ETSI TS 103 744 V1.1.1 (2020-12)

CYBER;
Quantum-safe Hybrid Key Exchanges

Reference
DTS/CYBER-QSC-0015

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, 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.

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 Exchanges are constructions that combine a traditional key exchange, such as elliptic curve Diffie Hellman [1], with a quantum-safe key exchange such as Supersingular Isogeny Key Establishment (SIKE) [i.17], into a single key exchange. Hybrid key exchanges 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 classical key agreement 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 at https://docbox.etsi.org/Reference.

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".

NOTE: Available at https://nvlpubs.nist.gov/nistpubs/SpecialPublications/nist.sp.800-56Ar3.pdf.

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

NOTE: Available at https://tools.ietf.org/html/rfc2104.

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

NOTE: Available at https://tools.ietf.org/html/rfc5869.

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

NOTE: Available at https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf.

[5] FIPS PUB 202: "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions".

NOTE: Available at https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf.

[6] FIPS PUB 186-4: "Digital Signature Standard (DSS)".

NOTE: Available at https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf.

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

NOTE: Available at https://tools.ietf.org/html/rfc5639.

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

NOTE: Available at https://tools.ietf.org/html/rfc7748.

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] NIST Post Quantum Round 2 Submission: "BIKE: Bit Flipping Key Encapsulation", Round 3 Submission, 22 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.2] NIST Post Quantum Round 3 Submission: "Classic McEliece: conservative code-based cryptography", 10 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.3] NIST Post Quantum Round 3 Submission: "CRYSTALS-Kyber", Version 3.0, 1 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.4] NIST Post Quantum Round 3 Submission: "FrodoKEM Learning With Errors Key Encapsulation", 30 September 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.5] NIST Post Quantum Round 3 Submission: "Hamming Quasi-Cyclic (HQC)", 1 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.6] NIST Post Quantum Round 3 Submission: "NTRU", 30 September 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.7] NIST Post Quantum Round 3 Submission: "NTRU Prime: round 3", 7 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.8] NIST Post Quantum Round 3 Submission: "SABER: Mod-LWR based KEM (Round 3 Submission)", Accessed 26 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.9] NIST Post Quantum Round 3 Submission: "Supersingular Isogeny Key Encapsulation", 1 October 2020.

NOTE: Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[i.10] 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.11] F. Giacon, F. Heuer, B. Poettering: "KEM Combiners", Public-Key Cryptography - PKC 2018, LNCS 10769.

NOTE: Available at https://eprint.iacr.org/2018/024.pdf.

[i.12] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, D. Stebila: "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange", IACR eprint 2018-903.

NOTE: Available at https://eprint.iacr.org/2018/903.pdf.

[i.13] N. Bindel, U. Herath, M. McKague, D. Stebila: "Transitioning to a Quantum-Resistant Public Key Infrastructure", Post-Quantum Cryptography, 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands Proceedings. pp. 384-405. Springer International Publishing, Cham (2017).

[i.14] 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.

NOTE: Available at https://doi.org/10.1109/SFCS.1994.365701.

[i.15] 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.

NOTE: Available at https://dl.acm.org/doi/abs/10.1109/SFCS.1994.365700.

[i.16] NIST CAVP SP 800-56A ECC CDH Primitive Test Vectors.

NOTE: Available at https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/components/ecccdhtestvectors.zip.

[i.17] SIKE Round 2 Known Answers Tests (KATs).

NOTE: Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/SIKE-Round2.zip.

[i.18] Campagna, M., Petcher, A.: "Security of Hybrid Key Encapsulation", IACR eprint 2020-1364.

NOTE: Available at https://eprint.iacr.org/2020/1364.pdf.

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 which 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 one of the processes defined or referenced in the present document

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 that 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: method to secure the establishment of a cryptographic key for transmission using public key cryptography

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 application or instance of the key derivation or exchange

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: measure of the strength of a cryptographic algorithm. If 2^n operations are required to break the cryptographic algorithm/scheme/method, then the security level is *n*. 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
C	A ciphertext value created by a KEM
d	A private key for elliptic curve cryptography
k	A cryptographic secret or key
P	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
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
LNCS	Lecture Notes in Computer Science
MA	Message from entity A
MB	Message from entity B
NIST	National Institute of Standards and Technology
OW-CPA	One-Way Chosen-Plaintext Attack
PRF	PseudoRandom Function
QA	A public-key from entity A
QB	A public-key from entity B
QKD	Quantum Key Distribution
RSA	Rivest, Shamir and Adelman
SIKE	Supersingular Isogeny Key Encapsulation
SP	Special Publication
SSH	Secure Shell
TLS	Transport Layer Security

4 Purpose of quantum-safe hybrid key exchanges

4.1 Status of quantum-safe key exchange protocols

NIST has initiated a process of analysing and standardizing one or more new quantum-safe key encapsulation mechanisms suitable to replace classical key exchanges. At the time of the present document, there are 9 round 3 post-quantum KEMs still under consideration [i.1] to [i.9].

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.

The quantum-safe hybrid key exchanges specified in the present document ensure that the derived key is at least as secure as the maximum security of the key exchange method. The resulting hybrid scheme will remain secure if one of the key exchange methods 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 exchange

5.1 Functional entities

There are two entities defined for quantum-safe hybrid key exchange, an Initiator *A* that initiates a key exchange mechanism, 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 initiates a hybrid key exchange by the transmission of a message to *B*. *B* responds to this message. The exchange between the entities can consist of a single message in each direction or multiple rounds of messages.

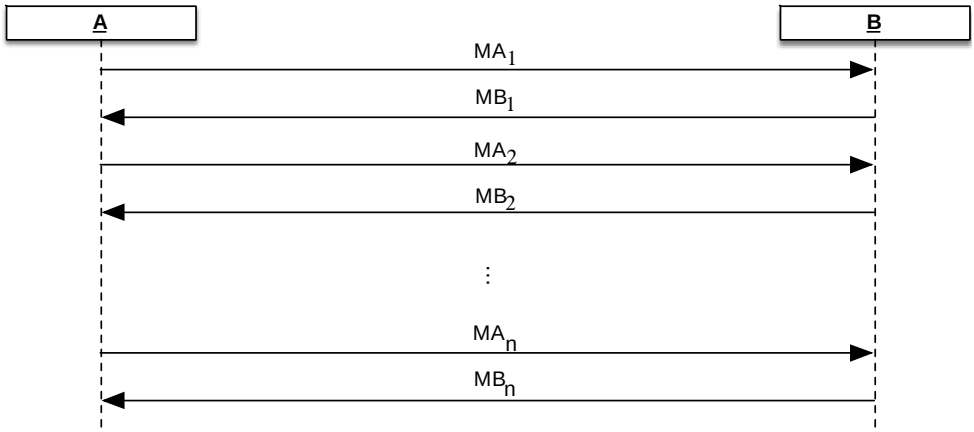

Figure 2: Messages exchanged between entities *A* and *B*

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

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

6 Introductory information

6.1 Introduction

Quantum-safe hybrid key exchange mechanisms combine a classic key exchange method like ECDH and a quantum-safe key-encapsulation mechanism (KEM). The hybrid exchange mechanisms 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 the hybrid key exchanges specified in the present document provide both the key expansion property and random extraction as per Crypto 04, LNCS 3152 [i.10].

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:

0x0123456789ABCDEFFEDCBA98765432l086545381AB594FC28786404C50A37…

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, SHA-512, SHA-512/256 as defined in FIPS PUB 180-4 [4].
- SHA3-256, SHA3-384, SHA3-512 as defined in FIPS PUB 202 [5].

7.2 Context formatting function (*f*)

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

$$f\_context = f(val_1, val_2, \ldots)$$

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, 64}).

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* = ∅
2) For $i = 1, \ldots, n$
   a) Set $len_i = len(val_i)$, returns the length of val_i in octets:
      i) $L_i = [len_i]_{32}$ - a 32-bit integer value expressed as 4 octets
      ii) $V_i = val_i$
   b) Set $buffer = buffer \,||\, L_i \,||\, V_i$
3) Set *f_context = hash(buffer)*

Output:

f_context - a *digest_len* octet string representing the context value for a KDF.

NOTE: In practice, the hash can be iterated and a buffer does not need to be allocated.

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 hash function from clause 7.1.

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 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:

f - a context formatting function as defined in clause 7.2.

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:

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

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 octet string. This label is defined by the application.

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 hash function from clause 7.1.

7.4.2 KDF to HKDF mapping

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

HKDF is defined in IETF RFC 5869 [3] and has the 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).

7.5 Elliptic Curve Diffie-Hellman (ECDH)

7.5.1 ECDH description

One of the key-agreement 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-agreement 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 D.1.2.4 of FIPS PUB 186-4 [6].
- NIST P-256/secp256r1 defined in clause D.1.2.3 of FIPS PUB 186-4 [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)*.

- *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 ⊥ an error indicator.

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 include the variants in the 9 round 3 post-quantum KEMs still under consideration [i.1] to [i.18].

8 Hybrid key agreement schemes

8.1 General

8.1.1 Key exchange abstraction

The present clause specifies how to combine two or more key exchange mechanisms into a hybrid key agreement 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 exchange 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.

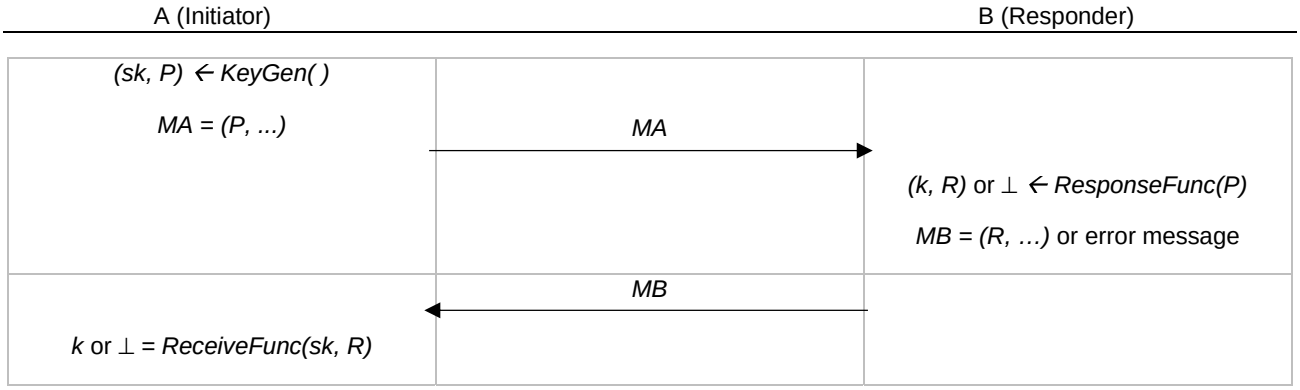

Figure 3: Key exchange abstraction

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.

Collectively such an instance of a key exchange is named a *key exchange 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 exchange abstraction to ECDHE

This clause specifies the mapping of the key exchange 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)*.

8.1.3 Key exchange abstraction to KEM

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

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

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

The *ReceiveFunc* shall be the *Decaps( )* function, k or $\perp$ = *ReceiveFunc(sk, R) = Decaps(sk, R)*.

8.2 Concatenate hybrid key agreement scheme

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

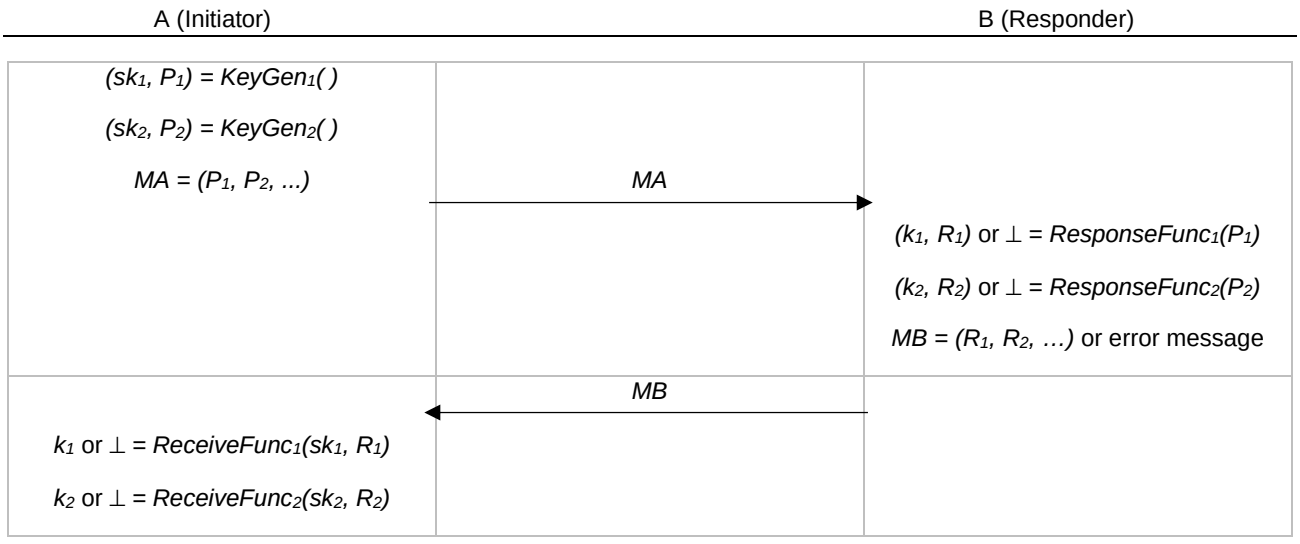

Figure 4: Concatenate hybrid key agreement

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. In the case where more than two key establishment schemes are being used, *MA* shall contain all of the public keys.

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.

The CatKDF mode shall be defined as follows.

Fixed values:

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

f - The context formatting function being used from clause 7.2 and the underlying hash function *hash* and *digest_len*.

Input:

psk - a secret key. It may be present. If not present this value shall be the empty octet string, ∅.

(k_1, k_2, …, k_n) - *n*-tuple of octet strings containing shared secrets k_i, exchanged through a hybrid key exchange, see Figure 4.

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

context - octet string context set by the instance of the key exchange transaction - this may include a transcript of additional exchanged messages.

label - an octet string that specifies a separation of use for the application or instance of the key-exchange. Any labels used in the key exchange 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 $secret = psk \,||\, k_1 \,||\, k_2 \,||\, \ldots \,||\, k_n$.
2) Set *f_context = f(context, MA, MB)*, where *f* is a context formatting function.
3) *key_material = KDF(secret, label, f_context, length).*
4) Return *key_material*.

Output:

key_material - derived key material.

NOTE: For a given set of key exchange mechanisms, the lengths of the k_i's are independent of the execution of the protocol, i.e. k_1 will be $length_1$, k_2 will be $length_2$, etc.

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

8.3 Cascade hybrid key agreement scheme

This clause specifies the cascade hybrid key agreement scheme and the cascade KDF CasKDF. The cascade hybrid key agreement scheme is depicted in Figure 5.

