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

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

Reference

RTS/CYBER-QSC-0019

Keywords

key exchange, quantum safe cryptography

ETSI

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

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

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

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

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

Foreword

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

Modal verbs terminology

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

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

Introduction

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

1 Scope

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

2 References

2.1 Normative references

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

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

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

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

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

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

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

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

[5] Void.

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

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

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

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

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

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

2.2 Informative references

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

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

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

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

[i.2] Void.

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

[i.4] Void.

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

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

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

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

[i.9] Void.

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

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

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

[i.13] Void.

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

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

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

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

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

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

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

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

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

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

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

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

info: octet string set by the application as additional information

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

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

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

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

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

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

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

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

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

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

public key cryptography: See asymmetric cryptography.

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

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

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

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

3.2 Symbols

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

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

sk	A private key for an asymmetric cryptographic scheme

3.3 Abbreviations

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

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

4 Purpose of quantum-safe hybrid key establishment

4.1 Status of quantum-safe key encapsulation mechanisms

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

The present document addresses the following cases:

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

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

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

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

5 Architecture for quantum-safe hybrid key establishment

5.1 Functional entities

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

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

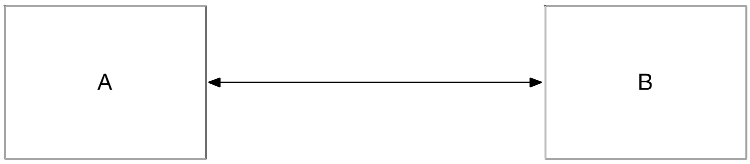

Figure 1: Communicating entities *A* and *B*

5.2 Information relationships (reference points)

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

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

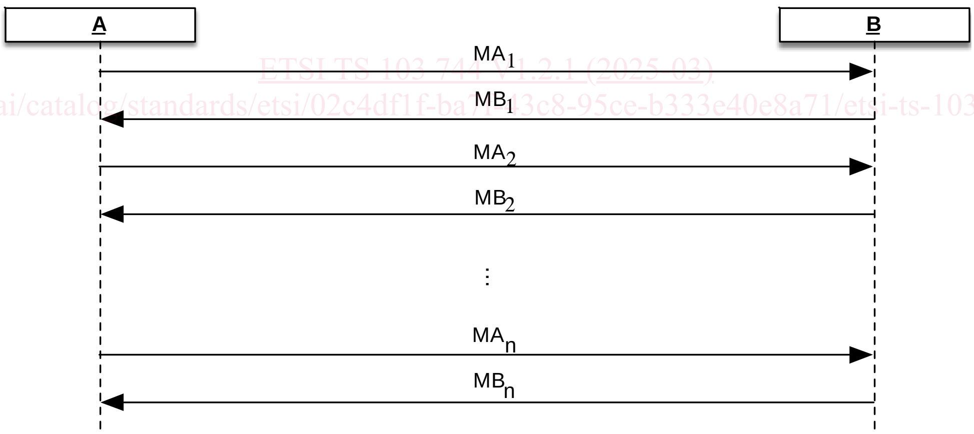

Figure 2: Messages exchanged between entities *A* and *B*

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

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

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

6 Introductory information

6.1 Introduction

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

6.2 Notation

6.2.1 Radix

The prefix "0x" indicates hexadecimal numbers.

6.2.2 Conventions

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

<variable> = <expression>

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

$$x = x + y + 3$$

means:

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

6.2.3 Bit/Byte ordering

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

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

0x0123456789ABCDEFFEDCBA98765432108654538lAB594FC28786404C50A37…

then:

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

6.2.4 Integer encoding

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

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

I = 0x00000025

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

7 Cryptographic primitives

7.1 Hash functions (hash)

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

$$digest = hash(input)$$

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

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

7.2 Context formatting function (*f*)

7.2.1 Context formatting function (*f*) description

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

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

where the parameters and output shall be defined as follows.

Input:

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

Output:

context - an octet string representing the context value.

7.2.2 Concatenate-based context formatting function

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

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

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

Input:

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

Process:

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

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

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

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

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

   b) Set $context = context \,||\, L_i \,||\, val_i$.

3) Return *context*.