C343B7F70B9D475CCC10EA5889910E04CA7EAF0B762DB56962EC40CA47CEB4D228600595CF22421148CF02B4CFCE76
AD27701F925C01C71C5405534ED54B0100C280DD861D89E967BED4829C568BB76CA500342803835F7983179CDA15899033F7
97951031138A83669AB4CCB4630058088AA0845FAF7B0761B405AE866C426711EB152889D175E7B20DA03126B7419A766416
A68AAC0FBC355DA07D6E6339BC0539CEB55711F633B8C7392627497821CB9D7174C4E655B5928BF722C84B8233A0993B6DE0
2D7364443EA4C73A754B8190243DF936184819E1B8AB6F0B0BFBAC6C47E66DDF11757BE55E0787ADC112C63E644D1B904877
764D856ABE78B583600CBB2119CB916C689D4543F4D553CAC33AD19B245B4948EF0BA3E7D1151179B9FB2941F3674E5F8771
677737C435B2383119A3E75434114439506E1B9145FF16B1458879A1039962434151619B77915D033A3FAB7385FE861E4A25
B125A205C3F1B16972470B2791534B5FB8731B20C03684CCBEB97A1FA4417655BB485C347519710522C357D50C43FC264409
8533FE549C825F5DC8CA82A0DD908891BB4F92EAAB51CBE036CD6FB59A749269A0A3
LB = 202030405060708090A0B0C0D0E0F100202030405060708E020304050607080C
PB1 = DE9EDB7D7B7DC1B4D35B61C2ECE435373F8343C85B78674DADFC7E146F882B4F
PB2 =
DB3D4F38A49CC5C1C7766573873F3926184923F570FCDBA6D7086BEA39601FADFEBABB2F9ED0EBD86A6468BE03D3B05713DE
50B5E5D58FFDB5500F2AF7419682BC7857870D1E624C2B48A7E0D30A91F141F0764014FBED12A9F72C958EB53A7ED4A16825
78B4348A05C183C790FF56D66B23BD601FB372C771F52045E13CA2D6DAB80A165A2159B8F990B5BBF125883DEC0A2AF4A54C
3F386DE4DF6EE96923E9BB2551753DDC6D1F00D7EE910835082C6260974D11FBA49951DB2E27B47362EC4EBF804327AD9682
01EDB71BDC8C32887214FB2AAA842AABBE0A4E03FDC695B19610377B9BEA9D371016674E977E9E45382D236365EFEDF56665
9F9936D9D14CA8595DDF121AE7ACC6056DD6FE1B58DC0F0B23CF5366D2488C4D6361B09875242B3A39374159CF3654ABD25F
8B080BAE117A5F77D15031AD8B350C6922FFD426EC9F9F80A1918FE722CB7A4B537210EF8A5538A52048BADCE3CAF5C46454
EEAF413819D57EFBA834D7017F9343B302679A0A28F532E93FE8C5B6AEC3C42C160A3FAD20FA2ABADD29C2564DB59FA25A11
CA502D11421F7F080A616F9656942932E11BFDCC47B1D8EFFA81D724FFABF4613830631737EB7E52DA78A6B026DB1C4FB80B
BF2D8F9393875330B2F9C1A124436028BF2624C1E64797ABE2E526967DD99BC21B555A5833A31ACE7969E3ECC64C1697DEC0
B074108F6B17EFA2A517923B435D8446043FB1A96B2EF3B3D3952F08657997D1D9169FFD14316185EC30457E0FAF7EBA2B5A
02A01993EC530BB9C44C091D2E9D9BC82951E577D76DC2757F2E04E822B6A29F840887A297202E7389A0999F9E69255DE0A3
3AEB8CD0B21B7F3B5AA7BFD864CF8FF4FFE78FFF166CAF1B805464D5E4B60EB0277E3BA4F3F3390640F5F16E0B784CC8A679
40977B2F0202D37A74472161589A7B9E7721CC532904A17AFF8FA86ECDC46A410BD6F9309B2CC310D7941A02A5720CD64241
B5C13C78A86010805AC74C3B81D83348C63C410451BD2052CE764DA83CF80B2654CB1D033F259DA36039E4DF1A5C4CE3F861
3E28CFC2B25EAE2A21B425B44D0C998977B3ADF5156BFCA1E357EC3F6B5805B2CDC4D46A4C83154AEF0F50D493BAB0BBCA13
0E92CA21AF8BDF817F7616B55C4C24C066EBF896F39A7BF50A241561E578BDD931CB07EADD193986C455F8798AB97B8D732D
37C796E09C1C7ABB978BF666CED961373198607AC355FCF6903B4F3047A1D97C344A2D8B73DB310769CCFB4AE3DA6C4E82DE
B83BF4060722D9E5A9AFBD3E03B4517E3B7D563F31821D5D8EF7F06F0D92A9F401EA9BEFD52CA8DB331360E603C14A340B80
6155DE690CD2D92EE6A8EE53BCEED7EEB308D375518C851C109A03EE61960D6CD7CD1FDEE5CC2A401039A15EE4F97A734008
01A090FBF09BE42119662D06C3FF8217FCFC2C70CDE6E1C2F9A396A2E08A3C22AE3CE90A5C97C9F69EBC1546898A047C9DCA
D128003A90B29B62F7B093CC90C9CD38104C1A3CE62BEA6A3646EF7E3B9FB7C4C44E78FF650C
k1 = 4A5D9D5BA4CE2DE1728E3BF480350F25E07E21C947D19E3376F09B3C1E161742
k2 = E3E4F4E9D4F5C9AE03836BB9266C50B033285ACE9BC56F73817CE19679D1429A
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label = LA ^ LB
MA = cid || len(LA)|| LA || len(PA1) || PA1 || len(PA2) || PA2
MB = cid || len(LB)|| LB || len(PB1) || PB1 || len(PB1) || PB2

key material = 1B697069B3382C389E30ED03D59D2BAA
```

D.3 Test vectors for CasKDF

D.3.1 KDF as HKDF with SHA256, ECDH with NIST P-256, and ML-KEM768

```
CID 1122
LA1 = 101020304050607080 90A0B0C0D0E0F101020304050607080 90A0B0C0D0E0F14
PA1 =
EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA7002523ACC85A429CB0
6EE6648325389F59EDFCE1405141
LB1 = 0202030405060708090A0B0C0D0E0F142020304050607080 90A0B0C0D0E0F14B
PB1 =
700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DDB20BA5C51DCC5948D
46FBF640DFE0441782CAB85FA4AC
previous_chain_secret = psk = <NULL>
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label1 = LA1 ^ LB1
MA1 = cid || len(LA1) || LA1 || len(PA1) || PA1
MB1 = cid || len(LB1) || LB1 || len(PB1) || PB1
chain_secret1 = DA1B0A9081A422C6510559DA924B25AFFDAC9DC19C100FDDA1C2D825695C125D
key_material1 = 49A376AEA4059C01DB384944659F82E2

