ETSI TR 103 966 V1.1.1 (2024-10)

CYBER Security (CYBER);
Quantum-Safe Cryptography (QSC);
Deployment Considerations for Hybrid Schemes

Reference
DTR/CYBER-QSC-0021

Keywords
cybersecurity, hybrid, 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 Web server (https://ipr.etsi.org/).

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™** and **LTE™** 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 Report (TR) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**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.

1 Scope

The present document explores issues around combining traditional and post-quantum algorithms to construct hybrid cryptographic schemes.

Specifically, the present document examines some of the reasons for proposing and adopting hybrid schemes, both for key establishment and digital signatures; clarifies some of the terminology used to describe hybrid schemes; discusses some of the security, efficiency, and agility trade-offs; highlights some important things to consider when selecting algorithm and parameter combinations; explores some potential deployment and migration issues; and identifies situations where hybrid schemes will need to be deprecated in favour of purely post-quantum algorithms.

The present document does not provide guidance on whether or not to use hybrid schemes.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

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] ETSI TR 103 616: "CYBER; Quantum-safe signatures".

[i.2] ETSI TR 103 692: "CYBER; State management for stateful authentication mechanisms".

[i.3] ETSI TS 103 744: "CYBER; Quantum-Safe Cryptography (QSC); Quantum-safe hybrid key exchanges".

[i.4] ETSI TR 103 823: "CYBER; Quantum-safe public-key encryption and key encapsulation".

[i.5] IETF RFC 5652: "Cryptographic Message Syntax (CMS)".

[i.6] IETF RFC 6090: "Fundamental elliptic curve cryptography algorithms".

[i.7] IETF RFC 8551: "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 message specification".

[i.8] IETF RFC 9370: "Multiple key exchanges in the Internet Key Exchange protocol Version 2 (IKEv2)".

[i.9] IRTF IETF RFC 7748: "Elliptic curves for security".

[i.10] IRTF IETF RFC 8391: "XMSS: eXtended Merkle Signature Scheme".

[i.11] IRTF IETF RFC 8554: "Leighton-Micali hash-based signatures".

[i.12] IRTF IETF RFC 9180: "Hybrid public key encryption".

[i.13] NIST FIPS 140-3: "Security requirements for cryptographic modules".

[i.14] NIST FIPS 186-4: "Digital Signature Standard (DSS)".

[i.15] NIST FIPS 203 (initial public draft): "Module-lattice-based key-encapsulation mechanism standard".

[i.16] NIST FIPS 204 (initial public draft): "Module-lattice-based digital signature standard".

[i.17] NIST FIPS 205 (initial public draft): "Stateless hash-based digital signature standard".

[i.18] NIST SP 800-56A Rev. 3: "Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography".

[i.19] NIST SP 800-56B Rev. 2: "Recommendation for pair-wise key establishment schemes using integer factorization cryptography".

[i.20] NIST: "Post-quantum cryptography FAQs".

[i.21] Recommendation ITU-T X.509: "Information technology - Open Systems Interconnection - The Directory: Public key and attribute certificate frameworks".

[i.22] M.R. Albrecht, et al: "Classic McEliece: conservative code-based cryptography". NIST round 3 post-quantum submission.

[i.23] R. Avanzi et al: "CRYSTALS-Kyber: Algorithm specifications and supporting documentation". NIST round 3 post-quantum submission.

[i.24] J.-P. Aumasson et al: "SPHINCS+: Submission to the NIST post-quantum project". NIST round 3 post-quantum submission.

[i.25] N. Aviram et al: "DROWN: Breaking TLS using SSLv2". USENIX Security 2016.

[i.26] J. Baena et al: "Improving support-minors rank attacks: Applications to GeMSS and Rainbow". CRYPTO 2022.

[i.27] S. Bai et al: "CRYSTALS-Dilithium: Algorithm specifications and supporting documentation". NIST round 3 post-quantum submission.

[i.28] M. Barbosa et al: "X-Wing: The hybrid KEM you've been looking for". IACR ePrint 2024/039.

[i.29] N. Bindel and B. Hale: "A note on hybrid signature schemes". IACR ePrint 2023/423.

