ETSI TS 104 015 V1.1.1 (2025-02)

Cyber Security (CYBER); Quantum-Safe Cryptography (QSC); Efficient Quantum-Safe Hybrid Key Exchanges with Hidden Access Policies

Reference

DTS/CYBER-QSC-0023

Keywords

anonymity, authentication, encryption,
key exchange, quantum safe cryptography,
traceability

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.

Executive summary

The present document will provide a framework to build Key Encapsulation Mechanisms (KEMs) retaining both pre-quantum and post-quantum security through hybridization, with additional features for practical use, while being efficient enough for browser or mobile use. Namely, keys will be encapsulated with respect to user-attributes, the encapsulations will be anonymous, and any user having attributes fulfilling the encapsulation policy will be able to retrieve the keys, while those who are not authorized will not be able to. Since many users could have the same attributes, the scheme includes an optional tracing feature, in which a tracing authority would have the means to distinguish users with the same attributes, to possibly later deactivate rights in case of abuse.

Introduction

Key Encapsulation Mechanisms (KEMs) provide the most efficient practical instantiations of Public-Key Encryption (PKE) mechanisms when combined with a Data Encapsulation Mechanism (DEM) to encrypt large amounts of data. The KEM-DEM paradigm introduced by Shoup in [i.4] wisely combines a public-key scheme together with a symmetric encryption scheme to create a scheme with ciphertexts of similar size to plaintexts.

In short, KEMs provide a tool to transmit session keys. First, a user runs the encapsulation procedure with respect to a recipient or set of recipients, generating a session key and an encapsulation of the latter. The recipients are provided with this encapsulation and, if they were among the set of intended recipients, are then able to derive the session key from it. The payload is then encrypted/decrypted under this session key using a DEM, which can be implemented by any authenticated encryption mechanism.

In order to provide additional confidence during the post-quantum migration, it is possible to hybridize two KEMs so that the security of the scheme relies on the stronger of two component algorithms. That is, if one component KEM algorithm is vulnerable to a cryptographic attack, then the privacy of the encapsulated keys is nonetheless maintained. This can be done with one pre-quantum and one post-quantum secure scheme (post-quantum schemes are resistant to adversaries with a cryptographically relevant quantum computer).

Moreover, for fine-grained access-control of users able to decrypt the payload, one just needs fine-grained access-control on the KEM part. Attribute-Based Encryption (ABE) - which often first generates an encapsulated session key, in the KEM-DEM paradigm- has been proposed to control decryption with respect to attributes and policies in ciphertexts and a user's keys [i.2]. More advanced ABE schemes have been proposed in the literature to handle complex access policies, but at a high computational cost and large ciphertexts, in particular when one wants post-quantum security.

To give two specific examples, a first work [i.3] proposed key-policy ABE, where a Boolean formula (the policy) is associated to the user's key, and attributes associated to the ciphertext, so that the user can decrypt if and only if the Boolean formula accepts on the ciphertext's attributes. The more general work [i.2] also defines ciphertext-policy ABE, where a Boolean formula (the policy) is associated to the ciphertext, and attributes associated to the user's key, so that the user can decrypt if and only if Boolean formula in the ciphertext accepts on the user's attributes.

The present document follows the approach of ciphertext-policy ABE, with policies with particular properties for efficiency reasons.

The scheme specified in the present document targets particular access-structures, with several orthogonal dimensions, with a hybrid KEM, providing fine-grained access control, key rotation to allow dynamicity of users and user's rights and an optional traceability feature that allows detection of abuse by individual users. It is described in a black box model, allowing component cryptographic algorithms to be selected according to the preferences of the implementer.

1 Scope

The present document specifies methods to efficiently build and instantiate Key Encapsulation Mechanisms (KEMs) with hidden access policies, while having the privacy of encapsulated keys relying on the best security of two hybridized schemes, namely with an instantiation where the privacy relies on the Computational Diffie-Hellman (CDH) classical assumption and the Learning With Errors (LWE) post-quantum assumption. Both problems have to be broken to endanger the privacy of the encapsulated key.

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

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

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

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

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

[6] FIPS PUB 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] Théophile Brézot, Paola de Perthuis, and David Pointcheval: "Covercrypt: an Efficient Early-Abort KEM for Hidden Access Policies with Traceability from the DDH and LWE". ESORICS 2023.