LA2 = 3102030405060708090A0B0C0D0E0F1101020304050607080 90A0B0C0D0E0F30
PA2 =
533145987CB33CD2C8F5AC8A73C03A97FB31D2B89D02B1576CFC77BC97656FC040FCC7569750459EE753EBD69839692B375A
0F53165B191CA8C6185B85626282C0A4AD63B36B6A289F661D505850832248D6F39084695C028B979835848931874E037EDF
33CD9F1521E2B478BF0C37D859572AB08C20F7687749032949AB7457B66C1206C08A78C58633BA008E8770224700B84758B2
2B290E9ED01F750439B00B97729960940817D3BA0CE7F10EB6A0C8EBD975C85A6299A6CE486132292189F9DCC13F27591870
4192A417E7B305834323C354213195BAD0A5AA30A3980B028A1E116E97E74ED8D053A2FA51E00068BCD69D325246E93A30AD
493D28099AF143016BE454413256A5DA144C9985C70664EE12013DF167E228B921EA3BAC494B5D5140581A4D7E7080B4C284
A6DAA955366EAE3217541A0B1109313F7A1AE9305D55B45CB6A5B2F3321A10DB137E92306A3B2A25FC1999495F7A732D7CD8
3BE0923050A5562A036699F92427CB71C0CA201D958F7A3171740AB477188DB276A52E576703493E289C14BF896D5A827586
492D23B605CDC6932A4ABF902A219F88B12D0479C91B4EE6C586A4A2061FC22D66264772359B569456111574D84606CBA282
92C37A40B81F5A113BB9E72871E14184270FE2A87C33BA7D01E95A578950D098C48A584428BAC781BA938F13BB80F4790EC3
6F1A394FEC97AB80054F60A90DA86ABE7D84799A604233A5A495294D13805B6CF3505DB833E9F223D102AAA9E666580B088E
1CBABC4CB52725CB768B5E990B89411AA652E06A2BDB937B185EB924C59AF6BA9BE2BA09409DB5C262D751731461A89AAC05
ADACC320B6AFC5C4A5FDBB2F56B2483D11779C7117707B121D0B36362B2BF8D4B28F6948F5EB1D5E41B1C8A5053D581019C3
6AC87988803CB934027909DC4C5BD1AE2EF451D9F7945209C0E9E60AF43C50DC98646F3BB13BE6491E91930D9AA1AA315FB3
B1A067579565D170C4E888E9042751106B72A60470A69992508E8CE3101A602A22DA7FB1A744F09488E704481F649DAB6158
E5306F573009EC63ACF11B549D280033343FC9750BB927B931B94F9C5475A2B803AF36CC3CC30521EA2A86040E03E7CF90B3
6D2E370856285AE4523367482E68473CA500200F75C8C423C8AB93C7BFE74DF253AF97114A3FEAC1DDD34E06747C6B386990
428100A99351BA14A3C19B1A7A6DA5A715F6A698E9FC330C13B98D61A2A85AC0E0896CC8032490B93FEB284703F21E6D1630
7D7886F948752FE55CEE7291C06AA7BBB79146C644AEA7C2765C5EFA535B729158EB51BC2D4C389B970B30086966D858D51B
78EF3829F9290DAF241C6D370425D83DDD61153DD2AC0FD206120434AD6792D60B2F084A2950AAC250B4001F309EA45B09D9
DB440D6A2C9BB3B140569C51C985A2B0BE03ABA918C2311B11069FC9A0C762CCA59C1BDAE47C0406AF0CC7BBB1821C0F8733
563ABBE896525E1156F7D593F55C39922B6B79E462CE587189247FA045CB19FA7256B960D0B25B7B364ECE250D43E19313C7
98CE424634DC358A0914FAF33EA2B06EAA748947C6C150E54D7BD3A4C4873B117266145833FF04CAA96424AB678B49BA94D6
619F01BBC62961850CF25765C7627910EF8E79F9B3749086AABF75407D7AC42748BC
LB2 = 4202030405060708090A0B0C0D0E0F1F202030405060708090A0B0C0D0E0F1F7
PB2 =
CC1160B24F3AB56D7E3EEF460A2AD73D0E5B50BC5FD6E06D74C80DF6295A5E7FED8664C9B819ABB90BF1B2481CA3958EDE01
9E5A8D215E3C3B2FC662E6A6E5F4D6CD36C30C747745FCD8F85560645EC010987B87A7B42E619F388FDBB9DF0B916A01AD06
40A9B761A9353DE373033EC5E2D16B59DB508BA33259B9791691FE35B2F08AFF381BFC9A74E380DFAC915B6165ED8C276DC7
112FD36B7C21710AC6A2201179642BACA216C3D4C39EA74E75A370B9ADE85A2AD8B00EADCC4AE2B3819374FDCE01F4AF0253
5194B88933ECE5A9B90C1886EFBCF40B1505D4D2603F7FED815F3B101664A90B239FE598E90C09069E102C500912C71F5339
6EA1C9035682C9ECB1627193FDEE803078E4CAB436B6CACE8A861AD95B5F53E6C9757EEDD7FC9EA369089E82D2D32C952094
60BD880C7D856DD8BD589E131C991E1148B5E4139E7F4EE32B25DA676FCB279ED94D9B28B6C4843F07A93FADC1314869D2E2
71D90D993E4F5B654157BDBF45215D691AFAECC498E9ADB3A798029C2D68665878634E02B34006BA953DB444203C88C4523B
EDCA14358E0B129BC712EAB69D1C253C490DD4E5F751FBA1B38BA200A3E73E78299306855237C6F72B15598B2EAB30224A22
C2C7883C6BEDE636C4D1C06ED0B6793E4609969CF8A442614C8C547598C26CC848BC4593E24591FC85F624548A1D01EA5BE0
4892C5C12EC07F70C8CA51C4CCD1C9F061EE1BFE07112C2577552966EB25759E77882F1B5B327964E6269E36C1D5AC9C94B0
1D891E1B3AC2660C5D99E32364AB31015986352ABB20A228C358A548D1B413FF534550F2EE0C80E0741B7F90DC22E8D701A4
4514810820A5A1D5EE62173283F91DAB585615E36A520FD7540FF7E7B9D6F2FB616251F1D2B6A034AF6427E66D559D7E04F1
C7C22FEFA5F3F7B256D7225D5F683FE29238754D66E10687290675202CD15A32A7F71B5750A95BFCC183D2C26BF31D41225C
04FBC123683FA911B994C5B9AE5172EE9847356C67302E7C405625543B7FA82D9BE737467F7C000F97FFBB647500DE9EE36A
266F97AE83AB58D239533C8A2FB440E649ACDF6EFC8D98BAF10A1BFA0F7681EC9708BAB9D00A78FFF74AA4C7099D6DA69FBE
E2B7C5EA7CEB04527D01CA2CCA8416181BA450C2519A8F383881FF439DEDCE45BF0117CAC0A612E52AD1A3B2F2D1C9917264
DD131813F3D9F6D2D574D6CE9AA0943AA7E84C9BDEC515B6827921366794FE6B9058769ADCB3A14593CA49A866DC7AF5BCAF
2A48BB06271B3948CC64572AE3A9D46A1209D3E70A02D50AC3CB0AA1B820F8EA25DA127931406AAFC9B00C0FD90C75973215
86694E35EA291A88665A2FFBAE5C9BE45938E60027C6C6DEB384F873457B5B5C7107651AE0273DF8A23FDFF74A2A6BD67703
854FA5E115712E84D6726F27370FDE6390B362142FCD93647BD2D7A3927DE5F5040F60E17C39D1248F6662CF441EE4B2470C
D412050CE09EC31075001286F401A58FE6217000D6A9C8BA753C4CE9D1D8CF8B908614DF0A81
previous_chain_secret = DA1B0A9081A422C6510559DA924B25AFFDAC9DC19C100FDDA1C2D825695C125D
```

```
k2 = D71BD5A07C158C130283EF854516D290A46ADE09A63831C7B83B8FD0724C8FB0
context2 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label2 = LA2 ^ LB2
MA2 = cid || len(LA2) || LA2 || len(PA2) || PA2
MB2 = cid || len(LB2) || LB2 || len(PB2) || PB2

chain_secret2 = 5E0A1BCC706338B2C8176D0476D28F551FCBB3EF68A917190BDE3E4E6B0447B5
key_material2 = B4D24420223C495C2E12A50FD93C05B9
```

D.3.2 KDF as HMAC with SHA256, ECDH with NIST P-256, and ML-KEM768

```
CID 4122
LA1 = 101020304050607080 90A0B0C0D0E0F101020304050607080 90A0B0C0D0E0F14
```

```
PA1 =
EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA7002523ACC85A429CB0
6EE6648325389F59EDFCE1405141
LB1 = 0202030405060708090A0B0C0D0E0F1420203040506070809 0A0B0C0D0E0F14B
PB1 =
700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DDB20BA5C51DCC5948D
46FBF640DFE0441782CAB85FA4AC
previous_chain_secret = psk = <NULL>
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label1 = LA1 ^ LB1
MA1 = cid || len(LA1) || LA1 || len(PA1) || PA1
MB1 = cid || len(LB1) || LB1 || len(PB1) || PB1
chain_secret1 = C8247E6FC6371394EB8FE74955D4A0C9BFE49B9AAD9F3EAD7F007D6F0E4AC8B9
key_material1 = C4BF20A5F972813B117F9507227D6FD9