[i.30] N. Bindel et al: "Hybrid key encapsulation mechanisms and authenticated key exchange". PQCrypto 2019.

[i.31] N. Bindel et al: "Transitioning to a quantum-resistant public key infrastructure". PQCrypto 2017.

[i.32] W. Beullens: "Breaking Rainbow takes a weekend on a laptop". CRYPTO 2022.

[i.33] D. Bleichenbacher: "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1". CRYPTO 1998.

[i.34] C. Bonnell et al: "A mechanism for encoding differences in paired certificates". IETF Internet-Draft (work in progress), draft-bonnell-lamps-chameleon-certs-03.

[i.35] F. Byszio, K.-D. Wirth and K. Nguyen: "Intelligent composed algorithms". IACR ePrint 2021/813.

[i.36] M. Campagna and A. Petcher: "Security of hybrid key encapsulation". IACR ePrint 2020/1364.

[i.37] W. Castryck and T. Decru: "An efficient key recovery attack on SIDH". EUROCRYPT 2023.

[i.38] D. Connolly, P. Schwabe and B. Westerbaan: "X-Wing: General-purpose hybrid post-quantum KEM". IETF Internet-Draft (work in progress), draft-connolly-cfrg-xwing-kem-01.

[i.39] Y. Dodis and J. Katz: "Chosen-ciphertext security of multiple encryption", TCC 2005.

[i.40] B. Dowling et al: "A cryptographic analysis of the TLS 1.3 handshake protocol candidates". ACM CCS 2015.

[i.41] P.-A. Fouque et al: "Falcon: Fast-Fourier lattice-based compact signatures over NTRU". NIST round 3 post-quantum submission.

[i.42] M. Friedl, J. Mojzis and S. Josefsson: "Secure Shell (SSH) key exchange method using hybrid Streamlined NTRU Prime sntrup761 and X25519 with SHA-512: sntrup761x25519-sha512". IETF Internet-Draft (expired), draft-josefsson-ntruprime-ssh-02.

[i.43] F. Giacon, F. Heuer and B. Poettering: "KEM combiners". PKC 2018.

[i.44] P. Kampanakis, D. Stebila and T. Hansen: "Post-quantum hybrid key exchange in SSH". IETF Internet-Draft (work in progress), draft-kampanakis-curdle-ssh-pq-ke-21.

[i.45] A. Kipnis, J. Patarin and L. Goubin: "Unbalanced oil and vinegar signature schemes". EUROCRYPT 1999.

[i.46] V. Klíma, O. Pokorný and T. Rosa: "Attacking RSA-based sessions in SSL/TLS". CHES 2003.

[i.47] S. Kousidis, J.Roth, F. Strenzke and A. Wussler: "Post-quantum cryptography in OpenPGP". IETF Internet-Draft (work in progress), draft-ietf-openpgp-pqc-02.

[i.48] K. Kwiatkowski and P. Kampanakis: "Post-quantum hybrid ECDHE-Kyber key agreement for TLSv1.3". IETF Internet-Draft (expired), draft-kwiatkowski-tls-ecdhe-kyber-01.

[i.49] A. Langley: "CECPQ1 results". Imperial Violet blog, 28 November 2016.

[i.50] A. Langley: "CECPQ2". Imperial Violet blog, 12 December 2018.

[i.51] Y. Nir: "A hybrid signature method with strong non-separability". IETF Internet-Draft (expired), draft-nir-lamps-altcompsigs-00.

[i.52] M. Ounsworth and J. Gray: "Composite ML-KEM for use in the internet X.509 Public Key Infrastructure and CMS". IETF Internet-Draft (work in progress), draft-ietf-lamps-pq-composite-kem-03.

[i.53] M. Ounsworth, J. Gray and S. Mister: "Composite encryption for use in internet PKI". IETF Internet-Draft (expired), draft-ounsworth-pq-composite-encryption-01.

[i.54] M. Ounsworth, J. Gray, M. Pala and J. Klaussner: "Composite signatures for use in internet PKI". IETF Internet-Draft (work in progress), draft-ietf-lamps-pq-composite-sigs-10.