[i.2] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters: "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". ACM CCS 2006.

[i.3] Amit Sahai and Brent Waters: "Fuzzy identity-based encryption". EUROCRYPT 2005.

[i.4] ISO/IEC 18033-2: "Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

adversary's advantage: probability for an adversary to distinguish two distributions

NOTE: Formally, for an adversary A, given two distributions D_0 and D_1, the advantage is defined as:

$$\mathrm{Adv}(A) = \Pr_{D_1}[A(x) = 1] - \Pr_{D_0}[A(x) = 1] = 2 \,.\, \Pr_{b,D_b}[A(x) = b] - 1.$$

negligible probability in κ: probability that is smaller than the inverse of any polynomial in κ, for κ large enough

oracle access: efficient evaluation of a function for inputs of their choice

overwhelming probability in κ: probability p such that 1-p is negligible in κ

polynomial time: running time can be expressed as a polynomial in the security parameter

security parameter: number of bits in the security level

NOTE: If the security parameter is equal to κ, then the security should hold except with probability less than $2^{-\kappa}$.

3.2 Symbols

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

1^{κ}	The security parameter κ taken as input to an algorithm
||	The logical (non-exclusive) OR
&&	The logical AND
$\oplus$	The logical XOR
$x \leftarrow f(y)$	*x* is the output of the algorithm *f* applied to the input *y*. Unless stated otherwise, *f* is a randomized algorithm, implicitly also using an input of random coins
$x \overset{\$}{\leftarrow} S$	x is drawn from a uniform distribution on the finite set S
$\neg A$	For an event A, the event in which A does not happen
D = { A : B }	The distribution of B given A (where A will specify the distribution from which B is taken). For instance, $D = \{ a \overset{\$}{\leftarrow} S : a \}$ denotes the distribution of a knowing that a is drawn from a uniform distribution on the finite set *S*
$f : X \rightarrow Y$	The function *f* takes input values in the space X and outputs values in Y
$\perp$	Output to an algorithm that indicates that it has failed and returns nothing, except for the indication that it did not terminate correctly

3.3 Abbreviations

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

ABE	Attribute-Based Encryption
CCA	Chosen-Ciphertext Attacks
CDH	Computational Diffie-Hellman
CPA	Chosen-Plaintext Attacks
DEM	Data Encryption Mechanism
DNF	Disjunctive Normal Form
IND	INDistinguishability
KEM	Key Encapsulation Mechanism
KEMAC	KEM with Access Control
LWE	Learning With Errors
NIKE	Non-Interactive Key Exchange

PKE	Public-Key Encryption
PK-IND	Public-Key privacy INDistinguishability
PPT	Probabilistic Polynomial Time
SK-IND	Session-Key privacy INDistinguishability

4 Cryptographic primitives

4.1 Hash functions

Hash functions are used to produce a fixed length random output y from an arbitrary length input x:

- $H(x) \rightarrow y$

Approved hash functions for the purpose of the present document are:

- 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].

4.2 Key Encapsulation Mechanisms (KEMs)

4.2.1 KEMs description

A Key Encapsulation Mechanism KEM is a public-key scheme defined by three algorithms:

- KEM.KeyGen(1^κ) → (pk, sk): on input of a security parameter κ, returns a public key pk and a secret key sk;
- KEM.Enc(pk) → (C, K): on input of the public key pk, generates a session key K, and its encapsulation C, and returns (C, K);
- KEM.Dec(sk, C) → K: on input of the encapsulation C and the secret key sk, returns the session key K encapsulated in C.

Correctness: A KEM is said to achieve correctness if the probability that KEM.Dec(sk,C) is not equal to K is negligible in κ, on the distribution of {(pk,sk) ←KEM.KeyGen(1^κ), (C,K)←KEM.Enc(pk)}.

Security: The main goal of a KEM is to encapsulate a session key K that can only be recovered from the encapsulation C with knowledge of the secret key. This is called Session Key Indistinguishability (SK-IND). One may also wish to protect the privacy of the recipient, meaning that an adversary cannot identify whether a particular encapsulation has been prepared for a specific public key. This is called Public Key Indistinguishability (PK-IND).