LA2 = 3102030405060708090A0B0C0D0E0F1101020304050607080 90A0B0C0D0E0F30
PA2 =
533145987CB33CD2C8F5AC8A73C03A97FB31D2B89D02B1576CFC77BC97656FC040FCC7569750459EE753EBD69839692B375A
0F53165B191CA8C6185B85626282C0A4AD63B36B6A289F661D505850832248D6F39084695C028B979835848931874E037EDF
33CD9F1521E2B478BF0C37D859572AB08C20F7687749032949AB7457B66C1206C08A78C58633BA008E8770224700B84758B2
2B290E9ED01F750439B00B97729960940817D3BA0CE7F10EB6A0C8EBD975C85A6299A6CE486132292189F9DCC13F27591870
4192A417E7B305834323C354213195BAD0A5AA30A3980B028A1E116E97E74ED8D053A2FA51E00068BCD69D325246E93A30AD
493D28099AF143016BE454413256A5DA144C9985C70664EE12013DF167E228B921EA3BAC494B5D5140581A4D7E7080B4C284
A6DAA955366EAE3217541A0B1109313F7A1AE9305D55B45CB6A5B2F3321A10DB137E92306A3B2A25FC1999495F7A732D7CD8
3BE0923050A5562A036699F92427CB71C0CA201D958F7A3171740AB477188DB276A52E576703493E289C14BF896D5A827586
492D23B605CDC6932A4ABF902A219F88B12D0479C91B4EE6C586A4A2061FC22D66264772359B569456111574D84606CBA282
92C37A40B81F5A113BB9E72871E14184270FE2A87C33BA7D01E95A578950D098C48A584428BAC781BA938F13BB80F4790EC3
6F1A394FEC97AB80054F60A90DA86ABE7D84799A604233A5A495294D13805B6CF3505DB833E9F223D102AAA9E666580B088E
1CBABC4CB52725CB768B5E990B89411AA652E06A2BDB937B185EB924C59AF6BA9BE2BA09409DB5C262D751731461A89AAC05
ADACC320B6AFC5C4A5FDBB2F56B2483D11779C7117707B121D0B36362B2BF8D4B28F6948F5EB1D5E41B1C8A5053D581019C3
6AC87988803CB934027909DC4C5BD1AE2EF451D9F7945209C0E9E60AF43C50DC98646F3BB13BE6491E91930D9AA1AA315FB3
B1A067579565D170C4E888E9042751106B72A60470A69992508E8CE3101A602A22DA7FB1A744F09488E704481F649DAB6158
E5306F573009EC63ACF11B549D280033343FC9750BB927B931B94F9C5475A2B803AF36CC3CC30521EA2A86040E03E7CF90B3
6D2E370856285AE4523367482E68473CA500200F75C8C423C8AB93C7BFE74DF253AF97114A3FEAC1DDD34E06747C6B386990
428100A99351BA14A3C19B1A7A6DA5A715F6A698E9FC330C13B98D61A2A85AC0E0896CC8032490B93FEB284703F21E6D1630
7D7886F948752FE55CEE7291C06AA7BBB79146C644AEA7C2765C5EFA535B729158EB51BC2D4C389B970B30086966D858D51B
78EF3829F9290DAF241C6D370425D83DDD61153DD2AC0FD206120434AD6792D60B2F084A2950AAC250B4001F309EA45B09D9
DB440D6A2C9BB3B140569C51C985A2B0BE03ABA918C2311B11069FC9A0C762CCA59C1BDAE47C0406AF0CC7BBB1821C0F8733
563ABBE896525E1156F7D593F55C39922B6B79E462CE587189247FA045CB19FA7256B960D0B25B7B364ECE250D43E19313C7
98CE424634DC358A0914FAF33EA2B06EAA748947C6C150E54D7BD3A4C4873B117266145833FF04CAA96424AB678B49BA94D6
619F01BBC62961850CF25765C7627910EF8E79F9B3749086AABF75407D7AC42748BC
LB2 = 4202030405060708090A0B0C0D0E0F1F2020304050607080 90A0B0C0D0E0F1F7
PB2 =
CC1160B24F3AB56D7E3EEF460A2AD73D0E5B50BC5FD6E06D74C80DF6295A5E7FED8664C9B819ABB90BF1B2481CA3958EDE01
9E5A8D215E3C3B2FC662E6A6E5F4D6CD36C30C747745FCD8F85560645EC010987B87A7B42E619F388FDBB9DF0B916A01AD06
40A9B761A9353DE373033EC5E2D16B59DB508BA33259B9791691FE35B2F08AFF381BFC9A74E380DFAC915B6165ED8C276DC7
112FD36B7C21710AC6A2201179642BACA216C3D4C39EA74E75A370B9ADE85A2AD8B00EADCC4AE2B3819374FDCE01F4AF0253
5194B88933ECE5A9B90C1886EFBCF40B1505D4D2603F7FED815F3B101664A90B239FE598E90C09069E102C500912C71F5339
6EA1C9035682C9ECB1627193FDEE803078E4CAB436B6CACE8A861AD95B5F53E6C9757EEDD7FC9EA369089E82D2D32C952094
60BD880C7D856DD8BD589E131C991E1148B5E4139E7F4EE32B25DA676FCB279ED94D9B28B6C4843F07A93FADC1314869D2E2
71D90D993E4F5B654157BDBF45215D691AFAECC498E9ADB3A798029C2D68665878634E02B34006BA953DB444203C88C4523B
EDCA14358E0B129BC712EAB69D1C253C490DD4E5F751FBA1B38BA200A3E73E78299306855237C6F72B15598B2EAB30224A22
C2C7883C6BEDE636C4D1C06ED0B6793E4609969CF8A442614C8C547598C26CC848BC4593E24591FC85F624548A1D01EA5BE0
4892C5C12EC07F70C8CA51C4CCD1C9F061EE1BFE07112C2577552966EB25759E77882F1B5B327964E6269E36C1D5AC9C94B0
1D891E1B3AC2660C5D99E32364AB31015986352ABB20A228C358A548D1B413FF534550F2EE0C80E0741B7F90DC22E8D701A4
4514810820A5A1D5EE62173283F91DAB585615E36A520FD7540FF7E7B9D6F2FB616251F1D2B6A034AF6427E66D559D7E04F1
C7C22FEFA5F3F7B256D7225D5F683FE29238754D66E10687290675202CD15A32A7F71B5750A95BFCC183D2C26BF31D41225C
04FBC123683FA911B994C5B9AE5172EE9847356C67302E7C405625543B7FA82D9BE737467F7C000F97FFBB647500DE9EE36A
266F97AE83AB58D239533C8A2FB440E649ACDF6EFC8D98BAF10A1BFA0F7681EC9708BAB9D00A78FFF74AA4C7099D6DA69FBE
E2B7C5EA7CEB04527D01CA2CCA8416181BA450C2519A8F383881FF439DEDCE45BF0117CAC0A612E52AD1A3B2F2D1C9917264
```

```
DD131813F3D9F6D2D574D6CE9AA0943AA7E84C9BDEC515B6827921366794FE6B9058769ADCB3A14593CA49A866DC7AF5BCAF
2A48BB06271B3948CC64572AE3A9D46A1209D3E70A02D50AC3CB0AA1B820F8EA25DA127931406AAFC9B00C0FD90C75973215
86694E35EA291A88665A2FFBAE5C9BE45938E60027C6C6DEB384F873457B5B5C7107651AE0273DF8A23FDFF74A2A6BD67703
854FA5E115712E84D6726F27370FDE6390B362142FCD93647BD2D7A3927DE5F5040F60E17C39D1248F6662CF441EE4B2470C
D412050CE09EC31075001286F401A58FE6217000D6A9C8BA753C4CE9D1D8CF8B908614DF0A81
previous_chain_secret = C8247E6FC6371394EB8FE74955D4A0C9BFE49B9AAD9F3EAD7F007D6F0E4AC8B9
k2 = D71BD5A07C158C130283EF854516D290A46ADE09A63831C7B83B8FD0724C8FB0
context2 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label2 = LA2 ^ LB2
MA2 = cid || len(LA2) || LA2 || len(PA2) || PA2
MB2 = cid || len(LB2) || LB2 || len(PB2) || PB2

chain_secret2 = F99F9E8EF11D595ED63C93F7C5DF216688658B4F558E8B91526DCEDD01A74D39
key_material2 = 7E12FBC4071218FA9D7B3DC21A651EA3
```

D.3.3 KDF as KMAC128, ECDH with NIST P-256, and ML-KEM768