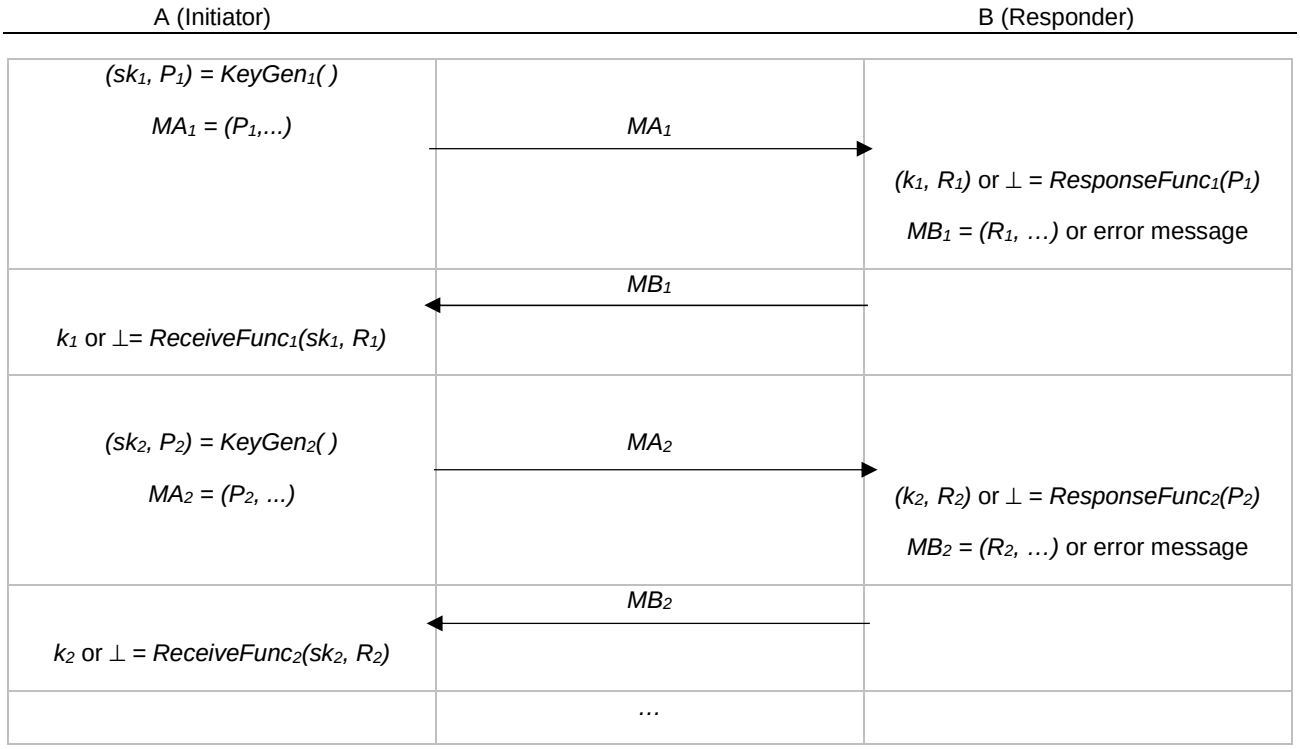

Figure 5: Cascade hybrid key exchange

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.

MB_i - shall be an octet string containing and encoding of the response value R_i. MB_i may include session negotiation information.

Two or more rounds may occur in the cascade model.

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 and the underlying hash function *hash* and *digest_len*.

Input:

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

$(k_1, k_2, \ldots, k_n)$ - a sequence of octet strings containing shared secrets k_i, exchanged through key exchange mechanisms, see Figure 5.

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

$context_i$ - a context set by the instance of the key exchange transaction to establish secret *i* - this may include a transcript of additional exchanged messages.

$label_i$ - an octet string that specifies a separation of use for the application or instance of the key-exchange. Any labels used in the key exchange 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 keys for *i = 1, …, n*.

Process:

1) *$chain\_secret_0$ = psk* a *digest_len* value or the empty octet string, ∅.
2) For *i = 1,…, n:*
   a) $round\_secret_i = PRF(chain\_secret_{i-1}, k_i, MA_i, MB_i)$
   b) $chain\_secret_i \| key\_material_i = KDF(round\_secret_i, label_i, context_i, digest\_len + length_i)$
3) Return *$chain\_secret_1$, $chain\_secret_2$, …, $chain\_secret_n$,* and *$key\_material_1$, $key\_material_2$,…, $key\_material_n$*.

Output:

$key\_material_1$, $key\_material_2$,…, $key\_material_n$ - a sequence of intermediate and final keys of length *$length_1$,…., $length_n$.*

$chain\_secret_1$, $chain\_secret_2$, …, $chain\_secret_n$ - a sequence of chain secrets.

The KDF is run *n* times, with each time injecting the shared secret from the next key exchange.

NOTE: This key derivation function can be useful if the separate key exchanges are required to be performed in distinct messages.

The intermediate key material, *$key\_material_i$* may be used in the exchange of messages *MA_j and MB_j for $j > i$*.

EXAMPLE: For instance, *$key\_material_i$* may be a set of keys used to protect the key exchanges in *MA_{i+1}* and *MB_{i+1}*.

Annex A (informative): Background

A.1 Quantum computing threats to classical 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^n complex 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.14] for the hidden subgroup problem, to factor large semiprimes and solve the discrete log problem [i.15]. 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 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 us with perfect forward secrecy, quantum-safe KEMs promise to deliver *quantum forward secrecy* - the property of a key agreement 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 exchanges

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 exchanges pair a high-assurance but quantum-vulnerable key exchange with a potentially quantum-safe key exchange 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 exchanges.

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.

Both CatKDF and CasKDF are IND-CPA secure in the random oracle model as long as at least one underlying scheme is OW-CPA. Because the values produced by the KeyGen algorithms are not reused, IND-CPA is sufficient to ensure the confidentiality of the exchanged key, and stronger security notions such as indistinguishability under chosen ciphertext attack are unnecessary.

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 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.18].

Annex C (informative): Test Vectors

C.1 Introduction

Portions of the test vectors are taken from NIST CAVP SP 800-56A ECC CDH Primitive Test Vectors [i.16], and SIKE Round 2 KATs [i.17]. PA1, and PB1 values are from ECC CDH test vectors, and PA2 and PB2 values are from SIKE Round 2 KATs.

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 1.1.1d libcrypto.

To build the code at the command line execute:

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

To run the code at the command line execute:

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

C.2 Test vectors for CatKDF

C.2.1 ECDH with NIST P-256, SIKEp434, and SHA-256

```
LA = 0102030405060708090A0B0C0D0E0F10
PA1 = 119F2F047902782AB0C9E27A54AFF5EB9B964829CA99C06B02DDBA95B0A3F6D08F52B726664CAC366FC
      98AC7A012B2682CBD962E5ACB544671D41B9445704D1D
PA2 = 4484D7AADB44B40CC180DC568B2C142A60E6E2863F5988614A6215254B2F5F6F79B48F329AD1A2DED2
      0B7ABAB10F7DBF59C3E20B59A700093060D2A44ACDC0083A53CF0808E0B3A827C45176BEE0DC6EC7CC1
      6461E38461C12451BB95191407C1E942BB50D4C7B25A49C644B630159E6C403653838E689FBF4A7ADEA
      693ED0657BA4A724786AF7953F7BA6E15F9BBF9F5007FB711569E72ACAB05D3463A458536CAB647F00C
      205D27D5311B2A5113D4B26548000DB237515931A040804E769361F94FF0167C78353D2630A1E6F595A
      1F80E87F6A5BCD679D7A64C5006F6191D4ADEFA1EA67F6388B7017D453F4FE2DFE80CCC709000B52175
      BFC3ADE52ECCB0CEBE1654F89D39131C357EACB61E5F13C80AB0165B7714D6BE6DF65F8DE73FF47B7F3
      304639F0903653ECCFA252F6E2104C4ABAD3C33AF24FD0E56F58DB92CC66859766035419AB2DF600
LB = 0202030405060708090A0B0C0D0E0F10
PB1 = 809F04289C64348C01515EB03D5CE7AC1A8CB9498F5CAA50197E58D43A86A7AEB29D84E811197F25EBA
      8F5194092CB6FF440E26D4421011372461F579271CDA3
PB2 = 0FDEB26DBD96E0CD272283CA5BDD1435BC9A7F9AB7FC24F83CA926DEED038AE4E47F39F9886E0BD7EEB
      EAACD12AB435CC92AA3383B2C01E6B9E02BC3BEF9C6C2719014562A96A0F3E784E3FA44E5C62ED8CEA7
      9E1108B6FECD5BF8836BF2DAE9FEB1863C4C8B3429220E2797F601FB4B8EBAFDD4F17355508D259CA60
      721D167F6E5480B5133E824F76D3240E97F31325DBB9A53E9A3EEE2E0712734825615A027857E2000D4
      D00E11988499A738452C93DA895BFA0E10294895CCF25E3C261CBE38F5D7E19ABE4E322094CB8DEC5BF
      7484902BABDE33CC69595F6013B20AABA9698C1DEA2BC6F65D57519294E6FEEA3B549599D480948374D
      2D21B643573C276E1A5B0745301F648D7982AB46A3065639960182BF365819EFC0D4E61E87D2820DBC0
      E849E99E875B21501D1CA7588A1D458CD70C7DF793D4993B9B1679886CAE8013A8DD854F010A100C993
      FA642DC0AEA9985786ED36B98D3
k1 = 057D636096CB80B67A8C038C890E887D1ADFA4195E9B3CE241C8A778C59CDA67
k2 = 35F7F8FF388714DEDC41F139078CEDC9
context = "CONCATENATION TEST VECTOR 1"
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = 5C366F23281D33EB85CAB026D3D9A35A
```

C.2.2 ECDH with NIST P-256, SIKEp434, and SHA3-256

The input test vectors for clause C.2.2 are the same as clause C.2.1, only the output changes.

```
key_material = A808667CC0275AEB4A646E442F13E8D0
```

C.2.3 ECDH with NIST P-384, SIKEp503 and SHA-384

```
LA = 1102030405060708090A0B0C0D0E0F100102030405060708
PA1 = 9803807F2F6D2FD966CDD0290BD410C0190352FBEC7FF6247DE1302DF86F25D34FE4A97BEF60CFF5483
      55C015DBB3E5FBA26CA69EC2F5B5D9DAD20CC9DA711383A9DBE34EA3FA5A2AF75B46502629AD54DD8B7
      D73A8ABB06A3A3BE47D650CC99
PA2 = 05279D27FF7E3A38ABB05DCFE23B5831C030D832D3EAE35FE06A6538597532D22A0F4012FB2263E1604
      95F8291B58D9DF8A8947C7CF3E6735520BB2D09491240882985 1AC4B85AA922069F2AAA0A4827DFA473
      0E9CF05485CBEE411C3D5169DD4953746B6A2E6574957EF920596B1612BE62A883740B5A0C157117AE1
      C3A07E4CE8CCCE7E9E88FE7C20A507FF019AE0893F34303E173D291F6CB7ECB4C3901FF34A48DE40771
      F5BAD72DA2B4C1CFD0A61F33E39327A8DA60F5640D4C2E71EF9C7297A4E9BC50493E3BA65D3664610A6
      D61035CB6600378D017D1E1810ACD113252D60F5915749C2B5CFB4452C40C86F1F40C63297DCCA90068
      6F2D2266F9444539D9BA13B1F52FB2FC3BD4F3EDAA6EB707AAFCA5261EA271ED961B2ED195D5E3B0299
      179251866CE0EAA31C5C90B7999A8D63BA2DE84A8AFA19F11F2DC0CACA39B982CE053F71D269931D9EE
      26BCE592A8EA818553BC8F8D244F62FB4F5E5386E3EFF5CD231401C9EC2BA57FF42DC3B3791357A53E1
      E31394008
LB = 1202030405060708090A0B0C0D0E0F100202030405060708
PB1 = A7C76B970C3B5FE8B05D2838AE04AB47697B9EAF52E764592EFDA27FE7513272734466B400091ADBF2D
      68C58E0C50066AC68F19F2E1CB879AED43A9969B91A0839C4C38A49749B661EFEDF243451915ED0905A
      32B060992B468C64766FC8437A
PB2 = 100692A8BD30F01BE8AC6B1AF8D93A060D3821B2587F4038D64B72426A194BEDE63CA60B75A5C3C1553
      2CE307115AA9D77AC232E14D99C1E1AFEF1EB2D6321AE342F990023466E683A2568D59A14325C2C6C27
      2029741D8E36976D1804059BC06B802F3A495EA50D0DBBA93FD263F4CF30BDB5F783BA6A0775715B05F
      700C85B316F7AA1A1624973885941DBFF91316BF47AC698E11D6B2418F553379D67A00F784B8643FB8A
      94029584391D488775EB4414A5E6E8122B0F282D900F3D05775F1DD994FB232ED826106203CD3433967
      F60FF925DF9E86CB376CAB5FD90B132E425682741F6AF078E75792CB4CE085D44993CFB6A4ED5AA3541
      640A0A67687922B92382CAC47C6AD358011A269CC7C17CE651CA2E2393F7DFE19D7054FEF69610A353D
      676B1F076549510590D406AD13F4A3292CCF206DBDAE47F08D448CC006449F27C1FB54E9C9E6F16ED2F
      3D120DD5AA2620D76690F00E31904C601310C76A843A58E1AEB9C5F515FCEC482C08205FDE99A89E644
      85EBBD43EEFE2E24D18EEE8F20DF6E113C6667512E28396862C98F5C0
k1 = 5F9D29DC5E31A163060356213669C8CE132E22F57C9A04F40BA7FCEAD493B457E5621E766C40A2E3D4D6
      A04B25E533F1
k2 = AF1280151C2C59B4D4150B18BA7F71590523CEA83C9BDDDA
context = "CONCATENATION TEST VECTOR 1"
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = F79E08982B3D2B7C2D0B27F921CCF71C0E3EE98877DA225A
```

C.2.4 ECDH with NIST P-384, SIKEp503, and SHA3-384

The input test vectors for clause C.2.4 are the same as clause C.2.3, only the output changes.

```
key_material = 6D3F348DAB7EB85F043288112862AD16A10A70F65E73F171
```

C.2.5 ECDH with NIST P-384, SIKEp610 and SHA-384

```
LA = 2102030405060708090A0B0C0D0E0F100102030405060709

PA1 = EA4018F5A307C379180BF6A62FD2CECEEBEEB7D4DF063A66FB838AA35243419791F7E2C9D4803C9319A
      A0EB03C416B6668835A91484F05EF028284DF6436FB88FFEBABCDD69AB0133E6735A1BCFB37203D10D3
      40A8328A7B68770CA75878A1A6
PA2 = 671B24769304DD18C97AF0C5DE741C53E0B45A9E18C7A13A15C1758125E41605587E450F8452A2BF98B
      51C2AF6B0503CB8E01F8553C36079EBFADF4948FFA063ABF4866E7AB9B9D4C9A07CA400C613607E6DB9
      BB6E7EB8ECA78894C7C8CE9E231B33179B2946C5C5BE1C783FA6AEA218F5EC4B4E6F914E5ED3724C5D7
      B79403F68438A40775E964C1B2C7D22E11A6C07474EB5D4CFF75965B400167E069FA9908A562DBABF5E
      30FED46BBA0A208ED4E50764CF320FB8556F07C7F6268084476A47D83B085DC77EB3CD30A2B5EE1E582
      9738077D52A0D7A4149EE9C1A70269BC047B4BE7E5B28007DEF74A4D813853396708A3A8498CC862F54
      015B79047014639EB8CA3BB786B27A2CFAF31E6BB9CCB152BEB3232465206973668597AA35EE1940A31
      6F71241FA40D1AC233931E1967E79AAA600AA6D83FEC6280A63924E7375F22F7A47E1DE483FEA17E0DA
      CBAEDBB13D58C0DC9BC21F2DC9525D46E4210AC5D88567E4F23304EA5BE08D89D57A0246EA21C0CD28C
      096366D7F3C8D98F5A1FB00FE2F3A183E53A7E8B6C19E9BF979E8D20C703C957D6F06A142BE86A0A09B
      05ED40953BBD7A15E92098633941730DEB5BC1C5F5154E8BCA38E035580E101E6EE858D91BD8462B906
      EB2004C6E01
LB = 220203040506070809 0A0B0C0D0E0F100202030405060709
PB1 = 30F43FCF2B6B00DE53F624F1543090681839717D53C7C955D1D69EFAF0349B7363ACB447240101CBB3A
      F6641CE4B88E025E46C0C54F0162A77EFCC27B6EA792002AE2BA82714299C860857A68153AB62E525EC
      0530D81B5AA15897981E858757
PB2 = FB75E7D835313132AC0B29D8732F1F62E6DD10BBF30375B4A50C7B153431BAE6259E1C5526C07164E87
      EDC70E4F0D8331D73285661D1F639D216372D05B4583C1302932B03FF184D115D0B250297FF26AE81DF
      A0DE01A1DFB237C8008B22285A289C06BF4BC89C0BD77576932A14B1FEB9CE6D7F8816D710F1B043C8E
      58DCE1B32EF4EC8FB67E10CD23B6D4CC653DD8CD83B5F4DB0B5B741D30125CF842EE13EB940650E1E34
      E4666935B178F2351553F0822C8B354C70E47350E74A08F16D4F39F8AA80C3F4E0083C4BA1F31F5F1D0
      4FD4CF835AEA688885E85509133FFE557A7892A0161AC01BBCC8A27CE37E8CB9C1916A0F62BCF1E82C3
      F9213275B10CA272BFABCA2713CEEAECD0007C9FB6B562AFA2231FF7FD2C1D20D8ED28C11A840FEE931
      FE7A0E3BB925D88A852C2EE9BF606AD4000FA27643155A6FECAD9D4BABA8DE8F8D767AEC7A770D007AD
      B0D9F76E521DE6EF8D3567A32047688E2E8130AAF3EB594A366F3C534E335A3E9EDA326E60394CA10A4
      4340CC78995742E48994002CEE1049870D14C23C9FF2E5899DD7E3A1516D2F6E70B3DE1D79987379296
      E99EBCCAC43DA9A475CA3FE756D4649934BADA6DFA8C8F8BB21136172798BDA13E247B2F27874AFE13C
      CCA31F53D01A94B9520C3CBCDD1B1EB9BBBD6B83C76F64FC5D7C1DCF33A
k1 = A23742A2C267D7425FDA94B93F93BBCC24791AC51CD8FD501A238D40812F4CBFC59AAC9520D758CF789C
     76300C69D2FF
k2 = 0A5CFC45865775D0CC10F89EFAD9FFD33A6C8A7AB868309D
context = "CONCATENATION TEST VECTOR 1"
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = 933c64d06f5f3fb02d74530bcb093072682fafdd64d42933
```