The adversary can be modelled as having access to an encapsulation oracle (equivalently, the KEM's public key), in which case the scheme should be resistant to a Chosen Plaintext Attack (CPA security), or having additional access to a decapsulation oracle, in which case the scheme should be resistant to a Chosen Ciphertext Attack (CCA security). The adversary is not allowed to submit any challenge values to the decapsulation oracle.

For a more detailed description of these properties, including the precise security games, see clause A.1.

Approved KEMs for the purpose of the present document are:

- ML-KEM [6].

4.2.2 KEMs with Access Control (KEMAC)

When several users are in a KEM system, a KEM with Access Control (KEMAC) can issue users keys according to a key-policy Y, and encapsulate session keys with respect to an encapsulation-policy X, so that a user with key-policy Y can decapsulate if and only if R(X,Y) evaluates to 1, for a fixed Boolean rule R. Said differently, the access control is defined with respect to the rule R on policies X and Y. For any user-policy Y and encapsulation-policy X, R(X,Y) evaluates to 1 if the user with keys corresponding to the policy Y is allowed to decapsulate an encapsulation made with the policy X; else, R(X,Y) evaluates to 0.

A KEMAC KEMAC is defined with the following algorithms:

- KEMAC.Setup(R,1^κ) → (MPK, MSK): on input of the rule R and the security parameter κ, outputs the global public parameters MPK and the master secret key MSK;
- KEMAC.KeyGen(MSK, Y) → USK: on input of the master secret key MSK and the user-policy Y, outputs a user secret key USK;
- KEMAC.Enc(MPK, X) → (C, K): on input of the global public parameters MPK and the encapsulation -policy X, outputs the session key K and an encapsulation C of K;
- KEMAC.Dec(USK, C) → K: on input of the user secret key USK and the encapsulation C, outputs the key K encapsulated in C.

Correctness. KEMAC is said to achieve correctness with respect to the rule R if for each user-policy Y and encapsulation-policy X such that R(X,Y)=1, given the security parameter κ, the distribution of user keys built with respect to Y, and of encapsulations C of keys K with respect to the policy X is such that except with probability negligible in κ, the decapsulation of C using these user keys is equal to K.

Security. The challenge setup consists of chosen policies X and Y according to R, a random key pair (MPK,MSK) ← KEM.Setup (R,1^κ), a random encapsulation (C,K_0)←KEM.Enc(MPK, X), a random bit *b*, and a random key K_1. For SK-IND-CPA security, given (C, K_b), no adversary, that can only ask keys for user-policies Y' such that R(X,Y')=0, can guess *b* with non-negligible advantage. Note that allowing key queries for a user-policy Y' such that R(X,Y')=1 would allow decapsulating C, and trivially guess *b*. For PK-IND-CCA security, the adversary has additional access to a decapsulation oracle, which provides the encapsulated key K for any encapsulation C' under a key USK, according to any user-policy Y', except for the challenge encapsulation C.

Traceability. An optional feature of a KEMAC is offering traceability in the case of a pirate decoder in which a particular user's key has been embedded. This is a recommended but not required feature, which gives each user distinct keys even if they have common attributes. Several levels of traceability exist. The simplest one is called *white-box tracing*, where from the key extracted in the pirate decoder one can trace back the traitor. In this case, the KeyGen algorithm takes an additional input U, the identity of the user. Then no adversary should be able to design a decapsulating algorithm that uses a key that does not correspond to a user U.

4.2.3 NIKE-based KEM

A Non-Interactive Key Exchange (NIKE) is defined by two algorithms:

- NIKE.KeyGen(1^κ) → (pk, sk): on input of a security parameter κ, returns a public key pk and a secret key sk;
- NIKE.SessionKey (sk, pk') → K: on input of a secret key sk and a public key pk', generates a session key K.

With the two properties:

- **Correctness**: for any (pk_0, sk_0), (pk_1, sk_1) ← NIKE.KeyGen(1^κ),
  NIKE.SessionKey(sk_1, pk_0) = NIKE.SessionKey(sk_0, pk_1);
- **Security**: for (pk_0, sk_0), (pk_1, sk_1) ← NIKE.KeyGen(1^κ), *K* ← NIKE.SessionKey(sk_1, pk_0), given (pk_0, pk_1) only, recovering *K* is hard.