[i.55] M. Ounsworth, M. Pala and J. Klaussner: "Composite public and private keys for use in internet PKI". IETF Internet-Draft (expired), draft-ounsworth-pq-composite-keys-05.

[i.56] M. Ounsworth, A. Wussler and S. Kousidis: "Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs)". IETF Internet-Draft (work in progress), draft-ounsworth-cfrg-kem-combiners-05.

[i.57] C. Paquin, D. Stebila and G. Tamvada: "Benchmarking post-quantum cryptography in TLS". PQCrypto 2020.

[i.58] A. Petcher and M. Campagna: "Security of hybrid key establishment using concatenation". IACR ePrint 2023/972.

[i.59] B. Poettering and S. Rastikian: "A study of KEM generalizations". SSR 2023.

[i.60] J. Proos and C. Zalka: "Shor's discrete logarithm quantum algorithm for elliptic curves". Quantum Information and Computation 3.4 (2003), 317-344.

[i.61] P.W. Shor: "Algorithms for quantum computation: discrete logarithms and factoring". FOCS, 1994.

[i.62] D. Sikeridis, P. Kampanakis and M. Devetsikiotis: "Post-quantum authentication in TLS 1.3: A performance study". NDSS 2020.

[i.63] D. Sikeridis, P. Kampanakis and M. Devetsikiotis: "Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH". CoNEXT 2020.

[i.64] V. Shoup: "A proposal for an ISO standard for public key encryption". IACR ePrint 2001/112.

[i.65] D. Stebila, S. Fluhrer and S. Gueron: "Hybrid key exchange in TLS 1.3". IETF Internet-Draft (work in progress), draft-ietf-tls-hybrid-design-10.

[i.66] G. Taspoulos et al: "Energy consumption evaluation of post-quantum TLS 1.3 for resource-constrained embedded devices". IACR ePrint 2023/506.

[i.67] B. Westerbaan and D. Stebila: "X25519Kyber768Draft00 hybrid post-quantum key agreement". IETF Internet-Draft (expired), draft-tls-westerbaan-xyber768d00-03.

[i.68] B. Westerbaan and C.A. Wood: "X25519Kyber768Draft00 hybrid post-quantum KEM for HPKE". IETF Internet-Draft (work in progress), draft-westerbaan-cfrg-hpke-xyber768d00-03.

[i.69] R. Zhang, et al: "On the security of multiple encryption or CCA-security + CCA-security = CCA-security?". PKC 2004.

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

active adversary: adversary who can query a decapsulation oracle with chosen ciphertexts or a signing oracle with chosen messages

NOTE 1: An active adversary who is attempting to recover the session key for a target ciphertext is not permitted to query the decapsulation oracle with that ciphertext.

NOTE 2: An active adversary who is attempting to forge a signature value for a target message is not permitted to query the signing oracle with that message.

NOTE 3: The present document assumes that active adversaries only have classical access to the decapsulation or signing oracles. Adversaries who can query decapsulation or signing oracles with inputs in quantum superposition are out of scope.

classical adversary: adversary who can only implement attacks on classical computers

component algorithm: cryptographic algorithm that forms part of a hybrid scheme or hybrid protocol

hybrid interoperability: property that a hybrid scheme or hybrid protocol can be completed successfully provided that at least one component algorithm is supported by both parties

hybrid protocol: protocol that incorporates two or more component algorithms providing the same cryptographic functionality

NOTE 1: The present document only considers hybrid protocols where at least one component is a post-quantum algorithm and at least one is a traditional algorithm. Hybrid protocols that combine two or more post-quantum algorithms and no traditional algorithms are out of scope.

EXAMPLE 1: A protocol that uses a hybrid key establishment scheme for confidentiality and a traditional digital signature algorithm for authentication.

EXAMPLE 2: A protocol that uses a post-quantum key encapsulation mechanism for confidentiality and a hybrid digital signature scheme for authentication.

EXAMPLE 3: A protocol that establishes an initial session key using a traditional key exchange, updates the session key using a post-quantum key encapsulation mechanism, and then performs authentication using a traditional digital signature algorithm.