```
CID 7122
LA1 = 10102030405060708090A0B0C0D0E0F10102030405060708090A0B0C0D0E0F14
PA1 =
EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA7002523ACC85A429CB0
6EE6648325389F59EDFCE1405141
LB1 = 0202030405060708090A0B0C0D0E0F14202030405060708090A0B0C0D0E0F14B
PB1 =
700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DDB20BA5C51DCC5948D
46FBF640DFE0441782CAB85FA4AC
previous_chain_secret = psk = <NULL>
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
context1 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label1 = LA1 ^ LB1
MA1 = cid || len(LA1) || LA1 || len(PA1) || PA1
MB1 = cid || len(LB1) || LB1 || len(PB1) || PB1

chain_secret1 = DDDDEB4EB4EDB9EC8E7DBA3BB90F581F87518F4C2DB2B7AF3BA49C2D505391D0
key_material1 = 1375E52A33EFCB595531CACEAE3A915D

LA2 = 3102030405060708090A0B0C0D0E0F1101020304050607080900A0B0C0D0E0F30
PA2 =
533145987CB33CD2C8F5AC8A73C03A97FB31D2B89D02B1576CFC77BC97656FC040FCC7569750459EE753EBD69839692B375A
0F53165B191CA8C6185B85626282C0A4AD63B36B6A289F661D505850832248D6F39084695C028B979835848931874E037EDF
33CD9F1521E2B478BF0C37D859572AB08C20F7687749032949AB7457B66C1206C08A78C58633BA008E8770224700B84758B2
2B290E9ED01F750439B00B97729960940817D3BA0CE7F10EB6A0C8EBD975C85A6299A6CE486132292189F9DCC13F27591870
4192A417E7B305834323C354213195BAD0A5AA30A3980B028A1E116E97E74ED8D053A2FA51E00068BCD69D325246E93A30AD
493D28099AF143016BE454413256A5DA144C9985C70664EE12013DF167E228B921EA3BAC494B5D5140581A4D7E7080B4C284
A6DAA955366EAE3217541A0B1109313F7A1AE9305D55B45CB6A5B2F3321A10DB137E92306A3B2A25FC1999495F7A732D7CD8
3BE0923050A5562A036699F92427CB71C0CA201D958F7A3171740AB477188DB276A52E576703493E289C14BF896D5A827586
492D23B605CDC6932A4ABF902A219F88B12D0479C91B4EE6C586A4A2061FC22D66264772359B569456111574D84606CBA282
92C37A40B81F5A113BB9E72871E14184270FE2A87C33BA7D01E95A578950D098C48A584428BAC781BA938F13BB80F4790EC3
6F1A394FEC97AB80054F60A90DA86ABE7D84799A604233A5A495294D13805B6CF3505DB833E9F223D102AAA9E666580B088E
1CBABC4CB52725CB768B5E990B89411AA652E06A2BDB937B185EB924C59AF6BA9BE2BA09409DB5C262D751731461A89AAC05
ADACC320B6AFC5C4A5FDBB2F56B2483D11779C7117707B121D0B36362B2BF8D4B28F6948F5EB1D5E41B1C8A5053D581019C3
6AC87988803CB934027909DC4C5BD1AE2EF451D9F7945209C0E9E60AF43C50DC98646F3BB13BE6491E91930D9AA1AA315FB3
B1A067579565D170C4E888E9042751106B72A60470A69992508E8CE3101A602A22DA7FB1A744F09488E704481F649DAB6158
E5306F573009EC63ACF11B549D280033343FC9750BB927B931B94F9C5475A2B803AF36CC3CC30521EA2A86040E03E7CF90B3
6D2E370856285AE4523367482E68473CA500200F75C8C423C8AB93C7BFE74DF253AF97114A3FEAC1DDD34E06747C6B386990
428100A99351BA14A3C19B1A7A6DA5A715F6A698E9FC330C13B98D61A2A85AC0E0896CC8032490B93FEB284703F21E6D1630
7D7886F948752FE55CEE7291C06AA7BBB79146C644AEA7C2765C5EFA535B729158EB51BC2D4C389B970B30086966D858D51B
78EF3829F9290DAF241C6D370425D83DDD61153DD2AC0FD206120434AD6792D60B2F084A2950AAC250B4001F309EA45B09D9
DB440D6A2C9BB3B140569C51C985A2B0BE03ABA918C2311B11069FC9A0C762CCA59C1BDAE47C0406AF0CC7BBB1821C0F8733
563ABBE896525E1156F7D593F55C39922B6B79E462CE587189247FA045CB19FA7256B960D0B25B7B364ECE250D43E19313C7
98CE424634DC358A0914FAF33EA2B06EAA748947C6C150E54D7BD3A4C4873B117266145833FF04CAA96424AB678B49BA94D6
619F01BBC62961850CF25765C7627910EF8E79F9B3749086AABF75407D7AC42748BC
LB2 = 4202030405060708090A0B0C0D0E0F1F202030405060708090A0B0C0D0E0F1F7
PB2 =
CC1160B24F3AB56D7E3EEF460A2AD73D0E5B50BC5FD6E06D74C80DF6295A5E7FED8664C9B819ABB90BF1B2481CA3958EDE01
9E5A8D215E3C3B2FC662E6A6E5F4D6CD36C30C747745FCD8F85560645EC010987B87A7B42E619F388FDBB9DF0B916A01AD06
40A9B761A9353DE373033EC5E2D16B59DB508BA33259B9791691FE35B2F08AFF381BFC9A74E380DFAC915B6165ED8C276DC7
112FD36B7C21710AC6A2201179642BACA216C3D4C39EA74E75A370B9ADE85A2AD8B00EADCC4AE2B3819374FDCE01F4AF0253
5194B88933ECE5A9B90C1886EFBCF40B1505D4D2603F7FED815F3B101664A90B239FE598E90C09069E102C500912C71F5339
6EA1C9035682C9ECB1627193FDEE803078E4CAB436B6CACE8A861AD95B5F53E6C9757EEDD7FC9EA369089E82D2D32C952094
60BD880C7D856DD8BD589E131C991E1148B5E4139E7F4EE32B25DA676FCB279ED94D9B28B6C4843F07A93FADC1314869D2E2
71D90D993E4F5B654157BDBF45215D691AFAECC498E9ADB3A798029C2D68665878634E02B34006BA953DB444203C88C4523B
EDCA14358E0B129BC712EAB69D1C253C490DD4E5F751FBA1B38BA200A3E73E78299306855237C6F72B15598B2EAB30224A22
C2C7883C6BEDE636C4D1C06ED0B6793E4609969CF8A442614C8C547598C26CC848BC4593E24591FC85F624548A1D01EA5BE0
```