C.2.6 ECDH with NIST P-384, SIKEp610, and SHA3-384

The input test vectors for clause C.2.6 are the same as clause C.2.5, only the output changes.

```
key_material = 05c6e23af74e3a8e1b1aa9f599c517468b86ab2068591af8
```

C.2.7 ECDH with NIST P-521, SIKEp751 and SHA-512

```
LA = 3102030405060708090A0B0C0D0E0F100102030405060708090A0B0C0D0E0F10
PA1 = 00602F9D0CF9E526B29E22381C203C48A886C2B0673033366314F1FFBCBA240BA42F4EF38A76174635F
      91E6B4ED34275EB01C8467D05CA80315BF1A7BBD945F550A501B7C85F26F5D4B2D7355CF6B021176599
      43762B6D1DB5AB4F1DBC44CE7B2946EB6C7DE342962893FD387D1B73D7A8672D1F236961170B7EB3579
      953EE5CDC88CD2D
PA2 = E1A758EC0D418BFE86D8077B5BB169133C06C1F2A067D8B202D9D058FFC51F63FD26155A6577C74BA7F
      1A27E7BA51982517B923615DEB00BE408920A07831DF5978CFDDD0BF690A264353A4A16B666F90586D7
      F89A193CE09375D389C1379A7A528581C3ACB002CD2DC4F0FD672568FF9050BA8365C7FEFC5E6ED089B
      921DE6804091A0744DE3EB14D426A3F7DA215C50312617C1C2697243980D06056F2CCE88AE7AE73C734
      3C0B7104C9F2870A94FED744CF6E94630514B6CEAB0E64733BB6FA67B931E5D8206010475CBE8BC5872
      48D65D89D8CD9C8BBFA93E8B5F9EB9130773DED665D52ABBD91C4C8C255F73C0FC82501AE33330E9F30
      8DE7177CBF83E4E26E334D7CB09019E638147FC58ED372AF660F14C194BC80E9666325C98E0F8087727
      1D4A6BF514F603703D8A697874CD50A34D92F5AAEA84633CCF96801BD517BF425DEE4A32AAF06684052
      473EA14643C3D535440FB2240A988D09F297C5A388CB3DE60ED943F124034B90EFF611221F80F78EC12
      4956338A105F6636B063D7E48BFBD5D614310FB97D86F122E4AE6F9DDF4977A93ED7D0CE2A94E346A1A
      03D3219CF21907B85A5BCDC713F93A4406A22E03B1655A66E1F6741A2F953E6FE0868B2614BABEF1943
      BBBCB1B66D3E7017E533EA84F291240B56AB33EF1DC3F3DE99DBF9E8BE51A0076E462BCDD825EA96D7F
      63C99177C305C257B31461F4C23D43115F0220409E8880BBB2468586D03461E807BE824B693874911B2
      B52AF06FDBDC47F5A0159729641A7C950AB9E03F2DC045135
LB = 3202030405060708090A0B0C0D0E0F100202030405060708090A0B0C0D0E0F10
PB1 = 00685A48E86C79F0F0875F7BC18D25EB5FC8C0B07E5DA4F4370F3A9490340854334B1E1B87FA395464
      C60626124A4E70D0F785601D37C09870EBF176666877A2046D01BA52C56FC8776D9E8F5DB4F0CC27636
      D0B741BBE05400697942E80B739884A83BDE99E0F6716939E632BC8986FA18DCCD443A348B6C3E52249
      7955A4F3C302F676
PB2 = 66D24BC4630B2EE312F01B26EC1D3EC1F583795EC93B90FD9B5453E0BEDA2A70FB6181C9B9EA86A9866
      F1468E62CE853C6C65AA5D0E4535828B3A97E2D1D31DC4372E6A36DA7B0C4733574B91CCE215086C59B
      54F2364F61298004C8410B17658B4021CD36859C94210DE338869CACF0E2DC11412FA5172E1663AAEBE
      F4B5EB0ED9175D6C86C5107DA92B8772342A2F44C93EFFE61F6C76AB8ABA194E862543EDB707E9D2EE8
      84995B1062FE2F60627D5C7673C7AC0D15B08C2F8510DC239463B1B32AD46873F6D1CB5A8579457386F
      D75700989BEED2CA547FE505C581B6B436AABC0F75AD6373A08CEC1504258A972C64EC4A1FEB86BFE32
      ACF3A73ECF815CBC883F39B42C6429A5875BD0BD6A94CEAD587AF49AC8EFB43E1A447D2D8555CB0ADFB
      C9F335F1C599BC9FEAB3E4FE5F2D06D930A58C2FFEEDE0E2726EBD85EC890D1CB0E6870DD784AE30286
      F1A336D57FC41D2F2E2F89765C6A110853BB63E478A64D54A31A18FB4BA44FF58A3662F4D82D544BD9B
      0E94FC88ABC4E4D27D5F6084B5F2162B357A04A1A28C8938834ECC987E50C0A2CACA442850493CE16C0
      47DF677097D3F7EA034BC3D2535504276003DBBEB12F1949C3D369E7EFBA09831E83D622AB2D9277F52
      3946FBAB1DDE14015857EA47663C5CCF30BDF261CCBF31DBE2A560E96CE87FBA80B783350A42C837EB3
      6B2F39A9FED1B649B8ECCE3D3235825F7C800834740546E0CF42C9C2C8B12495225F991B14547E5EEDB
      22858B26EE6E0AE13DBE3D50C6C1EF79C4B97DAD1B0239C4037C1AAC29EE1505E0E527EC81348900E7C
      216A1A1B34B8D2753AF2693647C412
k1 = 005FC70477C3E63BC3954BD0DF3EA0D1F41EE21746ED95FC5E1FDF90930D5E136672D72CC770742D1711
      C3C3A4C334A0AD9759436A4D3C5BF6E74B9578FAC148C831
k2 = FEE94595E8A05C50113C044D4D8558DA101035EBBF604AA41D0AAA75B8A7F786
context = "CONCATENATION TEST VECTOR 1"
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = 4DB991EEDA686DDA8CB5F128C16267CC7516643FE5EBD8C6176405C9AB4600FA
```

C.2.8 ECDH with NIST P-384, SIKEp751, and SHA3-512

The input test vectors for clause C.2.8 are the same as clause C.2.7, only the output changes.

```
key_material = 270EF22EEDE80D0100872E307388027E8E12AE331AB81A27D6D166F7241B7BAF
```

C.2.9 ECDH with NIST P-256, Kyber512, and SHA-256

```
LA = 101020304050607080090A0B0C0D0E0F1
PA1 = EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA70
      02523ACC85A429CB06EE6648325389F59EDFCE1405141
PA2 = 115ACE0E64677CBB7DCFC93C16D3A305F67615A488D711AA56698C5663AB7AC9CE66D547C0595F98A43
      F4650BBE08C364D976789117D34F6AE51AC063CB55C6CA32558227DFEF807D19C30DE414424097F6AA2
      36A1053B4A07A76BE372A5C6B6002791EBE0AFDAF54E1CA237FF545BA68343E745C04AD1639DBC59034
      6B6B9569B56DBBFE53151913066E5C85527DC9468110A136A411497C227DCB8C9B25570B7A0E42AADA6
      709F23208F5D496EBAB7843F6483BF0C0C73A40296EC2C6440001394C99CA173D5C775B7F415D02A5A2
      6A07407918587C41169F2B7178755ACC27FC8B19C4C4B3FCD41053F2C74C8A10A8321241B2802432875
      AE808B9EF1365C7B8A52902F1317BA2FB0269F47930672107B4726FEF64547394D3320C8F120B3C2F47
      25B0305FAB88CC7981FCB09A76A1CBF7F179F43BB0A4C8B0590857F1E69708466C7F8607391E7BC5268
      BFD3D7A1DFFCB4ECA2A1C9B597593013D5FC4202EC2B74E57AB76BBCF3632BBAF97CDC418A6F1639283
      8CA9BF45DDF023777B7561833C105190F94F302C59B531900BBC816361FAA5B3380CA3A893104CA7388
      B185671B3E5FE3790E9A626EC46D9B0B33C7A419AF7B32B6859894F575D82AC5456B5490A7AF8FE6104
      6360589ECBA7244236F4123116B6174AA179249A49195B356C72FC6641F0251812EAA98570B04669907
      0E0819DC2713F469137DFC6A3D7B92B298995EE780369153AC366B06D7249CD09E1B3378FB04399CECB
      8650581D637C79AE67D6F2CAF6ABACF598159A7792CB3C971D1499D2373AD20F63F03BB59ED137384AC
      61A7155143B8CA4932612EC915E4CA346A9BCE5DD60417C6B2A89B1CC435643F875BDC5A7E5B3481CF9
      19EA09172FEBC46D4FC3FB0CB9591704EE2DBB61844B2F3314A06BB6C6D34005E485CE667BDC7D09858
      6928D2D91340F00419EA401351A240A0B041058BEFB0C2FD32645B7A2DF8F5CBFD873327C978D7B351A
      2808438837024C52B9C295CD713646FB5D6C0CCFB470734AC2B2BC8123C2C13DF6938E92455A862639
      FEB8A64B85163E32707E037B38D8AC3922B45187BB65EAFD465FC64A0C5F8F3F9003489415899D59A54
      3D8208C54A3166529B53922
LB = 102020304050607080090A0B0C0D0E0F1
PB1 = 700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DD
      B20BA5C51DCC5948D46FBF640DFE0441782CAB85FA4AC
PB2 = EDF24145E43B4F6DC6BF8332F54E02CAB02DBF3B5605DDC90A15C886AD3ED489462699E4ABED44350BC
      3757E2696FBFB2534412E8DD201F1E4540A3970B055FE3B0BEC3A71F9E115B3F9F39102065B1CCA8314
      DCC795E3C0E8FA98EE83CA6628457028A4D09E839E554862CF0B7BF56C5C0A829E8657947945FE9C225
      64FBAEBC1B3AF350D7955508A26D8A8EB547B8B1A2CF03CCA1AABCE6C3497783B6465BA0B6E7ACBA821
      195124AEF09E628382A1F914043BE7096E952CBC4FB4AFED13609046117C011FD741EE286C83771690F
      0AEB50DA0D71285A179B215C6036DEB780F4D16769F72DE16FDADAC73BEFA5BEF8943197F44C59589DC
      9F4973DE1450BA1D0C3290D6B1D683F294E759C954ABE8A7DA5B1054FD6D21329B8E73D3756AFDA0DCB
      1FC8B1582D1F90CF275A102ABC6AC699DF0C5870E50A1F989E4E6241B60AAA2ECF9E8E33E0FFCF40FE8
      31E8FDC2E83B52CA7AB6D93F146D29DCA53C7DA1DB4AC4F2DB39EA120D90FA60F4D437C6D00EF483BC9
      4A3175CDA163FC1C2828BE4DBD6430507B584BB5177E171B8DDA9A4293C3200295C803A865D6D2166F6
      6BA5401FB7A0E853168600A2948437E036E3BF19E12FD3F2A2B8B343F784248E8D685EB0AFDE6315338
      730E7A1001C27D8D2A76FA69D157BA1AC7AD56DA5A8C70FE4B5B8D786DC6FC0566BA8E1B8816334D32A
      3FB1CE7D4D5E4C332AF7B003D091741A3D5C965292255DFF8ED2BBF1F9116BE50C17B8E548748AD4B2E
      957BBD1953482A2E1718CEC66CD2C81F572D552B7187885E6B8943D6431413C59EBB7E036048490BE52
      89E95B20A89E8B159F61A9A9886E147568F4C9021F362F02688A1C8C3BB0D24086880E55B6EDB43F374
      5D2C166DC1CB743C76FE6BE523A893CC764D16435C37851252A81E2FFBA0F18971A3DEE37D4877CB928
      E36E5235037A6B2057897D518A5F0E348E3AB6D5B52DFC60757F3B41A4FEC7828F1DEEAF4587CCC8EAD
      F647F4D203B2FAA05A649B582340CB4CACE57A30711BE752FACF0227D0A80C4128442DDC544BE805B9C
      FE8FE9B1237C80F96787CD9281CCF270C1AFC0670D
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
k2 = 0A6925676F24B22C286F4C81A4224CEC506C9B257D480E02E3B49F44CAA3237F
context = "CONCATENATION TEST VECTOR 1"
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = 1ACFDC16EEC7C8A669EC6007A4CAAB1A
```

C.2.10 ECDH with NIST P-256, Kyber512, and SHA3-256

The input test vectors for clause C.2.10 are the same as clause C.2.9, only the output changes.

```
key_material = 9B8DD1A341ADFB821E32980DA418A114
```