Then one can derive a KEM:

- KEM.KeyGen(1^κ) → (pk, sk): for (pk, sk) ← NIKE.KeyGen(1^κ);

- KEM.Enc(pk) → (C, K): for (pk', sk') ← NIKE.KeyGen(1^κ) and K ← NIKE.SessionKey(sk', pk), then C ← pk';
- KEM.Dec(sk, C) → K': for K' = $\mathcal{H}$(K), with K ← NIKE.SessionKey(sk, pk'), where pk' ← C.

4.2.4 Key-Homomorphic NIKE (KH-NIKE)

A NIKE is called key-homomorphic, if there are two internal group-laws ⊗,⊙ on the secret and the public keys that make them correspond to each other: from (pk_0, sk_0), (pk_1, sk_1) ←NIKE.KeyGen(1^κ), the secret key sk ← sk_0 ⊗ sk_1 corresponds to the public key pk ← pk_0 ⊙ pk_1. So, for any scalar x, the secret key sk' ← x . sk = sk ⊗ ... ⊗ sk corresponds to the public key pk' ← x . pk = pk ⊙ ... ⊙ pk.

Approved KEMs for the purpose of the present document are based on the Diffie-Hellman NIKE in a group (**G**, P, p), where P is a generator of **G**, of prime order p.

The DH algorithms are:

- DH.KeyGen(1^κ) → (pk, sk): for sk $\overset{\$}{\leftarrow}$ $[\![1; p-1]\!]$ and pk ← sk.P;
- DH.SessionKey (sk_1, pk_2) → Q: where Q ← sk_1.pk_2.

The security relies on the Computational Diffie-Hellman (CDH) problem, and it provides key homomorphism.

Approved NIKEs for the purpose of the present document are the above DH scheme on elliptic curves where **G** is instantiated with the P-256, P-384 and P-521 [1] or the Curve25519 and Curve448 [2] elliptic curves.

5 Hybrid Traceable KEMAC (HTKEMAC)

5.1 Description

After the definitions given in previous clauses, this clause specifies the KEM instantiation recommended in the present document, combining hybridization, access control and traceability, from a set Ω of rights (which are combinations of attributes, as shown below) that defines the rule: for any pair (X, Y) of subsets of Ω, $\mathsf{R}(X, Y) = 1$ if and only X and Y have a non-empty intersection. As already explained, X and Y will be the encapsulation-policy and the user-policy, respectively, as lists of rights or equivalently lists of the indices of the rights in the set Ω.

It makes use of a key-homomorphic NIKE (with secret keys in $[\![1; p-1]\!]$) and a KEM with output session keys in $\mathsf{K} = \{0,1\}^{256}$. It will use the following notations:

- $\Omega = \{S_1, \ldots, S_N\}$ is the set of rights, as described in clause 6;
- **G** is a group of prime order p, in which the CDH is assumed to be hard. It will be instantiated with the P-256, P-384 and P-521 [1] or the Curve25519 and Curve448 [2] elliptic curves;
- KEM is a KEM scheme achieving SK-IND-CCA and PK-IND-CCA security. It will be implemented with ML-KEM [6];
- $\mathcal{G}$, $\mathcal{H}$, $\mathcal{J}$ are hash functions, mapping elements to $[\![0; p-1]\!]$, 256-bit strings and 384-bit string respectively where p is the order of group **G**, an elliptic curve field defined by the curve. They will be implemented with SHAKE [3], [5].

The algorithms are:

- HTKEMAC.Setup(Ω, 1^κ) → (MPK, MSK): for **G** a group of prime order p corresponding to the security parameter κ, and P a generator of **G**:
  - the algorithm samples $(H, s), (P_1, s_1), (P_2, s_2) \leftarrow \text{NIKE.KeyGen}(1^\kappa)$;
  - the set of user identities ID, is initialized as an empty set, the tracing secret key is then set to: tsk ← $(s, s_1, s_2, \mathsf{ID})$ and the tracing public key to: tpk ← (P, H, P_1, P_2);