```
4892C5C12EC07F70C8CA51C4CCD1C9F061EE1BFE07112C2577552966EB25759E77882F1B5B327964E6269E36C1D5AC9C94B0
1D891E1B3AC2660C5D99E32364AB31015986352ABB20A228C358A548D1B413FF534550F2EE0C80E0741B7F90DC22E8D701A4
4514810820A5A1D5EE62173283F91DAB585615E36A520FD7540FF7E7B9D6F2FB616251F1D2B6A034AF6427E66D559D7E04F1
C7C22FEFA5F3F7B256D7225D5F683FE29238754D66E10687290675202CD15A32A7F71B5750A95BFCC183D2C26BF31D41225C
04FBC123683FA911B994C5B9AE5172EE9847356C67302E7C405625543B7FA82D9BE737467F7C000F97FFBB647500DE9EE36A
266F97AE83AB58D239533C8A2FB440E649ACDF6EFC8D98BAF10A1BFA0F7681EC9708BAB9D00A78FFF74AA4C7099D6DA69FBE
E2B7C5EA7CEB04527D01CA2CCA8416181BA450C2519A8F383881FF439DEDCE45BF0117CAC0A612E52AD1A3B2F2D1C9917264
DD131813F3D9F6D2D574D6CE9AA0943AA7E84C9BDEC515B6827921366794FE6B9058769ADCB3A14593CA49A866DC7AF5BCAF
2A48BB06271B3948CC64572AE3A9D46A1209D3E70A02D50AC3CB0AA1B820F8EA25DA127931406AAFC9B00C0FD90C75973215
86694E35EA291A88665A2FFBAE5C9BE45938E60027C6C6DEB384F873457B5B5C7107651AE0273DF8A23FDFF74A2A6BD67703
854FA5E115712E84D6726F27370FDE6390B362142FCD93647BD2D7A3927DE5F5040F60E17C39D1248F6662CF441EE4B2470C
D412050CE09EC31075001286F401A58FE6217000D6A9C8BA753C4CE9D1D8CF8B908614DF0A81
previous_chain_secret = DDDDEB4EB4EDB9EC8E7DBA3BB90F581F87518F4C2DB2B7AF3BA49C2D505391D0
k2 = D71BD5A07C158C130283EF854516D290A46ADE09A63831C7B83B8FD0724C8FB0
context2 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label2 = LA2 ^ LB2
MA2 = cid || len(LA2) || LA2 || len(PA2) || PA2
MB2 = cid || len(LB2) || LB2 || len(PB2) || PB2

chain_secret2 = 65A510FF49080D4E9EF8AA392C14C1D9C4CCFADD781BD84D2491B7CA81D13852
key_material2 = A21B0F3D7546FFD4C2A7058AC9AE4D5B
```

D.3.4 KDF as HKDF with SHA256, ECDH with Curve25519, and ML-KEM768

```
CID 1722
LA1 = 0102030405060708090A0B0C0D0E0F1101020304050607081020304050607085
PA1 = 8520F0098930A754748B7DDCB43EF75A0DBF3A0D26381AF4EBA4A98EAA9B4E6A
LB1 = 202030405060708090A0B0C0D0E0F100202030405060708E020304050607080C
PB1 = DE9EDB7D7B7DC1B4D35B61C2ECE435373F8343C85B78674DADFC7E146F882B4F
previous_chain_secret = psk = <NULL>
k1 = 4A5D9D5BA4CE2DE1728E3BF480350F25E07E21C947D19E3376F09B3C1E161742
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label1 = LA1 ^ LB1
MA1 = cid || len(LA1) || LA1 || len(PA1) || PA1
MB1 = cid || len(LB1) || LB1 || len(PB1) || PB1
chain_secret1 = F1A0E202AC8F33A77AAF0BD39852A997DCE9EA5F4FF6882BF1D0F1536C84FA4C
key_material1 = 083D51054638A7BE42551230786E0F60

LA2 = 141020304050607080900A0B0C0D0E0F141020304050607080900A0B0C0D0E0F11
PA2 =
06F51174F03C2FE07CC31B0659B87EBF124237811B88B980BC277F8C050B265CBEE1A4AE7145CB9253281BBC291F2808E065
855FD91C7D14122C589FB21B45FD3B7FBB6562FD53006B565189EC983C7A0D4C21596EF3795AE22ECAD71F75395558B55220
662BB66C389ED9AB6B784476E1788F34B28B59A4C6F71F9F6A49DA80C4D3107DB2C306968B5600E18285F8BFA293677F699F
D2D2820DE490F530818D6667686A765FCC0009484D71629C1F148104B462325AC55EF34DD5C7098441A06757CA35C8ADFDAB
6D36333310241F9970B1B123478DCBB75AE6CA39BB84FFCB23A593C24BC54E74B4086165BB4B10AE39F45447AA4EA5F55762
386B11051B84B0A688C12C73336FC87164B7FBC7F0B07D2A51BB05560606D5558666CC322963AD3CA2BC1293260A94707770
C030B2BC9C00477A8D5FF6737961480D48109F66075DA9C8AB516F70A78DA9E444AB65798A2565965A68AD9146FD9267D2A8
0C4352A0A988157522046620A63D9BA7BBCCAFD50A61384453F8545845B25E7A05745C4B26EF7211200A916241047FDABB66
936BB7351C04F5CD06A10AE3C459B74BB0CD85044BF8B6BF2B33294014B9D68517CB1A9813230BA08813807A2DFCC6A739B6
AC79575F19BAB96A7CDED404D0E52508214AD43763A0A048FC36785B2C718F75B3432769C1D2A6240336A82B39D865739AF7
8E94A92BD7B4703210297FB82E0BE57B97384DB5B24FFA5486C982AF48642D967222A28C6F83E5AD6A59CA486733808BA3BD
86A780E59888011965A351CA949B0D6C72FB3BA0F867834D166819C252D9600207C88997318B2D204AC3619247A01829484F
12BCA2E8F16AAC677C9E8AC209F9BCCF019BBDA66E8F073DFBD934E08CA15BE77576029654E049C1C20B1B123E5F0B501356
28CD2C96A5D0B1EEF48FBE08AC4685247721B128F71DC748452B94A458F39B337BC2305CCC505BB18BF6ABE3EC2C825799B9
5C13A1F6411CD3B031F3CEC3D99AFC1A28F2019C1BD129A2EA35F06415C8B717337957CAA964E794881257073FF5C12C3C62
034861C343B7F70B9D475CCC10EA5889910E04CA7EAF0B762DB56962EC40CA47CEB4D228600595CF22421148CF02B4CFCE76
AD27701F925C01C71C5405534ED54B0100C280DD861D89E967BED4829C568BB76CA500342803835F7983179CDA15899033F7
97951031138A83669AB4CCB4630058088AA0845FAF7B0761B405AE866C426711EB152889D175E7B20DA03126B7419A766416
A68AAC0FBC355DA07D6E6339BC0539CEB55711F633B8C7392627497821CB9D7174C4E655B5928BF722C84B8233A0993B6DE0
2D7364443EA4C73A754B8190243DF936184819E1B8AB6F0B0BFBAC6C47E66DDF11757BE55E0787ADC112C63E644D1B904877
764D856ABE78B583600CBB2119CB916C689D4543F4D553CAC33AD19B245B4948EF0BA3E7D1151179B9FB2941F3674E5F8771
677737C435B2383119A3E75434114439506E1B9145FF16B1458879A1039962434151619B77915D033A3FAB7385FE861E4A25
B125A205C3F1B16972470B2791534B5FB8731B20C03684CCBEB97A1FA4417655BB485C347519710522C357D50C43FC264409
8533FE549C825F5DC8CA82A0DD908891BB4F92EAAB51CBE036CD6FB59A749269A0A3
LB2 = 202030405060708090A0B0C0D0E0F110202030405060708A0203040506070808
PB2 =
DB3D4F38A49CC5C1C7766573873F3926184923F570FCDBA6D7086BEA39601FADFEBABB2F9ED0EBD86A6468BE03D3B05713DE
50B5E5D58FFDB5500F2AF7419682BC7857870D1E624C2B48A7E0D30A91F141F0764014FBED12A9F72C958EB53A7ED4A16825
78B4348A05C183C790FF56D66B23BD601FB372C771F52045E13CA2D6DAB80A165A2159B8F990B5BBF125883DEC0A2AF4A54C
3F386DE4DF6EE96923E9BB2551753DDC6D1F00D7EE910835082C6260974D11FBA49951DB2E27B47362EC4EBF804327AD9682
01EDB71BDC8C32887214FB2AAA842AABBE0A4E03FDC695B19610377B9BEA9D371016674E977E9E45382D236365EFEDF56665
9F9936D9D14CA8595DDF121AE7ACC6056DD6FE1B58DC0F0B23CF5366D2488C4D6361B09875242B3A39374159CF3654ABD25F
8B080BAE117A5F77D15031AD8B350C6922FFD426EC9F9F80A1918FE722CB7A4B537210EF8A5538A52048BADCE3CAF5C46454
EEAF413819D57EFBA834D7017F9343B302679A0A28F532E93FE8C5B6AEC3C42C160A3FAD20FA2ABADD29C2564DB59FA25A11
```