NOTE 2: A protocol that negotiates the use of either a traditional algorithm or a post-quantum algorithm, but not the use of both algorithms, is not considered to be a hybrid protocol.

hybrid scheme: cryptographic scheme that incorporates two or more component algorithms providing the same cryptographic functionality

NOTE: The present document only considers hybrid schemes where at least one component is a post-quantum algorithm and at least one is a traditional algorithm. Hybrid schemes that combine two or more post-quantum algorithms and no traditional algorithms are out of scope.

EXAMPLE 1: A hybrid key establishment scheme that combines a traditional key exchange and a post-quantum key encapsulation mechanism.

EXAMPLE 2: A hybrid digital signature scheme that combines a traditional digital signature algorithm and a post-quantum digital signature algorithm.

hybrid security: property that a hybrid scheme or hybrid protocol remains secure provided that at least one component algorithm is secure

oracle: functionality that provides an adversary with the output of a cryptographic operation without the adversary needing to know the keys used in the operation

passive adversary: adversary who can only query an encapsulation oracle or a verification oracle

post-quantum algorithm: public-key algorithm believed to be secure against both classical and quantum adversaries

EXAMPLE 1: Key encapsulation mechanisms based on lattices such as the Module-Lattice-based Key Encapsulation Mechanism (ML-KEM) [i.15], or error correcting codes such as Classic McEliece [i.22].

NOTE 1: ML-KEM is derived from the Kyber [i.23] submission to the NIST Post-Quantum Cryptography Standardisation Project

EXAMPLE 2: Digital signature algorithms based on lattices such as the Module-Lattice-based Digital Signature Algorithm (ML-DSA) [i.16], or hash functions such as the Stateless Hash-based Digital Signature Algorithm [i.17].

NOTE 2: ML-DSA is derived from the Dilithium [i.27] submission to the NIST Post-Quantum Cryptography Standardisation Project.

NOTE 3: SLH-DSA is derived from the SPHINCS$^+$ [i.24] submission to the NIST Post-Quantum Cryptography Standardisation Project.

quantum adversary: adversary who can implement attacks on both classical and quantum computers

traditional algorithm: public-key algorithm based on integer factorisation, finite field discrete logarithms, or elliptic curve discrete logarithms

EXAMPLE 1: Key establishment algorithms such as RSA [i.19], Finite-Field Diffie-Hellman (FFDH), and Elliptic Curve Diffie-Hellman (ECDH) [i.18].

EXAMPLE 2: Digital signature algorithms such as RSA, the finite-field Digital Signature Algorithm (DSA), and the Elliptic Curve Digital Signature Algorithm (ECDSA) [i.14].

3.2 Symbols

Void.

3.3 Abbreviations

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

CCA	Chosen-Ciphertext Attack
CMS	Cryptographic Message Syntax
CRQC	Cryptographically Relevant Quantum Computer
DHKEM	Diffie-Hellman Key Encapsulation Mechanism

DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EUF-CMA	Existential Unforgeability under Chosen-Message Attack
FFDH	Finite-Field Diffie-Hellman
FIPS	Federal Information Processing Standard
HPKE	Hybrid Public-Key Encryption
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IND-CCA	Indistinguishability under Chosen-Ciphertext Attack
IND-CPA	Indistinguishability under Chosen-Plaintext Attack
IRTF	Internet Research Task Force
KDF	Key Derivation Function
KDFEM	Key Derivation Function Encapsulation Mechanism
KEM	Key Encapsulation Mechanism
LMS	Leighton-Micali Signature
ML-DSA	Module-Lattice-based Digital Signature Algorithm
ML-KEM	Module-Lattice-based Key Encapsulation Mechanism
OW-CCA	One-Way under Chosen-Ciphertext Attack
OW-CPA	One-Way under Chosen-Plaintext Attack
PKCS	Public-Key Cryptography Standards
PRF	Pseudo-Random Function
RSA	Rivest-Shamir-Adleman
S/MIME	Secure/Multipurpose Internet Mail Extensions
SIKE	Supersingular Isogeny Key Encapsulation
SLH-DSA	Stateless Hash-based Digital Signature Algorithm
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UOV	Unbalanced Oil and Vinegar
XMSS	eXtended Merkle Signature Scheme

4 Introduction

The security of traditional approaches to public-key cryptography, including key establishment and digital signatures, relies on the difficulty of factoring integers or computing discrete logarithms over finite fields or elliptic curves. When suitably parameterised, these algorithms are believed to be hard to break using classical computers. However, they are known to be vulnerable to attacks using quantum computers [i.60] and [i.61].