C.2.11 ECDH with NIST P-384, Kyber768 and SHA-384

```
LA = 2110203040506070809 0A0B0C0D0E0F10010203040506070
PA1 = FCFCEA085E8CF74D0DCED1620BA8423694F903A219BBF901B0B59D6AC81BAAD316A242BA32BDE85CB24
      8119B852FAB66972E3C68C7AB402C5836F2A16ED451A33120A7750A6039F3FF15388EE622B7065F7122
      BF6D51AEFBC29B37B03404581B
PA2 = A72C2D9C843EE9F8313ECC7F86D6294D59159D9A879A542E260922ADF999051CC45200C9FFDB60449C4
      9465979272367C083A7D6267A3ED7A7FD47957C219327F7CA73A4007E1627F00B11CC80573C15AEE664
      0FB8562DFA6B240CA0AD351AC4AC155B96C14C8AB13DD262CDFD51C4BB5572FD616553D17BDD430ACBE
      A3E95F0B698D66990AB51E5D03783A8B3D278A5720454CF9695CFDCA08485BA099C51CD92A7EA7587C1
      D15C28E609A81852601B0604010679AA482D51261EC36E36B8719676217FD74C54786488F4B4969C05A
      8BA27CA3A77CCE73B965923CA554E422B9B61F4754641608AC16C9B8587A32C1C5DD788F88B36B717A4
      6965635DEB67F45B129B99070909C93EB80B42C2B3F3F70343A7CF37E8520E7BCFC416ACA4F18C79812
      62BA2BFC756AE03278F0EC66DC2057696824BA6769865A601D7148EF6F54E5AF5686AA2906F994CE38A
      5E0B938F239007003022C03392DF3401B1E4A3A7EBC6161449F73374C8B0140369343D9295FDF511845
      C4A46EBAAB6CA5492F6800B98C0CC803653A4B1D6E6AAED1932BACC5FEFAA818BA502859BA5494C5F54
      02C8536A9C4C1888150617F80098F6B2A99C39BC5DC7CF3B5900A21329AB59053ABAA64ED163E859A8B
      3B3CA3359B750CCC3E710C7AC43C8191CB5D68870C06391C0CB8AEC72B897AC6BE7FBAACC676ED66314
      C83630E89448C88A1DF04ACEB23ABF2E409EF333C622289C18A2134E650C45257E47475FA33AA537A5A
      8F7680214716C50D470E3284963CA64F54677AEC54B5272162BF52BC8142E1D4183FC017454A6B5A496
      8317590640247459 78CBD51A6CEDC8955DE4CC6D363670A47466E82BE5C23603A17BF22ACDB7CC984AF
      08C87E14E27753CF587A8EC3447E62C649E887A67C36C9CE98721B697213275646B194F36758673A8ED
      11284455AFC7A8529F69C97A3C2D7B8C636C0BA55614B768E624E712930F776169B01715725351BC74B
      47395ED52B25A1313C95164814C34C979CBDFAB85954662CAB485E75087A98CC74BB82CA2D1B5BF2803
      238480638C40E90B43C7460E7AA917F010151FAB1169987B372ABB59271F7006C24E60236B84B9DDD60
      0623704254617FB498D89E58B0368BCB2103E79353EB587860C1422E476162E425BC2381DB82C659273
      7E1DD602864B0167A71EC1F223305C02FE25052AF2B3B5A55A0D7A2022D9A798DC0C5874A98702AAF40
      54C5D80338A5248B5B7BD09C53B5E2A084B047D277A861B1A73BB51488DE04EF573C85230A0470B7317
      5C9FA50594F66A5F50B4150054C93B68186F8B5CBC49316C8548A642B2B36A1D454C7489AC33B2D2CE6
      668096782A2C1E0866D21A65E16B585E7AF8618BDF3184C1986878508917277B93E10706B1614972B2A
      94C7310FE9C708C231A1A8AC8D9314A529A97F469BF64962D820648443099A076D55D4CEA824A583048
      44F99497C10A25148618A315D72CA857D1B04D575B94F85C01D19BEF211BF0AA3362E7041FD16596D80
      8E867B44C4C00D1CDA3418967717F147D0EB21B42AAEE74AC35D0B92414B958531AADF463EC6305AE5E
      CAF79174002F26DDECC813BF32672E8529D95A4E730A7AB4A3E8F8A8AF979A665EAFD465FC64A0C5F8F
      3F9003489415899D59A543D8208C54A3166529B53922
LB = 2120203040506070809 0A0B0C0D0E0F10020203040506070
PB1 = 1AEFBFA2C6C8C855A1A216774550B79A24CDA37607BB1F7CC906650EE4B3816D68F6A9C75DA6E4242CE
       BFB6652F65180419D28B723EBADB7658FCEBB9AD9B7ADEA674F1DA3DC6B6397B55DA0F61A3
PB2 = B52C56B92A4B7CE9E4CB7C5B1B163167A8A1675B2FDEF84A5B67CA15DB694C9F11BD027C30AE22EC92
      1A1D911599AF0585E48D20DA70DF9F39E32EF95D4C8F44BFEFDAA5DA64F1054631D04D6D3CFD0A540DD
      7BA3886E4B5F13E878788604C95C096EAB3919F427521419A946C26CC041475D7124CDC01D0373E5B09
      C7A70603CFDB4FB3405023F2264DC3F983C4FC02A2D1B268F2208A1F6E2A6209BFF12F6F465F0B069C3
      A7F84F606D8A94064003D6EC114C8E808D3053884C1D5A142FBF20112EB360FDA3F0F28B172AE50F5E7
      D83801FB3F0064B687187074BD7FE30EDDAA334CF8FC04FA8CED899CEADE4B4F28B68372BAF98FF482A
      415B731155B75CEB976BE0EA0285BA01A27F1857A8FB377A3AE0C23B2AA9A079BFABFF0D5B2F1CD9B71
      8BEA03C42F343A39B4F142D01AD8ACBB50E38853CF9A50C8B44C3CF671A4A9043B26DDBB24959AD6715
      C08521855C79A23B9C3D6471749C40725BDD5C2776D43AED20204BAA141EFB3304917474B7F9F7A4B08
      B1A93DAED98C67495359D37D67F7438BEE5E43585634B26C6B3810D7CDCBC0F6EB877A6087E68ACB848
      0D3A8CF6900447E49B417F15A53B607A0E216B855970D37406870B4568722DA77A4084703816784E2F1
      6BED18996532C5D8B7F5D214464E5F3F6E905867B0CE119E252A66713253544685D208E1723908A0CE9
      7834652E08AE7BDC881A131B73C71E84D20D68FDEFF4F5D70CD1AF57B78E3491A9865942321800A203C
      05ED1FEEB5A28E584E19F6535E7F84E4A24F84A72DCAF5648B4A4235DD664464482F03176E888C28BFC
      6C1CB238CFFA35A321E71791D9EA8ED0878C61121BF8D2A4AB2C1A5E120BC40ABB1892D1715090A0EE4
      8252CA297A99AA0E510CF26B1ADD06CA543E1C5D6BDCD3B9C585C8538045DB5C252EC3C8C3C954D9BE5
      907094A894E60EAB43538CFEE82E8FFC0791B0D0F43AC1627830A61D56DAD96C62958B0DE780B78BD47
      A604550DAB83FFF227C324049471F35248CFB849B25724FF704D5277AA352D550958BE3B237DFF473EC
      2ADBAEA48CA2658AEFCC77BBD4264AB374D70EAE5B964416CE8226A7E3255A0F8D7E2ADCA062BCD6D78
      D60D1B32E11405BE54B66EF0FDDD567702A3BCCFEDE3C584701269ED14809F06F8968356BB9267FE86E
      514252E88BB5C30A7ECB3D0E621021EE0FBF7871B09342BF84F55C97EAF86C48189C7FF4DF389F077E2
      806E5FA73B3E9458A16C7E275F4F602275580EB7B7135FB537FA0CD95D6EA58C108CD8943D70C164311
      1F4F01CA8A8276A902666ED81B78D168B006F16AAA3D8E4CE4F4D0FB0997E41AEFFB5B3DAA838732F35
      7349447F387776C793C0479DE9E99498CC356FDB0075A703F23C55D47B550EC89B02ADE89329086A508
      43456FEDC3788AC8D97233C54560467EE1D0F024B18428F0D73B30E19F5C63B9ABF11415BEA4D017013
      0BAABD33C05E6524E5FB5581B22B0433342248266D0F1053B245CC2462DC44D34965102482A8ED9E4E9
      64D5683E5D45D0C8269
k1 = 3D2E640F350805EED1FF43B40A72B2ABED0A518BCEBE8F2D15B111B6773223DA3C3489121DB173D414B5
      BD5AD7153435
k2 = 914CB67FE5C38E73BF74181C0AC50428DEDF7750A98058F7D536708774535B29
context = "CONCATENATION TEST VECTOR 1"
```

```
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = 1529723880E3E1777F73F8D89724D323572E56858D3DD732
```

C.2.12 ECDH with NIST P-384, Kyber768, and SHA3-384

The input test vectors for clause C.2.12 are the same as clause C.2.11, only the output changes.

```
key_material = 7A954FCF3DE3C845B65E1D6F3CCD6186BD2F10FF83525B7A
```

C.2.13 ECDH with NIST P-512, Kyber1024 and SHA-512

```
LA = 331020304050607080090A0B0C0D0E0F100102030405060708090A0A0B0C0D0E0F1
PA1 = 00D45615ED5D37FDE699610A62CD43BA76BEDD8F85ED31005FE00D6450FBBD101291ABD96D4945A8B57
      BC73B3FE9F4671105309EC9B6879D0551D930DAC8BA45D25501425332844E592B440C0027972AD15264
      31C06732DF19CD46A242172D4DD67C2C8C99DFC22E49949A56CF90C6473635CE82F25B33682FB19BC33
      BD910ED8CE3A7FA
PA2 = D22302CBD3399FACC630991FC8F28BDB4354762541527678BCF61F65C241146C426D23B9BFAA6B7DF1
      8C97F20C1B6125BF874B1D89475852C448215DB0EB7737F91480E8CEBD9A0871574F5AB62D9020175EC
      6927CA0B54C09818E42CF92A383172422C7DC1831D63B0C295DE75159DB8034E9E07F7B0B910C3C1E5F
      B66B3DC523F1FA6EB4910CB89A6C17562C83AB4C18D0CD7E0796592A372AA409B1C557347CCACDC4644
      A119064D06DD474929D1C6FB4D686E5491CE4BC89A30BB4B8C41BCE5157DFC1360823B1AB618C14B10F
      98C25067398EA7018C278A4B3DF31334D603B2044EF187CD9BC6CE42725BD962C264983E9E18155A8B9
      C47143D70460A26A56FE7658C1F150348C6087EF758AD167887860A007A5FC37358D43B5EBEE820ACEA
      474F0AC07B76802866199C61231D5C747C93774D2C1E0C1C67E6C81B82752173E125BAF39B4FD19A4F4
      53DC57976B1D97FE6996992BBB65B7CB25D077BBAA6A13322899AF659CF1B3558C1B5001154B625809E
      D89AEEBB89E6EA7D67F723D045AB05715C42355DA6A5C8DD39C8ABE3037751A01ED1C7374919F3121B5
      A52C53D1487316769F80721DEEAAAD3C90F76E7AE9E12BA92B32B5FD457E3C752C2650DFB885771CB77
      AC3C785A8C562E6A1C63C2A55EA47CF8B90EB8225C123C346452566235B2F31823A33521E087937A345
      D8D663EEAA05658917BBAA008C2E335F8850A90A326D0E66432F44CEB8289E4ECB2D12958E984072ECA
      CB88E1348FF0B55654ACBA5B54971CBAEBA88EC4B91A94C37192FA982BECB9F3DA421603B61A51BC8E3
      6CBD053851C77B1B926B17A272AA9023246B02B3ED47F66A00BD5684823634E7CE58CF8F306E35B1E53
      22824D904801F0A2FA7C2BC9C252B0A56B7BA2AB0F636021745A70A9A43E2B0A8D615970B65309624B5
      184BCC30B911679AEDD76025FE3908FD67897B0CF4BE5A6F5413D7DD98564B23E42A93E4AA8821CD450
      54C643EDC1158DB6B3DEB13FB5A51EBD1A8A78B87225A7338E101104C4A220D9BDEDD48C85A1C2DAE78
      1A80C40E13B87EAC73A764201C9B760CCFB1AE392699C7039D27C39362B27B8FC6F07A8A3D4410F1547
      C48A9997F62C61074452EF1515F8A649EBCA9437205A4E8A61606B41DAF6834D671F4D852C0C9C40966
      11648C6A3170678B1537CC1828D93580C9E5849A9653175ACB753F2BE7437BE45F6C603E485F2EC301B
      B42B6C37C225D7495A584AE231890AB5C8C35C268CF4BBB0213C096019319561A8A6947637AA40D006B
      415BB2CFA2237E0890B6A3BC134ABF8F6585E108D15940F91F4BF5B0C818055B21DEA6E63B553988C47
      F4B94E7CF800A493B4734705EDC56A4B6021C629500675876804CF0B951F038A5C7FE58E89774EF2992
      FD7C63099D352A7D21560B788B405709861817E59A96B3A3A83CBA803B16934331071905BBEC6532900
      155D8AC88CB32E4E21A3BD3A03FDEC325A51CD2773964E6784FCF1853737AA64EB67564727272661ABF
      84313A57A44B123C65509CFB7A6F6641CDCC3B57FE628C7B8192DB44FFBF5796A8613B1FA126F607688
      3C783DC24E2A4464C40B3A41CA70AE87620866CF4FCB2BD204BF5C283812BA056AC0C345E379C4BA24D
      750901279BB2F3A16F612BFADB35703332C7C136F68EAB6755C66B6A4AD1AABA7B768A58ACAACC10A45
      9A1CC8EF29377BC200E4D315A30A6BCC3256F9734D06E9779CAA5442A9A16069081377C76E751543680
      72DC446ED6C8B8E622A21E383CF9BA1FB434E2ECC81E7B78CEE986B8FF798AB18CF9634543546284EDA
      2A26B47F05B735BCDB1202220076DC8B4E4B9F853533C8F6C7FF38817BA49712835785F17F14CA01D0C
      1C1E98810FE0B36E5B427157B9418449CEDD641A4293C85C32700102ACEC22EBAD98ED160A5F027BD4C
      DA57F1F3720A12C134654DD5E73F829676495390D0E7929D6034E9C55F7D55BA658BC587988E8AF9496
      0F6CFB8D5AF7A0021535A6E25E437D49A780698BE22AC9953949F571B85A685725F8207A2B0AE849B60
      1AB91B159B3DF4A154C2041E776070AFC42969322380917C97510799F3149131477E16663D3174C7C1C
      AEA788535C6C005A64F2868631B31B66E205FD38C1D84542D0F1B578F58C9BF5A0FAEAB6AB649489305
      3165EAFD465FC64A0C5F8F3F90034894158 99D59A543D8208C54A3166529B53922
LB = 33202030405060708090A0B0C0D0E0F10020203040506070 8090A0B0C0D0E0F1
PB1 = 01DF277C152108349BC34D539EE0CF06B24F5D3500677B4445453CCC21409453AAFB8A72A0BE9EBE54D
      12270AA51B3AB7F316AA5E74A951C5E53F74CD95FC29AEE7A013D52F33A9F3C14384D1587FA8ABE7AED
      74BC33749AD9C570B471776422C7D4505D9B0A96B3BFAC041E4C6A6990AE7F700E5B4A6640229112DEA
      FA0CD8BB0D089B0
```

```
PB2 = A6AF29D5F5B80BD130F518BADDD6C8F17545413D860FB3DE451979EBFA5E4E3112C7C0ADF99824BB526
      F2C3550748ED0E134F0457A7C61F9F526F002BAADC03FC13E38131219513C3EDE061661E74F603C4FCF
      7951C8E52C9C213B0D22D9293663D669A6B58ED8FCEFCF8249D7BB5298F55761445B2B83CE7F005CB04
      248AEC8BDA22FD2D42AA766322014EA038CC32C55C8E4B9E28EC9119F527341E4F66A035121073B85DE
      6706DA19E0838A9F33B719A68F039B664DC002659EABFC398679AA7009CE0CD01CDAFB6CD2A26FE4101
      672C98FF58F7C47D5BDA2906653B3A6F9651F7A121EA77EA74723FAE5B873F9BB7B664F0C8A93831EF9
      D51C7CC1EF44AC0E55A55CA76D137FE9B75F40509CEF156E5AD18F9FB999680008E547D55EECD5B4D1C
      B1D9F076CEC21501C7402509ECB77AFB2CB9A61340A8BD1514C6E71B4AA45E47EC37512271B911F8FB4
      6C9082C9DF07204ABB5A50E6E3647A8AD4D8D5D7BFF19C8A509308BCFB895536D045CA2B97CB16A29BB
      7181CAD0509DDB91735028EBA8C31D74BD275EAA65B5340B3A43FBFE0B3061D6BAE7E75B7098CDABE91
      D4B31E36C9AA7A8298862AD63C8FD282E03B460B3AB464CE0F27B1C3D11155ACAA011EB9E2AE3E6DDA0
      7D6F491737CBCE9B05F9BC56BE20E8D326BA132C57FB235161144519CDF40560FBE279BDE411E112531
      F826D6AB10D4547350ADD2A9DE8D62C2AC82CABE6815646F4DC9742BB0C2A3F77EC7B46C6B537605FA3
      1798CD89281221A33DFB9796E644305630332C2CB931408AB481A16D953F6BEAE3891D6D9AC1FAB3822
      2D9271872D9D0CADB91ABE9B4E265F75C6E5E829E146C3D8CE1E9D12E0D129801957F46B0D2DBE1F749
      B1D08E2345F6239A731342EB75B0CF1BF411749BC2CAF2810B788C6B7238B4D3DA2D6315CE9542E2440
      4F145755A30AB851E4445841BD33F716A586884888ECC6BC6498AA32919AE81D20C26973C2BD54582A0
      F6AD98ABFD2627E15690A727E69F581DD2A7127982A90E33E2D4A03FE339142C7E44C326AC46ED395A2
      25D3033389917328B45316B1585A01B2C304B2944E903ABBB3EC5619441CFC8965A446DF75DEFA80C6E
      15ADBD506B7AB2DE12DDA9BC81441CFC89052E2E5808F7126C6FD3AC6AC8081258A84A09AE50F6CD7CC
      0F4AF336FD1D643E99079996268C2D32D909F22E3504F07FBB563196D4312FDDB9335D5C1D36E8C5EEA
      2278DBA23B94D193C947CC41CA993DC7DB1396340AD9C4FE687DD7B8D0C7A5120AE0204F2C665BD5F47
      3D644C7FF26BFFBA7A36980830702128A7E661D677A092A36E7428A4139FB29B0095CC11086F447D2A9
      EF6C9B161F189C6299E084CB7AA00FAF787797BFB069FBC087FDE26252A1664F19C5A8A22EC5EE1AEB0
      76357B7DC37E6B0F1520F958F7851BACB92C89FD114A72FEAC54652D45B09E1AE7651ABD164BCD537D5
      8FA39D3EC8ACDCDF98425005862FA59692DE162B77E6297C66233348408A8AB695CE2F2728DB9FBE27E
      958967EC5974767C5A66023074B4A71AFD264AD2890E970A1F31D6E3311B736F9F9488793DDC88F2345
      8064254C82A1D9E59EAD2FCEC40B430687C4B7E28960926AFCACC9BD756A71088C78450E20A2E980AED
      E9EBEDFE7FABD6ABFE96F934C4B02C01CA194D01B73C25D5997039D3FCD0F099521F70CAEE69110AC1F
      C5A99917AD752FC96ADFAD7186D0A7C9CFE5601C07514EA6448D661C57AA20242103C4276A070A489A4
      CB6BCA0F9ECC4379FB220215FD91F81019D5B0AE619358B52468F272C178E3A74CF6775AA924FE329C3
      175D9E4C3E21AB9EC836EDC3ACAB2E3891EE8DEDA515D39AF9B8DDD0EE7B0164F805C3835F6D2BABDB3
      0EAB4756E7EC7F829ECE01E8EADFBBED12FC283B3D4C69F575E7F80417689FDFCFC7BE27EE3B8CDF57A
      AEBEC4A95B7E5BB585B85227F7C32BE30DB3E65E42E30DCF5A5FA073DBA399D942F2222ADB9B9898102
      AFE5432EDC7F04AE34A8FEC2D81CB49A9A9B43814CE71D97F726E2B1E8F64B50E65DFB4816E12E82A31
      97484A4E9BBA4D2D69E3F19D0B75C21E2BFFE9FC0C98CF48A3AAF08D467F72687DF0178174B7897F734
      349B181ECA86A598A0C5E8C25946F24DC5572BD324A40458A788E5137F3C7A7C97FC9F12A3C463A8FE9
      449101CCE966D7C009323932998D56EF430C73BC24F5D95F737858DDC4F32C013
k1 = 000B3920AC830ADE812C8F96805DA2236E002ACBBF13596A9AB254D44D0E91B6255EBF1229F366FB5A05
      C5884EF46032C26D42189273CA4EFA4C3DB6BD12A6853759
k2 = B10F7394926AD3B49C5D62D5AEB531D5757538BCC0DA9E550D438F1B61BD7419
context = "CONCATENATION TEST VECTOR 1"
label = LA || LB
MA = LA || PA1 || PA2
MB = LB || PB1 || PB2

key_material = 6A4D002A9B310D6E73D87BF36224E2F41EDD8F75CFF448D59493D65D61C07D8B
```