```
CA502D11421F7F080A616F9656942932E11BFDCC47B1D8EFFA81D724FFABF4613830631737EB7E52DA78A6B026DB1C4FB80B
BF2D8F9393875330B2F9C1A124436028BF2624C1E64797ABE2E526967DD99BC21B555A5833A31ACE7969E3ECC64C1697DEC0
B074108F6B17EFA2A517923B435D8446043FB1A96B2EF3B3D3952F08657997D1D9169FFD14316185EC30457E0FAF7EBA2B5A
02A01993EC530BB9C44C091D2E9D9BC82951E577D76DC2757F2E04E822B6A29F840887A297202E7389A0999F9E69255DE0A3
3AEB8CD0B21B7F3B5AA7BFD864CF8FF4FFE78FFF166CAF1B805464D5E4B60EB0277E3BA4F3F3390640F5F16E0B784CC8A679
40977B2F0202D37A74472161589A7B9E7721CC532904A17AFF8FA86ECDC46A410BD6F9309B2CC310D7941A02A5720CD64241
B5C13C78A86010805AC74C3B81D83348C63C410451BD2052CE764DA83CF80B2654CB1D033F259DA36039E4DF1A5C4CE3F861
3E28CFC2B25EAE2A21B425B44D0C998977B3ADF5156BFCA1E357EC3F6B5805B2CDC4D46A4C83154AEF0F50D493BAB0BBCA13
0E92CA21AF8BDF817F7616B55C4C24C066EBF896F39A7BF50A241561E578BDD931CB07EADD193986C455F8798AB97B8D732D
37C796E09C1C7ABB978BF666CED961373198607AC355FCF6903B4F3047A1D97C344A2D8B73DB310769CCFB4AE3DA6C4E82DE
B83BF4060722D9E5A9AFBD3E03B4517E3B7D563F31821D5D8EF7F06F0D92A9F401EA9BEFD52CA8DB331360E603C14A340B80
6155DE690CD2D92EE6A8EE53BCEED7EEB308D375518C851C109A03EE61960D6CD7CD1FDEE5CC2A401039A15EE4F97A734008
01A090FBF09BE42119662D06C3FF8217FCFC2C70CDE6E1C2F9A396A2E08A3C22AE3CE90A5C97C9F69EBC1546898A047C9DCA
D128003A90B29B62F7B093CC90C9CD38104C1A3CE62BEA6A3646EF7E3B9FB7C4C44E78FF650C
previous_chain_secret = F1A0E202AC8F33A77AAF0BD39852A997DCE9EA5F4FF6882BF1D0F1536C84FA4C
k2 = E3E4F4E9D4F5C9AE03836BB9266C50B033285ACE9BC56F73817CE19679D1429A
context2 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label2 = LA2 ^ LB2
MA2 = cid || len(LA2) || LA2 || len(PA2) || PA2
MB2 = cid || len(LB2) || LB2 || len(PB2) || PB2

chain_secret2 = 9774DFF0962EEEDE5504442C32CE865101FB4C5EC91B6931BFDB5CECD88A9F0A
key_material2 = 2D490429EE1F9F17CB05AE44691CCD09
```

D.3.5 KDF as HMAC with SHA256, ECDH with Curve25519, and ML-KEM768

```
CID 4722
LA1 = 0102030405060708090A0B0C0D0E0F110102030405060708102030405060708 5
PA1 = 8520F0098930A754748B7DDCB43EF75A0DBF3A0D26381AF4EBA4A98EAA9B4E6A
LB1 = 202030405060708090A0B0C0D0E0F100202030405060708E020304050607080C
PB1 = DE9EDB7D7B7DC1B4D35B61C2ECE435373F8343C85B78674DADFC7E146F882B4F
previous_chain_secret = psk = <NULL>
k1 = 4A5D9D5BA4CE2DE1728E3BF480350F25E07E21C947D19E3376F09B3C1E161742
context = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label1 = LA1 ^ LB1
MA1 = cid || len(LA1) || LA1 || len(PA1) || PA1
MB1 = cid || len(LB1) || LB1 || len(PB1) || PB1
chain_secret1 = 66E1C15322867E26BCE237B5BE59BD39EE85F33A351F0E8194CB81A79457DC63
key_material1 = 320DD61D4E5345CB48764C6F39DF0E39

LA2 = 1410203040506070809 0A0B0C0D0E0F1410203040506070809 0A0B0C0D0E0F11
PA2 =
06F51174F03C2FE07CC31B0659B87EBF124237811B88B980BC277F8C050B265CBEE1A4AE7145CB9253281BBC291F2808E065
855FD91C7D14122C589FB21B45FD3B7FBB6562FD53006B565189EC983C7A0D4C21596EF3795AE22ECAD71F75395558B55220
662BB66C389ED9AB6B784476E1788F34B28B59A4C6F71F9F6A49DA80C4D3107DB2C306968B5600E18285F8BFA293677F699F
D2D2820DE490F530818D6667686A765FCC0009484D71629C1F148104B462325AC55EF34DD5C7098441A06757CA35C8ADFDAB
6D36333310241F9970B1B123478DCBB75AE6CA39BB84FFCB23A593C24BC54E74B4086165BB4B10AE39F45447AA4EA5F55762
386B11051B84B0A688C12C73336FC87164B7FBC7F0B07D2A51BB05560606D5558666CC322963AD3CA2BC1293260A94707770
C030B2BC9C00477A8D5FF6737961480D48109F66075DA9C8AB516F70A78DA9E444AB65798A2565965A68AD9146FD9267D2A8
0C4352A0A988157522046620A63D9BA7BBCCAFD50A61384453F8545845B25E7A05745C4B26EF7211200A916241047FDABB66
936BB7351C04F5CD06A10AE3C459B74BB0CD85044BF8B6BF2B33294014B9D68517CB1A9813230BA08813807A2DFCC6A739B6
AC79575F19BAB96A7CDED404D0E52508214AD43763A0A048FC36785B2C718F75B3432769C1D2A6240336A82B39D865739AF7
8E94A92BD7B4703210297FB82E0BE57B97384DB5B24FFA5486C982AF48642D967222A28C6F83E5AD6A59CA486733808BA3BD
86A780E59888011965A351CA949B0D6C72FB3BA0F867834D166819C252D9600207C88997318B2D204AC3619247A01829484F
12BCA2E8F16AAC677C9E8AC209F9BCCF019BBDA66E8F073DFBD934E08CA15BE77576029654E049C1C20B1B123E5F0B501356
28CD2C96A5D0B1EEF48FBE08AC4685247721B128F71DC748452B94A458F39B337BC2305CCC505BB18BF6ABE3EC2C825799B9
5C13A1F6411CD3B031F3CEC3D99AFC1A28F2019C1BD129A2EA35F06415C8B717337957CAA964E794881257073FF5C12C3C62
034861C343B7F70B9D475CCC10EA5889910E04CA7EAF0B762DB56962EC40CA47CEB4D228600595CF22421148CF02B4CFCE76
AD27701F925C01C71C5405534ED54B0100C280DD861D89E967BED4829C568BB76CA500342803835F7983179CDA15899033F7
97951031138A83669AB4CCB4630058088AA0845FAF7B0761B405AE866C426711EB152889D175E7B20DA03126B7419A766416
A68AAC0FBC355DA07D6E6339BC0539CEB55711F633B8C7392627497821CB9D7174C4E655B5928BF722C84B8233A0993B6DE0
2D7364443EA4C73A754B8190243DF936184819E1B8AB6F0B0BFBAC6C47E66DDF11757BE55E0787ADC112C63E644D1B904877
764D856ABE78B583600CBB2119CB916C689D4543F4D553CAC33AD19B245B4948EF0BA3E7D1151179B9FB2941F3674E5F8771
677737C435B2383119A3E75434114439506E1B9145FF16B1458879A1039962434151619B77915D033A3FAB7385FE861E4A25
B125A205C3F1B16972470B2791534B5FB8731B20C03684CCBEB97A1FA4417655BB485C347519710522C357D50C43FC264409
8533FE549C825F5DC8CA82A0DD908891BB4F92EAAB51CBE036CD6FB59A749269A0A3
LB2 = 202030405060708090A0B0C0D0E0F110202030405060708A0203040506070808
PB2 =
DB3D4F38A49CC5C1C7766573873F3926184923F570FCDBA6D7086BEA39601FADFEBABB2F9ED0EBD86A6468BE03D3B05713DE
50B5E5D58FFDB5500F2AF7419682BC7857870D1E624C2B48A7E0D30A91F141F0764014FBED12A9F72C958EB53A7ED4A16825
78B4348A05C183C790FF56D66B23BD601FB372C771F52045E13CA2D6DAB80A165A2159B8F990B5BBF125883DEC0A2AF4A54C
3F386DE4DF6EE96923E9BB2551753DDC6D1F00D7EE910835082C6260974D11FBA49951DB2E27B47362EC4EBF804327AD9682
01EDB71BDC8C32887214FB2AAA842AABBE0A4E03FDC695B19610377B9BEA9D371016674E977E9E45382D236365EFEDF56665
9F9936D9D14CA8595DDF121AE7ACC6056DD6FE1B58DC0F0B23CF5366D2488C4D6361B09875242B3A39374159CF3654ABD25F
```