Although existing quantum computers are not large enough to threaten currently deployed algorithms, the risk associated with the future development of a Cryptographically Relevant Quantum Computer (CRQC) is best mitigated by migrating to post-quantum cryptography.

EXAMPLE 1: In a store-and-decrypt attack, long-lived sensitive information protected by a traditional key establishment algorithm could be intercepted and stored by an adversary, and subsequently decrypted once a CRQC is available.

EXAMPLE 2: In a future forgery attack, a long-lived root of trust protected by a traditional digital signature algorithm might still be trusted and could be exploited by an adversary once a CRQC is available.

Post-quantum algorithms are intended to be secure against both classical and quantum computers. ETSI TR 103 616 [i.1] gives an overview of post-quantum digital signature algorithms and ETSI TR 103 823 [i.4] gives an overview of post-quantum public-key encryption algorithms and key encapsulation mechanisms. (See annexes A and B for background on key encapsulation mechanisms and digital signature algorithms.)

During the migration to post-quantum cryptography there will be situations where it could be desirable to combine existing traditional algorithms with post-quantum algorithms in a hybrid scheme. The main reasons for considering hybrid schemes are:

- to maintain security in the event that vulnerabilities are found in the post-quantum algorithm or its implementation (see clause 5.1);

- to provide backwards compatibility for clients that are not yet able to understand and use post-quantum algorithms (see clause 5.2); and
- to satisfy constraints imposed by protocols or validation requirements (see clause 5.3).

The different reasons for using hybrid schemes mean that different approaches have been proposed for their design and integration into hybrid protocols. This can lead to differences in:

- security guarantees provided by a hybrid scheme (see clause 6.1);
- efficiency of a hybrid protocol (see clause 6.2); and
- complexity of a hybrid protocol or its implementation (see clause 6.3).

There are a variety of issues that need to be considered when deploying hybrid schemes, which depend on the protocol and application. These include:

- algorithm choices for a hybrid scheme (see clause 7.1);
- key management approach for a hybrid protocol (see clause 7.2); and
- migration strategies and forward compatibility (see clause 7.3).

5 Motivations

5.1 Hybrid security

A common reason given for deploying hybrid schemes is the potential security risk involved in deprecating traditional algorithms and moving directly to post-quantum algorithms.

Although post-quantum cryptography has become a significant research topic, there are concerns about the maturity of the cryptanalysis of some algorithms and families of algorithms. There have been high-profile practical attacks against GeMSS [i.26], Rainbow [i.32], and SIKE [i.37] during the third round of the NIST Post-Quantum Cryptography Standardisation Project. It will also take time to build confidence in the assurance of post-quantum implementations. Post-quantum algorithms tend to be more complicated than traditional algorithms and implementation mistakes can be difficult to detect. Effective techniques for protecting post-quantum implementations against side-channel attacks are still being developed.

Well-designed hybrid schemes that remain secure if at least one of their component algorithms is secure can mitigate future vulnerabilities in post-quantum algorithms or their implementations. This approach has already been used in experiments exploring the viability of post-quantum algorithms in protocols such as TLS [i.49] and [i.50]. However, care needs to be taken as hybrid schemes proposed for hybrid interoperability (see clause 5.2) might not provide the same security guarantees as hybrid schemes proposed for hybrid security, while ad-hoc constructions can introduce weaknesses that would not be present when using post-quantum algorithms in a non-hybrid mode (see clause 6.1).

5.2 Hybrid interoperability

Another reason commonly given for deploying hybrid schemes is to provide backwards compatibility and allow a gradual migration to post-quantum algorithms.

In a public or large enterprise system, it might not be possible to update all clients simultaneously to support post-quantum cryptography. During a gradual migration, such systems would potentially need to cope with mixed populations of clients: traditional clients that can only understand and use traditional algorithms, post-quantum aware clients that can understand and use both traditional and post-quantum algorithms, and post-quantum clients that can only understand and use post-quantum algorithms.