C.2.14 ECDH with NIST P-512, Kyber1024, and SHA3-512

The input test vectors for clause C.2.14 are the same as clause C.2.13, only the output changes.

```
key_material = 203F634CFF7D7EC462281A65706E50342F025B7145E03A976640137C89F919A1
```

C.3 Test vectors for CasKDF

C.3.1 ECDH with NIST P-256, SIKEp434, and SHA-256

```
LA1 = 0102030405060708090A0B0C0D0E0F10
PA1 = 119F2F047902782AB0C9E27A54AFF5EB9B964829CA99C06B02DDBA95B0A3F6D08F52B726664CAC366FC
      98AC7A012B2682CBD962E5ACB544671D41B9445704D1D
LB1 = 0202030405060708090A0B0C0D0E0F10
PB1 = 809F04289C64348C01515EB03D5CE7AC1A8CB9498F5CAA50197E58D43A86A7AEB29D84E811197F25EBA
      8F5194092CB6FF440E26D4421011372461F579271CDA3
previous_chain_secret = psk = <NULL>
k1 = 057D636096CB80B67A8C038C890E887D1ADFA4195E9B3CE241C8A778C59CDA67
context1 = "CASCADE TEST VECTOR 1"
label1 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LB1 || PB1

chain secret1 = FFA4725EB8201C5BE4969A5DB5FDE22854DA4E935929DABBD91F5B7D9C01897D
key_material1 = E2753CA0E90F966D4B67A3125A86F854

LA2 = 4102030405060708090A0B0C0D0E0F11
PA2 = 4484D7AADB44B40CC180DC568B2C142A60E6E2863F5988614A6215254B2F5F6F79B48F329AD1A2DED2
      0B7ABAB10F7DBF59C3E20B59A700093060D2A44ACDC0083A53CF0808E0B3A827C45176BEE0DC6EC7CC1
      6461E38461C12451BB95191407C1E942BB50D4C7B25A49C644B630159E6C403653838E689FBF4A7ADEA
      693ED0657BA4A724786AF7953F7BA6E15F9BBF9F5007FB711569E72ACAB05D3463A458536CAB647F00C
      205D27D5311B2A5113D4B26548000DB237515931A040804E769361F94FF0167C78353D2630A1E6F595A
      1F80E87F6A5BCD679D7A64C5006F6191D4ADEFA1EA67F6388B7017D453F4FE2DFE80CCC709000B52175
      BFC3ADE52ECCB0CEBE1654F89D39131C357EACB61E5F13C80AB0165B7714D6BE6DF65F8DE73FF47B7F3
      304639F0903653ECCFA252F6E2104C4ABAD3C33AF24FD0E56F58DB92CC66859766035419AB2DF600
LB2 = 4202030405060708090A0B0C0D0E0F11
PB2 = 0FDEB26DBD96E0CD272283CA5BDD1435BC9A7F9AB7FC24F83CA926DEED038AE4E47F39F9886E0BD7EEB
      EAACD12AB435CC92AA3383B2C01E6B9E02BC3BEF9C6C2719014562A96A0F3E784E3FA44E5C62ED8CEA9
      E1108B6FECD5BF8836BF2DAE9FEB1863C4C8B3429220E2797F601FB4B8EBAFDD4F17355508D259CA607
      21D167F6E5480B5133E824F76D3240E97F31325DBB9A53E9A3EEE2E0712734825615A027857E2000D4D
      00E11988499A738452C93DA895BFA0E10294895CCF25E3C261CBE38F5D7E19ABE4E322094CB8DEC5BF7
      484902BABDE33CC69595F6013B20AABA9698C1DEA2BC6F65D57519294E6FEEA3B549599D480948374D2
      D21B643573C276E1A5B0745301F648D7982AB46A3065639960182BF365819EFC0D4E61E87D2820DBC0E
      849E99E875B21501D1CA7588A1D458CD70C7DF793D4993B9B1679886CAE8013A8DD854F010A100C9933
      FA642DC0AEA9985786ED36B98D3
previous_chain_secret = FFA4725EB8201C5BE4969A5DB5FDE22854DA4E935929DABBD91F5B7D9C01897D
k2 = 35F7F8FF388714DEDC41F139078CEDC9
context2 = "CASCADE TEST VECTOR 1"
label2 = LA2 || LB2
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain secret2 = C20B9AD0C09EFB17877D9FD3FB822FB6FFFCC83C78861A73EC07738591CD5EB0
key_material2 = C1EF63BE6F26F6EBE2078C940AC47E59
```

C.3.2 ECDH with NIST P-256, SIKEp434, and SHA3-256

The input test vectors for clause C.3.2 are the same as clause C.3.1, only the output changes.

```
chain secret1 = 266E4536DFD94CF7CEEC48D7D907FDF5C3FD95189646E219708FF322C474CA2C
key_material1 = BA453A0C95863A4510C65FA4200660C9

previous_chain_secret = 266E4536DFD94CF7CEEC48D7D907FDF5C3FD95189646E219708FF322C474CA2C
chain secret2 = A2775CE2471C75BC46384CD0F5E85F1A49CFE30D24F9338B67FBF4C7FB33C3F8
key_material2 = 5C0D6EE9B187EEE46C22EB3C3F36349A
```

C.3.3 ECDH with NIST P-384, SIKEp503 and SHA-384

```
LA1 = 1102030405060708090A0B0C0D0E0F100102030405060708
PA1 = 9803807F2F6D2FD966CDD0290BD410C0190352FBEC7FF6247DE1302DF86F25D34FE4A97BEF60CFF5483
      55C015DBB3E5FBA26CA69EC2F5B5D9DAD20CC9DA711383A9DBE34EA3FA5A2AF75B46502629AD54DD8B7
      D73A8ABB06A3A3BE47D650CC99
LB1 = 1202030405060708090A0B0C0D0E0F100202030405060708
PB1 = A7C76B970C3B5FE8B05D2838AE04AB47697B9EAF52E764592EFDA27FE7513272734466B400091ADBF2D
      68C58E0C50066AC68F19F2E1CB879AED43A9969B91A0839C4C38A49749B661EFEDF243451915ED0905A
      32B060992B468C64766FC8437A
previous_chain_secret = psk = <NULL>
k1 = 5F9D29DC5E31A163060356213669C8CE132E22F57C9A04F40BA7FCEAD493B457E5621E766C40A2E3D4D6
      A04B25E533F1
context1 = "CASCADE TEST VECTOR 1"
label1 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LB1 || PB1

chain secret1 = 53C3DF544C90AABC3F658DE47B107EAD6DE480A5884488A19E24992CAC73B104DAC5C55C8
                0772CC5176DF0559E0D325B
key_material1 = D36B8460812CE2D06D0A30108855AB0B866C4B59E77B34CC

LA2 = 5102030405060708090A0B0C0D0E0F110102030405060708
PA2 = 05279D27FF7E3A38ABB05DCFE23B5831C030D832D3EAE35FE06A6538597532D22A0F4012FB2263E1604
      95F8291B58D9DF8A8947C7CF3E6735520BB2D094912408829851AC4B85AA922069F2AAA0A4827DFA473
      0E9CF05485CBEE411C3D5169DD4953746B6A2E6574957EF920596B1612BE62A883740B5A0C157117AE1
      C3A07E4CE8CCCE7E9E88FE7C20A507FF019AE0893F34303E173D291F6CB7ECB4C3901FF34A48DE40771
      F5BAD72DA2B4C1CFD0A61F33E39327A8DA60F5640D4C2E71EF9C7297A4E9BC50493E3BA65D3664610A6
      D61035CB6600378D017D1E1810ACD113252D60F5915749C2B5CFB4452C40C86F1F40C63297DCCA90068
      6F2D2266F9444539D9BA13B1F52FB2FC3BD4F3EDAA6EB707AAFCA5261EA271ED961B2ED195D5E3B0299
      179251866CE0EAA31C5C90B7999A8D63BA2DE84A8AFA19F11F2DC0CACA39B982CE053F71D269931D9EE
      26BCE592A8EA818553BC8F8D244F62FB4F5E5386E3EFF5CD231401C9EC2BA57FF42DC3B3791357A53E1
      E31394008
LB2 = 5202030405060708090A0B0C0D0E0F110202030405060708
PB2 = 100692A8BD30F01BE8AC6B1AF8D93A060D3821B2587F4038D64B72426A194BEDE63CA60B75A5C3C1553
      2CE307115AA9D77AC232E14D99C1E1AFEF1EB2D6321AE342F990023466E683A2568D59A14325C2C6C27
      2029741D8E36976D1804059BC06B802F3A495EA50D0DBBA93FD263F4CF30BDB5F783BA6A0775715B05F
      700C85B316F7AA1A1624973885941DBFF91316BF47AC698E11D6B2418F553379D67A00F784B8643FB8A
      94029584391D488775EB4414A5E6E8122B0F282D900F3D05775F1DD994FB232ED826106203CD3433967
      F60FF925DF9E86CB376CAB5FD90B132E425682741F6AF078E75792CB4CE085D44993CFB6A4ED5AA3541
      640A0A67687922B92382CAC47C6AD358011A269CC7C17CE651CA2E2393F7DFE19D7054FEF69610A353D
      676B1F076549510590D406AD13F4A3292CCF206DBDAE47F08D448CC006449F27C1FB54E9C9E6F16ED2F
      3D120DD5AA2620D76690F00E31904C601310C76A843A58E1AEB9C5F515FCEC482C08205FDE99A89E644
      85EBBD43EEFE2E24D18EEE8F20DF6E113C6667512E28396862C98F5C0
previous_chain_secret = 53C3DF544C90AABC3F658DE47B107EAD6DE480A5884488A19E24992CAC73B104D
                        AC5C55C80772CC5176DF0559E0D325B
k2 = AF1280151C2C59B4D4150B18BA7F71590523CEA83C9BDDDA
context2 = "CASCADE TEST VECTOR 1"
label2 = LA2 || LB2
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain_secret2 = A487F0BCEFDFEFB330DE2D7BAAE9532AE66F0B21C6192AF0E30DBB4B3262184F1D6C5BD94
                2960AF153ABB1CC903BB5F8
key_material2 = CA58B87E512FCAEF2B11B1A1A8550FB4B1C1E2ACF7C364D5
```

C.3.4 ECDH with NIST P-384, SIKEp503, and SHA-384

The input test vectors for clause C.3.4 are the same as clause C.3.3, only the output changes.

```
chain secret1 = 58E430C3AD320B250B39DAC3C699ABCC2EA2316B019BBA02F317E0F300ED3FD6C980181AA
                355FC98D48DDA8659633C5D
key_material1 = 9EFBA9191FA785508409915FE9CDEB502CB92B234EB9F067

previous_chain_secret = 58E430C3AD320B250B39DAC3C699ABCC2EA2316B019BBA02F317E0F300ED3FD6C
                        980181AA355FC98D48DDA8659633C5D
chain_secret2 = BE00F2C2459791A41DC9E4CF2B14F8B1748A715568A96179BF0EBEB9BCB8BA7FE593913BB
                290AA5DDC856896FFB86195
key_material2 = 007B8F4E6C95851C3598F59484D75A2A6FCFC931B42D74E4
```