```
8B080BAE117A5F77D15031AD8B350C6922FFD426EC9F9F80A1918FE722CB7A4B537210EF8A5538A52048BADCE3CAF5C46454
EEAF413819D57EFBA834D7017F9343B302679A0A28F532E93FE8C5B6AEC3C42C160A3FAD20FA2ABADD29C2564DB59FA25A11
CA502D11421F7F080A616F9656942932E11BFDCC47B1D8EFFA81D724FFABF4613830631737EB7E52DA78A6B026DB1C4FB80B
BF2D8F9393875330B2F9C1A124436028BF2624C1E64797ABE2E526967DD99BC21B555A5833A31ACE7969E3ECC64C1697DEC0
B074108F6B17EFA2A517923B435D8446043FB1A96B2EF3B3D3952F08657997D1D9169FFD14316185EC30457E0FAF7EBA2B5A
02A01993EC530BB9C44C091D2E9D9BC82951E577D76DC2757F2E04E822B6A29F840887A297202E7389A0999F9E69255DE0A3
3AEB8CD0B21B7F3B5AA7BFD864CF8FF4FFE78FFF166CAF1B805464D5E4B60EB0277E3BA4F3F3390640F5F16E0B784CC8A679
40977B2F0202D37A74472161589A7B9E7721CC532904A17AFF8FA86ECDC46A410BD6F9309B2CC310D7941A02A5720CD64241
B5C13C78A86010805AC74C3B81D83348C63C410451BD2052CE764DA83CF80B2654CB1D033F259DA36039E4DF1A5C4CE3F861
3E28CFC2B25EAE2A21B425B44D0C998977B3ADF5156BFCA1E357EC3F6B5805B2CDC4D46A4C83154AEF0F50D493BAB0BBCA13
0E92CA21AF8BDF817F7616B55C4C24C066EBF896F39A7BF50A241561E578BDD931CB07EADD193986C455F8798AB97B8D732D
37C796E09C1C7ABB978BF666CED961373198607AC355FCF6903B4F3047A1D97C344A2D8B73DB310769CCFB4AE3DA6C4E82DE
B83BF4060722D9E5A9AFBD3E03B4517E3B7D563F31821D5D8EF7F06F0D92A9F401EA9BEFD52CA8DB331360E603C14A340B80
6155DE690CD2D92EE6A8EE53BCEED7EEB308D375518C851C109A03EE61960D6CD7CD1FDEE5CC2A401039A15EE4F97A734008
01A090FBF09BE42119662D06C3FF8217FCFC2C70CDE6E1C2F9A396A2E08A3C22AE3CE90A5C97C9F69EBC1546898A047C9DCA
D128003A90B29B62F7B093CC90C9CD38104C1A3CE62BEA6A3646EF7E3B9FB7C4C44E78FF650C
previous_chain_secret = 66E1C15322867E26BCE237B5BE59BD39EE85F33A351F0E8194CB81A79457DC63
k2 = E3E4F4E9D4F5C9AE03836BB9266C50B033285ACE9BC56F73817CE19679D1429A
context2 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label2 = LA2 ^ LB2
MA2 = cid || len(LA2) || LA2 || len(PA2) || PA2
MB2 = cid || len(LB2) || LB2 || len(PB2) || PB2

chain_secret2 = FDC9BA444B6BF90BFC286635EB55268F344EC32AA7DB1BDA6B0F4E61790A0344
key_material2 = 37A9900F776007E7FBE40A5486322855
```

D.3.6 KDF as KMAC128, ECDH with Curve25519, and ML-KEM768

```
CID 7722
LA1 = 0102030405060708090A0B0C0D0E0F1101020304050607081020304050607085
PA1 = 8520F0098930A754748B7DDCB43EF75A0DBF3A0D26381AF4EBA4A98EAA9B4E6A
LB1 = 202030405060708090A0B0C0D0E0F100202030405060708E020304050607080C
PB1 = DE9EDB7D7B7DC1B4D35B61C2ECE435373F8343C85B78674DADFC7E146F882B4F
previous_chain_secret = psk = <NULL>
k1 = 4A5D9D5BA4CE2DE1728E3BF480350F25E07E21C947D19E3376F09B3C1E161742
context1 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label1 = LA1 ^ LB1
MA1 = cid || len(LA1) || LA1 || len(PA1) || PA1
MB1 = cid || len(LB1) || LB1 || len(PB1) || PB1

chain_secret1 = C92543C00DEB478BDB6416DDEC53020ADA54F4BE288DDD7EA66D5DE96B49A880
key_material1 = 2491C2FBEE1FAAAB2092BBAD5EC91EC7