Hybrid schemes can provide a backwards compatibility mechanism for protocols that do not permit flexible negotiation of algorithms. For example, Recommendation ITU-T X.509 [i.21] specifies hybrid certificates with non-critical certificate extensions for post-quantum public keys and signature values that could be used by post-quantum aware clients, but that would be ignored by traditional clients. However, hybrid schemes proposed for hybrid security (see clause 5.1) might not provide the same backwards compatibility as hybrid schemes proposed for hybrid interoperability.

5.3 Protocol constraints and validation

Other reasons for retaining traditional algorithms in a hybrid scheme are limitations in the protocol or the need to satisfy accreditation requirements.

The size of the initial key exchange in IKEv2 is limited by the need to avoid packet fragmentation. Most post-quantum key exchanges have public keys that are too large to fit into the available space. The pragmatic solution proposed by IETF RFC 9370 [i.8] keeps a traditional algorithm for the initial key exchange where standard IKE fragmentation mechanisms cannot be used, and allows it to be followed by one or more key exchanges that can be safely fragmented, updating the session key appropriately.

FIPS 140-3 [i.13] validation for cryptographic modules requires the use of approved public-key algorithms. NIST have only approved certain traditional algorithms and have not yet approved any post-quantum algorithms. However, they indicate in their post-quantum FAQs [i.20] that a post-quantum key establishment or digital signature algorithm can be included in an implementation if it is used with an approved traditional algorithm in a FIPS-compliant hybrid mode.

6 Design considerations

6.1 Security

6.1.1 Hybrid security

A common assumption is that the use of hybrid schemes can only increase security. In practice, the security provided by a hybrid scheme depends on the type of scheme and the details of its construction.

Well-designed hybrid schemes can remain secure if at least one component algorithm is secure. It is important to note that traditional and post-quantum algorithms address different security considerations, so the security properties of a hybrid scheme intended to provide hybrid security can still be impacted if one if its component algorithms is insecure.

EXAMPLE 1: The "and-mode" hybrid digital signature schemes proposed in [i.31], [i.35] and [i.54] sign the message using all component digital signature algorithms, and the hybrid signature value is considered valid only if all component signature values are valid (see clause B.3.1).

When instantiated with a single traditional digital signature algorithm and a single post-quantum digital signature algorithm, the hybrid scheme will be secure against both classical and quantum adversaries provided that the post-quantum algorithm is secure.

If the post-quantum algorithm is vulnerable, the hybrid scheme remains secure against classical adversaries because they cannot forge signature values for the traditional algorithm. However, it will be insecure against quantum adversaries and so will no longer protect against future forgery attacks (see clause 4).

NOTE: Including more than one post-quantum algorithm in a hybrid scheme can provide additional mitigation against quantum adversaries at the cost of reducing the efficiency (see clause 6.2) and increasing the complexity (see clause 6.3) of the scheme.

Care needs to be taken when designing hybrid key encapsulation schemes. Naïve constructions can lead to hybrid schemes that are less secure than their component algorithms [i.69] and [i.39]. For example, the "concatenation only" hybrid key encapsulation scheme proposed in [i.35] constructs the hybrid session key as the concatenation of the session keys from the component algorithms. This will not be secure against active adversaries, regardless of the security of the component algorithms: changing one component ciphertext only affects the corresponding part of the session key, not the entire session key (see clause A.3.1 for details).

ETSI TS 103 744 [i.3] includes constructions for hybrid key encapsulation schemes that maintain their security against passive adversaries. Bindel et al [i.30] describe constructions for hybrid key encapsulation schemes that maintain their security against active adversaries.

EXAMPLE 2: The "dual-PRF" hybrid key encapsulation scheme proposed in [i.30] constructs the hybrid session key by concatenating the session keys from the component algorithms and then passing the result through a key derivation function together with the component ciphertexts (see clause A.3.3).

When instantiated with a single traditional key encapsulation mechanism and a single post-quantum key encapsulation mechanism, the hybrid scheme will be secure against both classical and quantum adversaries provided that the post-quantum algorithm is secure.