C.3.5 ECDH with NIST P-384, SIKEp610 and SHA-384

```
LA1 = 210203040506070809 0A0B0C0D0E0F100102030405060709
PA1 = EA4018F5A307C379180BF6A62FD2CECEEBEEB7D4DF063A66FB838AA35243419791F7E2C9D4803C9319A
      A0EB03C416B6668835A91484F05EF028284DF6436FB88FFEBABCDD69AB0133E6735A1BCFB37203D10D3
      40A8328A7B68770CA75878A1A6
LB1 = 220203040506070809 0A0B0C0D0E0F100202030405060709
PB1 = 30F43FCF2B6B00DE53F624F1543090681839717D53C7C955D1D69EFAF0349B7363ACB447240101CBB3A
      F6641CE4B88E025E46C0C54F0162A77EFCC27B6EA792002AE2BA82714299C860857A68153AB62E525EC
      0530D81B5AA15897981E858757
previous_chain_secret = psk = <NULL>
k1 = A23742A2C267D7425FDA94B93F93BBCC24791AC51CD8FD501A238D40812F4CBFC59AAC9520D758CF789C
      76300C69D2FF
context1 = "CASCADE TEST VECTOR 1"
label1 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LB1 || PB1

chain secret1 = 77C7118115C49436C9AD0E9DD53E5CABE58859FC3E91A9E5DB4729719D330B3FA141DBA1A
                E424D5C5BF3CAB9C5E40000
key_material1 = 2B1B2DD39E5BB0FF0BC99B310113DFA76D3B28E4D167C117

LA2 = 610203040506070809 0A0B0C0D0E0F110102030405060709

PA2 = 671B24769304DD18C97AF0C5DE741C53E0B45A9E18C7A13A15C1758125E41605587E450F8452A2BF98B
      51C2AF6B0503CB8E01F8553C36079EBFADF4948FFA063ABF4866E7AB9B9D4C9A07CA400C613607E6DB9
      BB6E7EB8ECA78894C7C8CE9E231B33179B2946C5C5BE1C783FA6AEA218F5EC4B4E6F914E5ED3724C5D7
      B79403F68438A40775E964C1B2C7D22E11A6C07474EB5D4CFF75965B400167E069FA9908A562DBABF5E
      30FED46BBA0A208ED4E50764CF320FB8556F07C7F6268084476A47D83B085DC77EB3CD30A2B5EE1E582
      9738077D52A0D7A4149EE9C1A70269BC047B4BE7E5B28007DEF74A4D813853396708A3A8498CC862F54
      015B79047014639EB8CA3BB786B27A2CFAF31E6BB9CCB152BEB3232465206973668597AA35EE1940A31
      6F71241FA40D1AC233931E1967E79AAA600AA6D83FEC6280A63924E7375F22F7A47E1DE483FEA17E0DA
      CBAEDBB13D58C0DC9BC21F2DC9525D46E4210AC5D88567E4F23304EA5BE08D89D57A0246EA21C0CD28C
      096366D7F3C8D98F5A1FB00FE2F3A183E53A7E8B6C19E9BF979E8D20C703C957D6F06A142BE86A0A09B
      05ED40953BBD7A15E92098633941730DEB5BC1C5F5154E8BCA38E035580E101E6EE858D91BD8462B906
      EB2004C6E01
LB2 = 620203040506070809 0A0B0C0D0E0F110202030405060709
PB2 = FB75E7D835313132AC0B29D8732F1F62E6DD10BBF30375B4A50C7B153431BAE6259E1C5526C07164E87
      EDC70E4F0D8331D73285661D1F639D216372D05B4583C1302932B03FF184D115D0B250297FF26AE81DF
      A0DE01A1DFB237C8008B22285A289C06BF4BC89C0BD77576932A14B1FEB9CE6D7F8816D710F1B043C8E
      58DCE1B32EF4EC8FB67E10CD23B6D4CC653DD8CD83B5F4DB0B5B741D30125CF842EE13EB940650E1E34
      E4666935B178F2351553F0822C8B354C70E47350E74A08F16D4F39F8AA80C3F4E0083C4BA1F31F5F1D0
      4FD4CF835AEA688885E85509133FFE557A7892A0161AC01BBCC8A27CE37E8CB9C1916A0F62BCF1E82C3
      F9213275B10CA272BFABCA2713CEEAECD0007C9FB6B562AFA2231FF7FD2C1D20D8ED28C11A840FEE931
      FE7A0E3BB925D88A852C2EE9BF606AD4000FA27643155A6FECAD9D4BABA8DE8F8D767AEC7A770D007AD
      B0D9F76E521DE6EF8D3567A32047688E2E8130AAF3EB594A366F3C534E335A3E9EDA326E60394CA10A4
      4340CC78995742E48994002CEE1049870D14C23C9FF2E5899DD7E3A1516D2F6E70B3DE1D79987379296
      E99EBCCAC43DA9A475CA3FE756D4649934BADA6DFA8C8F8BB21136172798BDA13E247B2F27874AFE13C
      CCA31F53D01A94B9520C3CBCDD1B1EB9BBBD6B83C76F64FC5D7C1DCF33A
previous_chain_secret = 77C7118115C49436C9AD0E9DD53E5CABE58859FC3E91A9E5DB4729719D330B3F
                        A141DBA1AE424D5C5BF3CAB9C5E40000
```

```
k2 = 0A5CFC45865775D0CC10F89EFAD9FFD33A6C8A7AB868309D
context2 = "CASCADE TEST VECTOR 1"
label2 = LA1 || LB1
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain secret2 = E6A3396C61F4492C41C02FBF9287D749902EF7CD623AF8325EA0658D86E62A0BBF5DC4571
                AA3E0306998F450984000DE
key_material2 = DBC55D5F607CF981CB9774E920288ED030C3A13CADD4E594
```

C.3.6 ECDH with NIST P-384, SIKEp610, and SHA3-384

The input test vectors for clause C.3.6 are the same as clause C.3.5, only the output changes.

```
chain secret1 = 965DEECA645163B0F37DB839865C8D808C978C943C476155DAFE80797FD5D8C50E58B3614
                56EC952588E09801B2F42D3
key_material1 = AAF932F4C638325E3436F185ADD815C0751339005887DF45

previous_chain_secret = 965DEECA645163B0F37DB839865C8D808C978C943C476155DAFE80797FD5D8C50
                        E58B361456EC952588E09801B2F42D3
chain_secret2 = BB712F8E91C696A3D31C9CEE7BD976B3D4013D552044543FD2472D21AAF49FC264552CD36
                C37D6FAF2219E1977E62F33
key_material2 = 944DC1D2AD91B40E3B3B6D2B19981D8571E27A0C49EE4767
```

C.3.7 ECDH with NIST P-521, SIKEp751 and SHA-512

```
LA1 = 3102030405060708090A0B0C0D0E0F100102030405060708090A0B0C0D0E0F10
PA1 = 00602F9D0CF9E526B29E22381C203C48A886C2B0673033366314F1FFBCBA240BA42F4EF38A76174635F
      91E6B4ED34275EB01C8467D05CA80315BF1A7BBD945F550A501B7C85F26F5D4B2D7355CF6B021176599
      43762B6D1DB5AB4F1DBC44CE7B2946EB6C7DE342962893FD387D1B73D7A8672D1F236961170B7EB3579
      953EE5CDC88CD2D
LB1 = 3202030405060708090A0B0C0D0E0F1002020304050607080 90A0B0C0D0E0F10
PB1 = 00685A48E86C79F0F0875F7BC18D25EB5FC8C0B07E5DA4F4370F3A9490340854334B1E1B87FA395464
      C60626124A4E70D0F785601D37C09870EBF176666877A2046D01BA52C56FC8776D9E8F5DB4F0CC27636
      D0B741BBE05400697942E80B739884A83BDE99E0F6716939E632BC8986FA18DCCD443A348B6C3E52249
      7955A4F3C302F676
previous_chain_secret = psk = <NULL>
k1 = 005FC70477C3E63BC3954BD0DF3EA0D1F41EE21746ED95FC5E1FDF90930D5E136672D72CC770742D1711
     C3C3A4C334A0AD9759436A4D3C5BF6E74B9578FAC148C831
context1 = "CASCADE TEST VECTOR 1"
label1 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LB1 || PB1

chain secret1 = 700D4B5BE6772E5384C5522EABFDBD342ED69504A81A0478598521FAF96A4D04E7A91B6A9
                AF99BB4EAB245521194BAA31292776D2457C719E5AD7CE49F3BCD9E
key_material1 = A3B504B4487AB7CE717AB175FDCCDF9EE2CDE91FFFA1E6B740D71C1D8EC2AB78
LA2 = 7102030405060708090A0B0C0D0E0F1101020304050607080 90A0B0C0D0E0F11
PA2 = E1A758EC0D418BFE86D8077B5BB169133C06C1F2A067D8B202D9D058FFC51F63FD26155A6577C74BA7F
      1A27E7BA51982517B923615DEB00BE408920A07831DF5978CFDDD0BF690A264353A4A16B666F90586D7
      F89A193CE09375D389C1379A7A528581C3ACB002CD2DC4F0FD672568FF9050BA8365C7FEFC5E6ED089B
      921DE6804091A0744DE3EB14D426A3F7DA215C50312617C1C2697243980D06056F2CCE88AE7AE73C734
      3C0B7104C9F2870A94FED744CF6E94630514B6CEAB0E64733BB6FA67B931E5D8206010475CBE8BC5872
      48D65D89D8CD9C8BBFA93E8B5F9EB9130773DED665D52ABBD91C4C8C255F73C0FC82501AE33330E9F30
      8DE7177CBF83E4E26E334D7CB09019E638147FC58ED372AF660F14C194BC80E9666325C98E0F8087727
      1D4A6BF514F603703D8A697874CD50A34D92F5AAEA84633CCF96801BD517BF425DEE4A32AAF06684052
      473EA14643C3D535440FB2240A988D09F297C5A388CB3DE60ED943F124034B90EFF611221F80F78EC12
      4956338A105F6636B063D7E48BFBD5D614310FB97D86F122E4AE6F9DDF4977A93ED7D0CE2A94E346A1A
      03D3219CF21907B85A5BCDC713F93A4406A22E03B1655A66E1F6741A2F953E6FE0868B2614BABEF1943
      BBBCB1B66D3E7017E533EA84F291240B56AB33EF1DC3F3DE99DBF9E8BE51A0076E462BCDD825EA96D7F
      63C99177C305C257B31461F4C23D43115F0220409E8880BBB2468586D03461E807BE824B693874911B2
      B52AF06FDBDC47F5A0159729641A7C950AB9E03F2DC045135
LB2 = 7202030405060708090A0B0C0D0E0F1102020304050607080 90A0B0C0D0E0F11
```

```
PB2 = 66D24BC4630B2EE312F01B26EC1D3EC1F583795EC93B90FD9B5453E0BEDA2A70FB6181C9B9EA86A9866
      F1468E62CE853C6C65AA5D0E4535828B3A97E2D1D31DC4372E6A36DA7B0C4733574B91CCE215086C59B
      54F2364F61298004C8410B17658B4021CD36859C94210DE338869CACF0E2DC11412FA5172E1663AAEBE
      F4B5EB0ED9175D6C86C5107DA92B8772342A2F44C93EFFE61F6C76AB8ABA194E862543EDB707E9D2EE8
      84995B1062FE2F60627D5C7673C7AC0D15B08C2F8510DC239463B1B32AD46873F6D1CB5A8579457386F
      D75700989BEED2CA547FE505C581B6B436AABC0F75AD6373A08CEC1504258A972C64EC4A1FEB86BFE32
      ACF3A73ECF815CBC883F39B42C6429A5875BD0BD6A94CEAD587AF49AC8EFB43E1A447D2D8555CB0ADFB
      C9F335F1C599BC9FEAB3E4FE5F2D06D930A58C2FFEEDE0E2726EBD85EC890D1CB0E6870DD784AE30286
      F1A336D57FC41D2F2E2F89765C6A110853BB63E478A64D54A31A18FB4BA44FF58A3662F4D82D544BD9B
      0E94FC88ABC4E4D27D5F6084B5F2162B357A04A1A28C8938834ECC987E50C0A2CACA442850493CE16C0
      47DF677097D3F7EA034BC3D2535504276003DBBEB12F1949C3D369E7EFBA09831E83D622AB2D9277F52
      3946FBAB1DDE14015857EA47663C5CCF30BDF261CCBF31DBE2A560E96CE87FBA80B783350A42C837EB3
      6B2F39A9FED1B649B8ECCE3D3235825F7C800834740546E0CF42C9C2C8B12495225F991B14547E5EEDB
      22858B26EE6E0AE13DBE3D50C6C1EF79C4B97DAD1B0239C4037C1AAC29EE1505E0E527EC81348900E7C
      216A1A1B34B8D2753AF2693647C412
previous_chain_secret = 700D4B5BE6772E5384C5522EABFDBD342ED69504A81A0478598521FAF96A4D04E
                        7A91B6A9AF99BB4EAB245521194BAA31292776D2457C719E5AD7CE49F3BCD9E
k2 = FEE94595E8A05C50113C044D4D8558DA101035EBBF604AA41D0AAA75B8A7F786
context2 = "CASCADE TEST VECTOR 1"
label2 = LA2 || LB2
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain secret2 = 9D231C876512F74EABA852A4D069DA849632A3E00C371BB6BEC243083BF0B6C026968C402
                583C071B1139BFE27F4399A0EB7D8DF2FD4A1B0E8C88CE3529E6035
key_material2 = 109654A3D902FC17C83AD709D436E10FDA9A901CF6A1BDF9A02CB087171A8732
```

C.3.8 ECDH with NIST P-384, SIKEp751, and SHA3-512

The input test vectors for clause C.3.8 are the same as clause C.3.7, only the output changes.

```
chain secret1 = 039666FA148FE83AAF21270BD493488489B0CFA6345EDC7F4757F90E2E0BA6B6959AB1F
                381EE1CE997D4C944B3EDED3E78ADFB6A381F83C439AAB196BDFD2A73
key_material1 = CC15B07AE1B498469441265AC6B24DFDA994ABA4B04E9819FACBD664DE8CCFBB

previous_chain_secret = 039666FA148FE83AAF21270BD493488489B0CFA6345EDC7F4757F90E2E0BA6B69
                        59AB1F381EE1CE997D4C944B3EDED3E78ADFB6A381F83C439AAB196BDFD2A73
chain_secret2 = EC2531D0D42DAB4BBF3831EC9CA6BF4C8DAF81A7174FB640A7D66E0D47BF3027CBE87FB67
                EF64ADB1AC7EDA70F80C127C8E69123B46BB1E79EC737C4EB41FBD1
key_material2 = 70D1D1AF1A736B6C648B7410F728D0AC689E2D8CD54F8B1DDA410CAD566F5CFE
```