LA2 = 14102030405060708090A0B0C0D0E0F14102030405060708090A0B0C0D0E0F11
PA2 =
06F51174F03C2FE07CC31B0659B87EBF124237811B88B980BC277F8C050B265CBEE1A4AE7145CB9253281BBC291F2808E065
855FD91C7D14122C589FB21B45FD3B7FBB6562FD53006B565189EC983C7A0D4C21596EF3795AE22ECAD71F75395558B55220
662BB66C389ED9AB6B784476E1788F34B28B59A4C6F71F9F6A49DA80C4D3107DB2C306968B5600E18285F8BFA293677F699F
D2D2820DE490F530818D6667686A765FCC0009484D71629C1F148104B462325AC55EF34DD5C7098441A06757CA35C8ADFDAB
6D36333310241F9970B1B123478DCBB75AE6CA39BB84FFCB23A593C24BC54E74B4086165BB4B10AE39F45447AA4EA5F55762
386B11051B84B0A688C12C73336FC87164B7FBC7F0B07D2A51BB05560606D5558666CC322963AD3CA2BC1293260A94707770
C030B2BC9C00477A8D5FF6737961480D48109F66075DA9C8AB516F70A78DA9E444AB65798A2565965A68AD9146FD9267D2A8
0C4352A0A988157522046620A63D9BA7BBCCAFD50A61384453F8545845B25E7A05745C4B26EF7211200A916241047FDABB66
936BB7351C04F5CD06A10AE3C459B74BB0CD85044BF8B6BF2B33294014B9D68517CB1A9813230BA08813807A2DFCC6A739B6
AC79575F19BAB96A7CDED404D0E52508214AD43763A0A048FC36785B2C718F75B3432769C1D2A6240336A82B39D865739AF7
8E94A92BD7B4703210297FB82E0BE57B97384DB5B24FFA5486C982AF48642D967222A28C6F83E5AD6A59CA486733808BA3BD
86A780E59888011965A351CA949B0D6C72FB3BA0F867834D166819C252D9600207C88997318B2D204AC3619247A01829484F
12BCA2E8F16AAC677C9E8AC209F9BCCF019BBDA66E8F073DFBD934E08CA15BE77576029654E049C1C20B1B123E5F0B501356
28CD2C96A5D0B1EEF48FBE08AC4685247721B128F71DC748452B94A458F39B337BC2305CCC505BB18BF6ABE3EC2C825799B9
5C13A1F6411CD3B031F3CEC3D99AFC1A28F2019C1BD129A2EA35F06415C8B717337957CAA964E794881257073FF5C12C3C62
034861C343B7F70B9D475CCC10EA5889910E04CA7EAF0B762DB56962EC40CA47CEB4D228600595CF22421148CF02B4CFCE76
AD27701F925C01C71C5405534ED54B0100C280DD861D89E967BED4829C568BB76CA500342803835F7983179CDA15899033F7
97951031138A83669AB4CCB4630058088AA0845FAF7B0761B405AE866C426711EB152889D175E7B20DA03126B7419A766416
A68AAC0FBC355DA07D6E6339BC0539CEB55711F633B8C7392627497821CB9D7174C4E655B5928BF722C84B8233A0993B6DE0
2D7364443EA4C73A754B8190243DF936184819E1B8AB6F0B0BFBAC6C47E66DDF11757BE55E0787ADC112C63E644D1B904877
764D856ABE78B583600CBB2119CB916C689D4543F4D553CAC33AD19B245B4948EF0BA3E7D1151179B9FB2941F3674E5F8771
677737C435B2383119A3E75434114439506E1B9145FF16B1458879A1039962434151619B77915D033A3FAB7385FE861E4A25
B125A205C3F1B16972470B2791534B5FB8731B20C03684CCBEB97A1FA4417655BB485C347519710522C357D50C43FC264409
8533FE549C825F5DC8CA82A0DD908891BB4F92EAAB51CBE036CD6FB59A749269A0A3
LB2 = 202030405060708090A0B0C0D0E0F110202030405060708A0203040506070808
PB2 =
DB3D4F38A49CC5C1C7766573873F3926184923F570FCDBA6D7086BEA39601FADFEBABB2F9ED0EBD86A6468BE03D3B05713DE
50B5E5D58FFDB5500F2AF7419682BC7857870D1E624C2B48A7E0D30A91F141F0764014FBED12A9F72C958EB53A7ED4A16825
78B4348A05C183C790FF56D66B23BD601FB372C771F52045E13CA2D6DAB80A165A2159B8F990B5BBF125883DEC0A2AF4A54C
```

```
3F386DE4DF6EE96923E9BB2551753DDC6D1F00D7EE910835082C6260974D11FBA49951DB2E27B47362EC4EBF804327AD9682
01EDB71BDC8C32887214FB2AAA842AABBE0A4E03FDC695B19610377B9BEA9D371016674E977E9E45382D236365EFEDF56665
9F9936D9D14CA8595DDF121AE7ACC6056DD6FE1B58DC0F0B23CF5366D2488C4D6361B09875242B3A39374159CF3654ABD25F
8B080BAE117A5F77D15031AD8B350C6922FFD426EC9F9F80A1918FE722CB7A4B537210EF8A5538A52048BADCE3CAF5C46454
EEAF413819D57EFBA834D7017F9343B302679A0A28F532E93FE8C5B6AEC3C42C160A3FAD20FA2ABADD29C2564DB59FA25A11
CA502D11421F7F080A616F9656942932E11BFDCC47B1D8EFFA81D724FFABF4613830631737EB7E52DA78A6B026DB1C4FB80B
BF2D8F9393875330B2F9C1A124436028BF2624C1E64797ABE2E526967DD99BC21B555A5833A31ACE7969E3ECC64C1697DEC0
B074108F6B17EFA2A517923B435D8446043FB1A96B2EF3B3D3952F08657997D1D9169FFD14316185EC30457E0FAF7EBA2B5A
02A01993EC530BB9C44C091D2E9D9BC82951E577D76DC2757F2E04E822B6A29F840887A297202E7389A0999F9E69255DE0A3
3AEB8CD0B21B7F3B5AA7BFD864CF8FF4FFE78FFF166CAF1B805464D5E4B60EB0277E3BA4F3F3390640F5F16E0B784CC8A679
40977B2F0202D37A74472161589A7B9E7721CC532904A17AFF8FA86ECDC46A410BD6F9309B2CC310D7941A02A5720CD64241
B5C13C78A86010805AC74C3B81D83348C63C410451BD2052CE764DA83CF80B2654CB1D033F259DA36039E4DF1A5C4CE3F861
3E28CFC2B25EAE2A21B425B44D0C998977B3ADF5156BFCA1E357EC3F6B5805B2CDC4D46A4C83154AEF0F50D493BAB0BBCA13
0E92CA21AF8BDF817F7616B55C4C24C066EBF896F39A7BF50A241561E578BDD931CB07EADD193986C455F8798AB97B8D732D
37C796E09C1C7ABB978BF666CED961373198607AC355FCF6903B4F3047A1D97C344A2D8B73DB310769CCFB4AE3DA6C4E82DE
B83BF4060722D9E5A9AFBD3E03B4517E3B7D563F31821D5D8EF7F06F0D92A9F401EA9BEFD52CA8DB331360E603C14A340B80
6155DE690CD2D92EE6A8EE53BCEED7EEB308D375518C851C109A03EE61960D6CD7CD1FDEE5CC2A401039A15EE4F97A734008
01A090FBF09BE42119662D06C3FF8217FCFC2C70CDE6E1C2F9A396A2E08A3C22AE3CE90A5C97C9F69EBC1546898A047C9DCA
D128003A90B29B62F7B093CC90C9CD38104C1A3CE62BEA6A3646EF7E3B9FB7C4C44E78FF650C
previous_chain_secret = C92543C00DEB478BDB6416DDEC53020ADA54F4BE288DDD7EA66D5DE96B49A880
k2 = E3E4F4E9D4F5C9AE03836BB9266C50B033285ACE9BC56F73817CE19679D1429A
context2 = "ETSI_QSHKE_TEST_VECTORS_V_1_2"
label2 = LA2 ^ LB2
MA2 = cid || len(LA2) || LA2 || len(PA2) || PA2
MB2 = cid || len(LB2) || LB2 || len(PB2) || PB2

chain_secret2 = C4A9F090FA80C91A082150B0B5F445F5549F7C9EBC0E0642B794F68851EFD06B
key_material2 = BF7487D94D53B67C9F73A40293481833
```

Annex E (informative):
Change history

Date	Version	Information about changes
January 2022	V1.1.5	Updated to include static-ephemeral use case, and an abstraction of the context formatting function.
March 2022	V1.1.6	Minor edits based on working group feedback.
September 2022	V1.1.7	Aligned draft to latest NIST selection, removed reference to schemes no longer being considered for NIST standardization.
May 2023	V1.1.8	Removed a context formatting function type because of incompatibility with current implementations. Updated references.
September 2023	V1.1.9	Added latest references for FIPS 203 (draft) and Hybrid KEM proof paper. Updated references through out the document. Security consideration is augmented to include a paragraph about IND-CCA for CatKDF.
February 2024	V1.1.10	Added description of KMAC as a KDF function.
September 2024	V1.1.11	Added in fixed parameter sets for curves and KEMs. Updated specification to restrict KEMs to FIPS 203 ML-KEM standard. Removed NIST P521 and brainpool curves. Restored the previous concatenate-based formatting function, and restored the PRF to KMAC mapping.
December 2024	V1.1.12	Refined the fixed parameter sets. Re-ordered some of the sections to have a single description for CasKDF and CatKDF. Added an Annex C on message formatting used for test vectors. Small modification for CasKDF to accommodate KMAC implementation detail.
January 2025	V1.1.13	Added HMAC KDF, added error conditions to formatting functions, fixed label lengths when used, strengthened the language around authenticating messages, and fixed the number of key establishment schemes and order for the shared secrets.
February 2025	V1.1.14	Addressed comments received from on V.1.1.13 (removing "shall" statements from notes, adding motivation for KDFs, and adding missing HMAC reference as a KDF).

History

Document history		
V1.1.1	December 2020	Publication
V1.2.1	March 2025	Publication