C.3.9 ECDH with NIST P-256, Kyber512, and SHA-256

```
LA1 = 101020304050607080090A0B0C0D0E0F1
PA1 = EAD218590119E8876B29146FF89CA61770C4EDBBF97D38CE385ED281D8A6B23028AF61281FD35E2FA70
      02523ACC85A429CB06EE6648325389F59EDFCE1405141
LB1 = 102020304050607080090A0B0C0D0E0F1
PB1 = 700C48F77F56584C5CC632CA65640DB91B6BACCE3A4DF6B42CE7CC838833D287DB71E509E3FD9B060DD
      B20BA5C51DCC5948D46FBF640DFE0441782CAB85FA4AC
previous_chain_secret = psk = <NULL>
k1 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
context1 = "CASCADE TEST VECTOR 1"
label11 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LA1 || PB1

chain secret1: E8633CC42B9C229A2FD60C14F40B47D3AAE16E6EB84C11E55D5138C8B233BF49
key_material1: 9232FCD5DDE269E6EAF001F459A02184

LA2 = 141020304050607080090A0B0C0D0E0F1
```

```
PA2 = 115ACE0E64677CBB7DCFC93C16D3A305F67615A488D711AA56698C5663AB7AC9CE66D547C0595F98A43
      F4650BBE08C364D976789117D34F6AE51AC063CB55C6CA32558227DFEF807D19C30DE414424097F6AA2
      36A1053B4A07A76BE372A5C6B6002791EBE0AFDAF54E1CA237FF545BA68343E745C04AD1639DBC59034
      6B6B9569B56DBBFE53151913066E5C85527DC9468110A136A411497C227DCB8C9B25570B7A0E42AADA6
      709F23208F5D496EBAB7843F6483BF0C0C73A40296EC2C6440001394C99CA173D5C775B7F415D02A5A2
      6A07407918587C41169F2B7178755ACC27FC8B19C4C4B3FCD41053F2C74C8A10A8321241B2802432875
      AE808B9EF1365C7B8A52902F1317BA2FB0269F47930672107B4726FEF64547394D3320C8F120B3C2F47
      25B0305FAB88CC7981FCB09A76A1CBF7F179F43BB0A4C8B0590857F1E69708466C7F8607391E7BC5268
      BFD3D7A1DFFCB4ECA2A1C9B597593013D5FC4202EC2B74E57AB76BBCF3632BBAF97CDC418A6F1639283
      8CA9BF45DDF023777B7561833C105190F94F302C59B531900BBC816361FAA5B3380CA3A893104CA7388
      B185671B3E5FE3790E9A626EC46D9B0B33C7A419AF7B32B6859894F575D82AC5456B5490A7AF8FE6104
      6360589ECBA7244236F4123116B6174AA179249A49195B356C72FC6641F0251812EAA98570B04669907
      0E0819DC2713F469137DFC6A3D7B92B298995EE780369153AC366B06D7249CD09E1B3378FB04399CECB
      8650581D637C79AE67D6F2CAF6ABACF598159A7792CB3C971D1499D2373AD20F63F03BB59ED137384AC
      61A7155143B8CA4932612EC915E4CA346A9BCE5DD60417C6B2A89B1CC435643F875BDC5A7E5B3481CF9
      19EA09172FEBC46D4FC3FB0CB9591704EE2DBB61844B2F3314A06BB6C6D34005E485CE667BDC7D09858
      6928D2D91340F00419EA401351A240A0B041058BEFB0C2FD32645B7A2DF8F5CBFD873327C978D7B351A
      28088438837024C52B9C295CD713646FB5D6C0CCFB470734AC2B2BC8123C2C13DF6938E92455A862639
      FEB8A64B85163E32707E037B38D8AC3922B45187BB65EAFD465FC64A0C5F8F3F9003489415899D59A54
      3D8208C54A3166529B53922
LB2 = 842020304050607080 90A0B0C0D0E0F1
PB2 = EDF24145E43B4F6DC6BF8332F54E02CAB02DBF3B5605DDC90A15C886AD3ED489462699E4ABED44350BC
      3757E2696FBFB2534412E8DD201F1E4540A3970B055FE3B0BEC3A71F9E115B3F9F39102065B1CCA8314
      DCC795E3C0E8FA98EE83CA6628457028A4D09E839E554862CF0B7BF56C5C0A829E8657947945FE9C225
      64FBAEBC1B3AF350D7955508A26D8A8EB547B8B1A2CF03CCA1AABCE6C3497783B6465BA0B6E7ACBA821
      195124AEF09E628382A1F914043BE7096E952CBC4FB4AFED13609046117C011FD741EE286C83771690F
      0AEB50DA0D71285A179B215C6036DEB780F4D16769F72DE16FDADAC73BEFA5BEF8943197F44C59589DC
      9F4973DE1450BA1D0C3290D6B1D683F294E759C954ABE8A7DA5B1054FD6D21329B8E73D3756AFDA0DCB
      1FC8B1582D1F90CF275A102ABC6AC699DF0C5870E50A1F989E4E6241B60AAA2ECF9E8E33E0FFCF40FE8
      31E8FDC2E83B52CA7AB6D93F146D29DCA53C7DA1DB4AC4F2DB39EA120D90FA60F4D437C6D00EF483BC9
      4A3175CDA163FC1C2828BE4DBD6430507B584BB5177E171B8DDA9A4293C3200295C803A865D6D2166F6
      6BA5401FB7A0E853168600A2948437E036E3BF19E12FD3F2A2B8B343F784248E8D685EB0AFDE6315338
      730E7A1001C27D8D2A76FA69D157BA1AC7AD56DA5A8C70FE4B5B8D786DC6FC0566BA8E1B8816334D32A
      3FB1CE7D4D5E4C332AF7B003D091741A3D5C965292255DFF8ED2BBF1F9116BE50C17B8E548748AD4B2E
      957BBD1953482A2E1718CEC66CD2C81F572D552B7187885E6B8943D6431413C59EBB7E036048490BE52
      89E95B20A89E8B159F61A9A9886E147568F4C9021F362F02688A1C8C3BB0D24086880E55B6EDB43F374
      5D2C166DC1CB743C76FE6BE523A893CC764D16435C37851252A81E2FFBA0F18971A3DEE37D4877CB928
      E36E5235037A6B2057897D518A5F0E348E3AB6D5B52DFC60757F3B41A4FEC7828F1DEEAF4587CCC8EAD
      F647F4D203B2FAA05A649B582340CB4CACE57A30711BE752FACF0227D0A80C4128442DDC544BE805B9C
      FE8FE9B1237C80F96787CD9281CCF270C1AFC0670D
previous_chain_secret = E8633CC42B9C229A2FD60C14F40B47D3AAE16E6EB84C11E55D5138C8B233BF49
k2 = 46FC62106420FF012E54A434FBDD2D25CCC5852060561E68040DD7778997BD7B
context2 = "CASCADE TEST VECTOR 1"
label2 = LA2 || LB2
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain_secret2 = E8633CC42B9C229A2FD60C14F40B47D3AAE16E6EB84C11E55D5138C8B233BF49
key_material2 = A324764EEA9EBA338F0596D6D64DFA55
```

C.3.10 ECDH with NIST P-256, Kyber512, and SHA3-256

The input test vectors for clause C.3.10 are the same as clause C.3.9, only the output changes.

```
chain secret1 = 87ED4CD0E14F2590EB2FDFA2767D05840C90DD0D8211F46A1321648F6CFE05F9
key_material1 = A791AA8C4A914F4937E794655076CA1B

previous_chain_secret = 87ED4CD0E14F2590EB2FDFA2767D05840C90DD0D8211F46A1321648F6CFE05F9
chain_secret2 = 87ED4CD0E14F2590EB2FDFA2767D05840C90DD0D8211F46A1321648F6CFE05F9
key_material2 = 07930E42BCA7B1C60F530DA08A1C85F1
```

C.3.11 ECDH with NIST P-384, Kyber768 and SHA-384

```
LA1 = 2110203040506070809 0A0B0C0D0E0F10010203040506070
PA1 = FCFCEA085E8CF74D0DCED1620BA8423694F903A219BBF901B0B59D6AC81BAAD316A242BA32BDE85CB24
      8119B852FAB66972E3C68C7AB402C5836F2A16ED451A33120A7750A6039F3FF15388EE622B7065F7122
      BF6D51AEFBC29B37B03404581B
LB1 = 2120203040506070809 0A0B0C0D0E0F10020203040506070
PB1 = 1AEFBFA2C6C8C855A1A216774550B79A24CDA37607BB1F7CC906650EE4B3816D68F6A9C75DA6E4242CE
      BFB6652F65180419D28B723EBADB7658FCEBB9AD9B7ADEA674F1DA3DC6B6397B55DA0F61A3EDDACB4AC
      DB14441CB214B04A0844C02FA3
previous_chain_secret = psk = <NULL>
k1 = 3D2E640F350805EED1FF43B40A72B2ABED0A518BCEBE8F2D15B111B6773223DA3C3489121DB173D414B
      5BD5AD7153435
context1 = "CASCADE TEST VECTOR 1"
label1 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LB1 || PB1

chain secret1 = A682BE9E6A411C51B1166331D6CCB807787879F97638DDCEAC4F9578F1010E90FCD11974
                E894F5D106A5E56362CB4792
key_material1 = E38249E087F5AD3C283C1A37D0B1B24DB2B63C1BE28A58F5

LA2 = 2510203040506070809 0A0B0C0D0E0F11010203040506070
PA2 = A72C2D9C843EE9F8313ECC7F86D6294D59159D9A879A542E260922ADF999051CC45200C9FFDB60449C4
      9465979272367C083A7D6267A3ED7A7FD47957C219327F7CA73A4007E1627F00B11CC80573C15AEE664
      0FB8562DFA6B240CA0AD351AC4AC155B96C14C8AB13DD262CDFD51C4BB5572FD616553D17BDD430ACBE
      A3E95F0B698D66990AB51E5D03783A8B3D278A5720454CF9695CFDCA08485BA099C51CD92A7EA7587C1
      D15C28E609A81852601B0604010679AA482D51261EC36E36B8719676217FD74C54786488F4B4969C05A
      8BA27CA3A77CCE73B965923CA554E422B9B61F4754641608AC16C9B8587A32C1C5DD788F88B36B717A4
      6965635DEB67F45B129B99070909C93EB80B42C2B3F3F70343A7CF37E8520E7BCFC416ACA4F18C79812
      62BA2BFC756AE03278F0EC66DC2057696824BA6769865A601D7148EF6F54E5AF5686AA2906F994CE38A
      5E0B938F239007003022C03392DF3401B1E4A3A7EBC6161449F73374C8B0140369343D9295FDF511845
      C4A46EBAAB6CA5492F6800B98C0CC803653A4B1D6E6AAED1932BACC5FEFAA818BA502859BA5494C5F54
      02C8536A9C4C1888150617F80098F6B2A99C39BC5DC7CF3B5900A21329AB59053ABAA64ED163E859A8B
      3B3CA3359B750CCC3E710C7AC43C8191CB5D68870C06391C0CB8AEC72B897AC6BE7FBAACC676ED66314
      C83630E89448C88A1DF04ACEB23ABF2E409EF333C622289C18A2134E650C45257E47475FA33AA537A5A
      8F7680214716C50D470E3284963CA64F54677AEC54B5272162BF52BC8142E1D4183FC017454A6B5A496
      831759064024745978CBD51A6CEDC8955DE4CC6D363670A47466E82BE5C23603A17BF22ACDB7CC984AF
      08C87E14E27753CF587A8EC3447E62C649E887A67C36C9CE98721B697213275646B194F36758673A8ED
      11284455AFC7A8529F69C97A3C2D7B8C636C0BA55614B768E624E712930F776169B01715725351BC74B
      47395ED52B25A1313C95164814C34C979CBDFAB85954662CAB485E75087A98CC74BB82CA2D1B5BF2803
      238480638C40E90B43C7460E7AA917F010151FAB1169987B372ABB59271F7006C24E60236B84B9DDD60
      0623704254617FB498D89E58B0368BCB2103E79353EB587860C1422E476162E425BC2381DB82C659273
      7E1DD602864B0167A71EC1F223305C02FE25052AF2B3B5A55A0D7A2022D9A798DC0C5874A98702AAF40
      54C5D80338A5248B5B7BD09C53B5E2A084B047D277A861B1A73BB51488DE04EF573C85230A0470B7317
      5C9FA50594F66A5F50B4150054C93B68186F8B5CBC49316C8548A642B2B36A1D454C7489AC33B2D2CE6
      668096782A2C1E0866D21A65E16B585E7AF8618BDF3184C1986878508917277B93E10706B1614972B2A
      94C7310FE9C708C231A1A8AC8D9314A529A97F469BF64962D820648443099A076D55D4CEA824A583048
      44F99497C10A25148618A315D72CA857D1B04D575B94F85C01D19BEF211BF0AA3362E7041FD16596D80
      8E867B44C4C00D1CDA3418967717F147D0EB21B42AAEE74AC35D0B92414B958531AADF463EC6305AE5E
      CAF79174002F26DDECC813BF32672E8529D95A4E730A7AB4A3E8F8A8AF979A665EAFD465FC64A0C5F8F
      3F90034894158 99D59A543D8208C54A3166529B53922
LB2 = 9520203040506070809 0A0B0C0D0E0F11020203040506070
```

```
PB2 = B52C56B92A4B7CE9E4CB7C5B1B163167A8A1675B2FDEF84A5B67CA15DB694C9F11BD027C30AE22EC92
      1A1D911599AF0585E48D20DA70DF9F39E32EF95D4C8F44BFEFDAA5DA64F1054631D04D6D3CFD0A540DD
      7BA3886E4B5F13E878788604C95C096EAB3919F427521419A946C26CC041475D7124CDC01D0373E5B09
      C7A70603CFDB4FB3405023F2264DC3F983C4FC02A2D1B268F2208A1F6E2A6209BFF12F6F465F0B069C3
      A7F84F606D8A94064003D6EC114C8E808D3053884C1D5A142FBF20112EB360FDA3F0F28B172AE50F5E7
      D83801FB3F0064B687187074BD7FE30EDDAA334CF8FC04FA8CED899CEADE4B4F28B68372BAF98FF482A
      415B731155B75CEB976BE0EA0285BA01A27F1857A8FB377A3AE0C23B2AA9A079BFABFF0D5B2F1CD9B71
      8BEA03C42F343A39B4F142D01AD8ACBB50E38853CF9A50C8B44C3CF671A4A9043B26DDBB24959AD6715
      C08521855C79A23B9C3D6471749C40725BDD5C2776D43AED20204BAA141EFB3304917474B7F9F7A4B08
      B1A93DAED98C67495359D37D67F7438BEE5E43585634B26C6B3810D7CDCBC0F6EB877A6087E68ACB848
      0D3A8CF6900447E49B417F15A53B607A0E216B855970D37406870B4568722DA77A4084703816784E2F1
      6BED18996532C5D8B7F5D214464E5F3F6E905867B0CE119E252A66713253544685D208E1723908A0CE9
      7834652E08AE7BDC881A131B73C71E84D20D68FDEFF4F5D70CD1AF57B78E3491A9865942321800A203C
      05ED1FEEB5A28E584E19F6535E7F84E4A24F84A72DCAF5648B4A4235DD664464482F03176E888C28BFC
      6C1CB238CFFA35A321E71791D9EA8ED0878C61121BF8D2A4AB2C1A5E120BC40ABB1892D1715090A0EE4
      8252CA297A99AA0E510CF26B1ADD06CA543E1C5D6BDCD3B9C585C8538045DB5C252EC3C8C3C954D9BE5
      907094A894E60EAB43538CFEE82E8FFC0791B0D0F43AC1627830A61D56DAD96C62958B0DE780B78BD47
      A604550DAB83FFF227C324049471F35248CFB849B25724FF704D5277AA352D550958BE3B237DFF473EC
      2ADBAEA48CA2658AEFCC77BBD4264AB374D70EAE5B964416CE8226A7E3255A0F8D7E2ADCA062BCD6D78
      D60D1B32E11405BE54B66EF0FDDD567702A3BCCFEDE3C584701269ED14809F06F8968356BB9267FE86E
      514252E88BB5C30A7ECB3D0E621021EE0FBF7871B09342BF84F55C97EAF86C48189C7FF4DF389F077E2
      806E5FA73B3E9458A16C7E275F4F602275580EB7B7135FB537FA0CD95D6EA58C108CD8943D70C164311
      1F4F01CA8A8276A902666ED81B78D168B006F16AAA3D8E4CE4F4D0FB0997E41AEFFB5B3DAA838732F35
      7349447F387776C793C0479DE9E99498CC356FDB0075A703F23C55D47B550EC89B02ADE89329086A508
      43456FEDC3788AC8D97233C54560467EE1D0F024B18428F0D73B30E19F5C63B9ABF11415BEA4D017013
      0BAABD33C05E6524E5FB5581B22B0433342248266D0F1053B245CC2462DC44D34965102482A8ED9E4E9
      64D5683E5D45D0C8269
previous_chain_secret = A682BE9E6A411C51B1166331D6CCB807787879F97638DDCEAC4F9578F1010E90F
                        CD11974E894F5D106A5E56362CB4792
k2 = 3D2E640F350805EED1FF43B40A72B2ABED0A518BCEBE8F2D15B111B6773223DA3C3489121DB173D414B5
     BD5AD7153435
context2 = "CASCADE TEST VECTOR 1"
label2 = LA2 || LB2
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain_secret2 = A682BE9E6A411C51B1166331D6CCB807787879F97638DDCEAC4F9578F1010E90FCD11974
                E894F5D106A5E56362CB4792
key_material2 = 45F0EA18A6B22D94139BAF38F3C932277B48B4113A6B4E27
```

C.3.12 ECDH with NIST P-384, Kyber768, and SHA3-384

The input test vectors for clause C.3.12 are the same as clause C.3.11, only the output changes.

```
chain secret1 = F1ACA57EF88380CC99CB466860AC7FD134E3965D92AB8CEE8892555ADE839382913874FFE
                1E4F7648A2130D47BFD8A01
key_material1 = 5F06BB2A810783E39A09BE5EE24A740C69854E998E77D29B

previous_chain_secret = F1ACA57EF88380CC99CB466860AC7FD134E3965D92AB8CEE8892555ADE8393829
                        13874FFE1E4F7648A2130D47BFD8A01

chain_secret2 = F1ACA57EF88380CC99CB466860AC7FD134E3965D92AB8CEE8892555ADE839382913874FFE
                1E4F7648A2130D47BFD8A01
key_material2 = 220D8F8C87FC3CD29FB1E9D24E5C6FAE4A6E42E085EB2D5F
```

C.3.13 ECDH with NIST P-384, Kyber1024 and SHA-512

```
LA1 = 3310203040506070809 0A0B0C0D0E0F1001020304050607080 90A0B0C0D0E0F1
PA1 = 00D45615ED5D37FDE699610A62CD43BA76BEDD8F85ED31005FE00D6450FBBD101291ABD96D4945A8B57
      BC73B3FE9F4671105309EC9B6879D0551D930DAC8BA45D25501425332844E592B440C0027972AD15264
      31C06732DF19CD46A242172D4DD67C2C8C99DFC22E49949A56CF90C6473635CE82F25B33682FB19BC33
      BD910ED8CE3A7FA
LB1 = 3320203040506070809 0A0B0C0D0E0F1002020304050607080 90A0B0C0D0E0F1
PB1 = 01DF277C152108349BC34D539EE0CF06B24F5D3500677B4445453CCC21409453AAFB8A72A0BE9EBE54D
      12270AA51B3AB7F316AA5E74A951C5E53F74CD95FC29AEE7A013D52F33A9F3C14384D1587FA8ABE7AED
      74BC33749AD9C570B471776422C7D4505D9B0A96B3BFAC041E4C6A6990AE7F700E5B4A6640229112DEA
      FA0CD8BB0D089B0
previous_chain_secret = psk = <NULL>
k1 = 000B3920AC830ADE812C8F96805DA2236E002ACBBF13596A9AB254D44D0E91B6255EBF1229F366FB5A05
      C5884EF46032C26D42189273CA4EFA4C3DB6BD12A6853759
context1 = "CASCADE TEST VECTOR 1"
label1 = LA1 || LB1
MA1 = LA1 || PA1
MB1 = LB1 || PB1

chain secret1 = 2B2449BED2C5272C1433A503D779745A36029857DCE637F6EEF6C1A92060877FCB0FA0AC4
                0DD02B31EB988A16AFEF26FC73DFAE580AEB0924547724426DB69A3
key_material1 = 55DB44982A5AB24429E498AB7FB1882114F73843C42650A0EE19FB70E3852B2A

LA2 = 3102030405060708090 A0B0C0D0E0F1101020304050607080 90A0B0C0D0E0F11
PA2 = D22302CBD3399FACC630991FC8F28BDB4354762541527678BCF61F65C241146C426D23B9BFAA6B7DF1
      8C97F20C1B6125BF874B1D89475852C448215DB0EB7737F91480E8CEBD9A0871574F5AB62D9020175EC
      6927CA0B54C09818E42CF92A383172422C7DC1831D63B0C295DE75159DB8034E9E07F7B0B910C3C1E5F
      B66B3DC523F1FA6EB4910CB89A6C17562C83AB4C18D0CD7E0796592A372AA409B1C557347CCACDC4644
      A119064D06DD474929D1C6FB4D686E5491CE4BC89A30BB4B8C41BCE5157DFC1360823B1AB618C14B10F
      98C25067398EA7018C278A4B3DF31334D603B2044EF187CD9BC6CE42725BD962C264983E9E18155A8B9
      C47143D70460A26A56FE7658C1F150348C6087EF758AD167887860A007A5FC37358D43B5EBEE820ACEA
      474F0AC07B76802866199C61231D5C747C93774D2C1E0C1C67E6C81B82752173E125BAF39B4FD19A4F4
      53DC57976B1D97FE6996992BBB65B7CB25D077BBAA6A13322899AF659CF1B3558C1B5001154B625809E
      D89AEEBB89E6EA7D67F723D045AB05715C42355DA6A5C8DD39C8ABE3037751A01ED1C7374919F3121B5
      A52C53D1487316769F80721DEEAAAD3C90F76E7AE9E12BA92B32B5FD457E3C752C2650DFB885771CB77
      AC3C785A8C562E6A1C63C2A55EA47CF8B90EB8225C123C346452566235B2F31823A33521E087937A345
      D8D663EEAA05658917BBAA008C2E335F8850A90A326D0E66432F44CEB8289E4ECB2D12958E984072ECA
      CB88E1348FF0B55654ACBA5B54971CBAEBA88EC4B91A94C37192FA982BECB9F3DA421603B61A51BC8E3
      6CBD053851C77B1B926B17A272AA9023246B02B3ED47F66A00BD5684823634E7CE58CF8F306E35B1E53
      22824D904801F0A2FA7C2BC9C252B0A56B7BA2AB0F636021745A70A9A43E2B0A8D615970B65309624B5
      184BCC30B911679AEDD76025FE3908FD67897B0CF4BE5A6F5413D7DD98564B23E42A93E4AA8821CD450
      54C643EDC1158DB6B3DEB13FB5A51EBD1A8A78B87225A7338E101104C4A220D9BDEDD48C85A1C2DAE78
      1A80C40E13B87EAC73A764201C9B760CCFB1AE392699C7039D27C39362B27B8FC6F07A8A3D4410F1547
      C48A9997F62C61074452EF1515F8A649EBCA9437205A4E8A61606B41DAF6834D671F4D852C0C9C40966
      11648C6A3170678B1537CC1828D93580C9E5849A9653175ACB753F2BE7437BE45F6C603E485F2EC301B
      B42B6C37C225D7495A584AE231890AB5C8C35C268CF4BBB0213C096019319561A8A6947637AA40D006B
      415BB2CFA2237E0890B6A3BC134ABF8F6585E108D15940F91F4BF5B0C818055B21DEA6E63B553988C47
      F4B94E7CF800A493B4734705EDC56A4B6021C629500675876804CF0B951F038A5C7FE58E89774EF2992
      FD7C63099D352A7D21560B788B405709861817E59A96B3A3A83CBA803B16934331071905BBEC6532900
      155D8AC88CB32E4E21A3BD3A03FDEC325A51CD2773964E6784FCF1853737AA64EB67564727272661ABF
      84313A57A44B123C65509CFB7A6F6641CDCC3B57FE628C7B8192DB44FFBF5796A8613B1FA126F607688
      3C783DC24E2A4464C40B3A41CA70AE87620866CF4FCB2BD204BF5C283812BA056AC0C345E379C4BA24D
      750901279BB2F3A16F612BFADB35703332C7C136F68EAB6755C66B6A4AD1AABA7B768A58ACAACC10A45
      9A1CC8EF29377BC200E4D315A30A6BCC3256F9734D06E9779CAA5442A9A16069081377C76E751543680
      72DC446ED6C8B8E622A21E383CF9BA1FB434E2ECC81E7B78CEE986B8FF798AB18CF9634543546284EDA
      2A26B47F05B735BCDB1202220076DC8B4E4B9F853533C8F6C7FF38817BA49712835785F17F14CA01D0C
      1C1E98810FE0B36E5B427157B9418449CEDD641A4293C85C32700102ACEC22EBAD98ED160A5F027BD4C
      DA57F1F3720A12C134654DD5E73F829676495390D0E7929D6034E9C55F7D55BA658BC587988E8AF9496
      0F6CFB8D5AF7A0021535A6E25E437D49A780698BE22AC9953949F571B85A685725F8207A2B0AE849B60
      1AB91B159B3DF4A154C2041E776070AFC42969322380917C97510799F3149131477E16663D3174C7C1C
      AEA788535C6C005A64F2868631B31B66E205FD38C1D84542D0F1B578F58C9BF5A0FAEAB6AB649489305
      3165EAFD465FC64A0C5F8F3F9003489415899D59A543D8208C54A3166529B53922
LB2 = A7202030405060708090A0B0C0D0E0F1102020304050607080 90A0B0C0D0E0F1
```

```
PB2 = A6AF29D5F5B80BD130F518BADDD6C8F17545413D860FB3DE451979EBFA5E4E3112C7C0ADF99824BB52
      6F2C3550748ED0E134F0457A7C61F9F526F002BAADC03FC13E38131219513C3EDE061661E74F603C4FC
      F7951C8E52C9C213B0D22D9293663D669A6B58ED8FCEFCF8249D7BB5298F55761445B2B83CE7F005CB0
      4248AEC8BDA22FD2D42AA766322014EA038CC32C55C8E4B9E28EC9119F527341E4F66A035121073B85D
      E6706DA19E0838A9F33B719A68F039B664DC002659EABFC398679AA7009CE0CD01CDAFB6CD2A26FE410
      1672C98FF58F7C47D5BDA2906653B3A6F9651F7A121EA77EA74723FAE5B873F9BB7B664F0C8A93831EF
      9D51C7CC1EF44AC0E55A55CA76D137FE9B75F40509CEF156E5AD18F9FB999680008E547D55EECD5B4D1
      CB1D9F076CEC21501C7402509ECB77AFB2CB9A61340A8BD1514C6E71B4AA45E47EC37512271B911F8FB
      46C9082C9DF07204ABB5A50E6E3647A8AD4D8D5D7BFF19C8A509308BCFB895536D045CA2B97CB16A29B
      B7181CAD0509DDB91735028EBA8C31D74BD275EAA65B5340B3A43FBFE0B3061D6BAE7E75B7098CDABE9
      1D4B31E36C9AA7A8298862AD63C8FD282E03B460B3AB464CE0F27B1C3D11155ACAA011EB9E2AE3E6DDA
      07D6F491737CBCE9B05F9BC56BE20E8D326BA132C57FB235161144519CDF40560FBE279BDE411E11253
      1F826D6AB10D4547350ADD2A9DE8D62C2AC82CABE6815646F4DC9742BB0C2A3F77EC7B46C6B537605FA
      31798CD89281221A33DFB9796E644305630332C2CB931408AB481A16D953F6BEAE3891D6D9AC1FAB382
      22D9271872D9D0CADB91ABE9B4E265F75C6E5E829E146C3D8CE1E9D12E0D129801957F46B0D2DBE1F74
      9B1D08E2345F6239A731342EB75B0CF1BF411749BC2CAF2810B788C6B7238B4D3DA2D6315CE9542E244
      04F145755A30AB851E4445841BD33F716A586884888ECC6BC6498AA32919AE81D20C26973C2BD54582A
      0F6AD98ABFD2627E15690A727E69F581DD2A7127982A90E33E2D4A03FE339142C7E44C326AC46ED395A
      225D3033389917328B45316B1585A01B2C304B2944E903ABBB3EC5619441CFC8965A446DF75DEFA80C6
      E15ADBD506B7AB2DE12DDA9BC81441CFC89052E2E5808F7126C6FD3AC6AC8081258A84A09AE50F6CD7C
      C0F4AF336FD1D643E99079996268C2D32D909F22E3504F07FBB563196D4312FDDB9335D5C1D36E8C5EE
      A2278DBA23B94D193C947CC41CA993DC7DB1396340AD9C4FE687DD7B8D0C7A5120AE0204F2C665BD5F4
      73D644C7FF26BFFBA7A36980830702128A7E661D677A092A36E7428A4139FB29B0095CC11086F447D2A
      9EF6C9B161F189C6299E084CB7AA00FAF787797BFB069FBC087FDE26252A1664F19C5A8A22EC5EE1AEB
      076357B7DC37E6B0F1520F958F7851BACB92C89FD114A72FEAC54652D45B09E1AE7651ABD164BCD537D
      58FA39D3EC8ACDCDF98425005862FA59692DE162B77E6297C66233348408A8AB695CE2F2728DB9FBE27
      E958967EC5974767C5A66023074B4A71AFD264AD2890E970A1F31D6E3311B736F9F9488793DDC88F234
      58064254C82A1D9E59EAD2FCEC40B430687C4B7E28960926AFCACC9BD756A71088C78450E20A2E980AE
      DE9EBEDFE7FABD6ABFE96F934C4B02C01CA194D01B73C25D5997039D3FCD0F099521F70CAEE69110AC1
      FC5A99917AD752FC96ADFAD7186D0A7C9CFE5601C07514EA6448D661C57AA20242103C4276A070A489A
      4CB6BCA0F9ECC4379FB220215FD91F81019D5B0AE619358B52468F272C178E3A74CF6775AA924FE329C
      3175D9E4C3E21AB9EC836EDC3ACAB2E3891EE8DEDA515D39AF9B8DDD0EE7B0164F805C3835F6D2BABDB
      30EAB4756E7EC7F829ECE01E8EADFBBED12FC283B3D4C69F575E7F80417689FDFCFC7BE27EE3B8CDF57
      AAEBEC4A95B7E5BB585B85227F7C32BE30DB3E65E42E30DCF5A5FA073DBA399D942F2222ADB9B989810
      2AFE5432EDC7F04AE34A8FEC2D81CB49A9A9B43814CE71D97F726E2B1E8F64B50E65DFB4816E12E82A3
      197484A4E9BBA4D2D69E3F19D0B75C21E2BFFE9FC0C98CF48A3AAF08D467F72687DF0178174B7897F73
      4349B181ECA86A598A0C5E8C25946F24DC5572BD324A40458A788E5137F3C7A7C97FC9F12A3C463A8FE
      9449101CCE966D7C009323932998D56EF430C73BC24F5D95F737858DDC4F32C013
previous_chain_secret = 2B2449BED2C5272C1433A503D779745A36029857DCE637F6EEF6C1A92060877FC
                        B0FA0AC40DD02B31EB988A16AFEF26FC73DFAE580AEB0924547724426DB69A3
k2 = 000B3920AC830ADE812C8F96805DA2236E002ACBBF13596A9AB254D44D0E91B6255EBF1229F366FB5A05
     C5884EF46032C26D42189273CA4EFA4C3DB6BD12A6853759
context2 = "CASCADE TEST VECTOR 1"
label2 = LA1 || LB1
MA2 = LA2 || PA2
MB2 = LB2 || PB2

chain_secret2 = 2B2449BED2C5272C1433A503D779745A36029857DCE637F6EEF6C1A92060877FCB0FA0AC
                40DD02B31EB988A16AFEF26FC73DFAE580AEB0924547724426DB69A3
key_material2 = EC38A129393864C9A949365155FD6A6943E30459DB80304EFEB83047B903D117
```

C.3.14 ECDH with NIST P-384, Kyber1024, and SHA3-384

The input test vectors for clause C.3.14 are the same as clause C.3.13, only the output changes.

```
chain secret1 = B28CEA3C1377E5775B6F88F412EFFDCC7F67E812526DC1C557CD23BF39F802A1F28FFDD1
                5E542C720FB6C91FA9BBF1B6CBF3498CB447C3B31819A72BF58D22EF
key_material1 = CABC697BDEA88DE1961E3DE50F1A124D2EC150C67FB0ABC0BA00D10E9A762EDA

previous_chain_secret = B28CEA3C1377E5775B6F88F412EFFDCC7F67E812526DC1C557CD23BF39F802A1F
                        28FFDD15E542C720FB6C91FA9BBF1B6CBF3498CB447C3B31819A72BF58D22EF

chain_secret2 = B28CEA3C1377E5775B6F88F412EFFDCC7F67E812526DC1C557CD23BF39F802A1F28FFDD15
                E542C720FB6C91FA9BBF1B6CBF3498CB447C3B31819A72BF58D22EF
key_material2 = C2D57FA0CF6175F31581634849703060E203346823D091C182B5520F3714A487
```

History

Document history		
V1.1.1	December 2020	Publication