If the post-quantum algorithm is vulnerable, the hybrid scheme remains secure against classical adversaries because they cannot recover session keys for the traditional algorithm. However, it will be insecure against quantum adversaries and so will no longer protect against store-and-decrypt attacks (see clause 4).

The "dual-PRF" hybrid key encapsulation scheme assumes that all component algorithms are key encapsulation mechanisms. Although traditional Diffie-Hellman key exchanges can be modelled as key encapsulation mechanisms (see, for example, the discussion in [i.65]), they will not be secure against active adversaries: it is possible to modify ciphertexts in a way that causes predictable changes in the session key (see clause A.2.2 for details). For [i.30] to give full security guarantees against active adversaries it is necessary to use a conversion such as DHKEM [i.12].

6.1.2 Hybrid interoperability

Hybrid schemes that are intended to provide backwards compatibility might not possess the same security guarantees as hybrid schemes intended to provide hybrid security. They can be vulnerable to downgrade attacks where an attacker exploits the use of an insecure component algorithm. It is possible for a backwards compatible hybrid scheme to be less secure than using a post-quantum algorithm in a non-hybrid mode.

EXAMPLE: The "or-mode" hybrid digital signature scheme proposed in [i.35] signs the message using all component digital signature algorithms and the hybrid signature value is considered valid if any component signature value is valid.

When instantiated with a single traditional digital signature algorithm and a single post-quantum digital signature algorithm, the hybrid scheme is secure against classical adversaries provided both the post-quantum and traditional algorithms are secure because they cannot forge signature values for either algorithm. However, it will be insecure against quantum adversaries as they can forge signature values for the traditional algorithm and so does not protect against future active attacks (see clause 4).

If the post-quantum algorithm is vulnerable to a classical attack, the hybrid scheme will be insecure against classical adversaries.

NOTE: The only difference between the "or-mode" hybrid digital signature scheme and the "and-mode" hybrid digital signature scheme in clause 6.1.1 is the verification process. Switching an existing client from an "or-mode" verification policy to an "and-mode" verification policy would prevent downgrade attacks on that client without affecting the validity of previously issued hybrid signature values, provided that the client supported both component algorithms. However, this would come at the cost of losing backwards compatibility since the client would no longer accept any hybrid signature values where it did not support one of the component algorithms.

Bindel et al [i.31] investigated a similar backwards compatible approach to S/MIME [i.7] signed messages by including SignedData and SignerInfo objects [i.5] for each component digital signature algorithm. Unfortunately, IETF RFC 5652 [i.5] and IETF RFC 8551 [i.7] do not specify how to validate multiple SignerInfo objects. In their experiments, Bindel et al found that the behaviour of existing S/MIME implementations was not consistent: one implementation would accept the message if it could validate any one of the SignerInfo objects, other implementations would reject the message if they could not validate all SignerInfo objects.

There have been fewer proposals for backwards compatible hybrid key establishment schemes. Although a backwards compatible hybrid encryption scheme was included in [i.53], that work has now been stopped and there is no equivalent in the newer hybrid key encapsulation work [i.52].

6.2 Efficiency

6.2.1 Bandwidth

Post-quantum algorithms typically have larger public keys, ciphertexts, or signature values than traditional algorithms. Migrating from a traditional algorithm to a post-quantum algorithm will have an impact on the bandwidth efficiency of a protocol.

Deploying a hybrid scheme that combines an elliptic curve algorithm and a single post-quantum algorithm can be almost as bandwidth efficient as using the post-quantum algorithm on its own.

EXAMPLE 1: A hybrid key exchange combining the traditional key exchange X25519 [i.9] and the post-quantum key encapsulation mechanism ML-KEM-768 would have a hybrid public key that is less than 3% larger than the public key for ML-KEM-768.

EXAMPLE 2: A hybrid digital signature scheme combining the traditional digital signature algorithm ECDSA-P256 and the post-quantum digital signature algorithm ML-DSA-44 would have a hybrid signature value that is less than 3 % larger than the signature value for ML-DSA-44.

Hybrid schemes incur a proportionally higher overhead for post-quantum algorithms that have small public keys, such as SLH-DSA; small ciphertexts, such as Classic McEliece [i.22]; or small signature values, such as UOV [i.45]. In applications with extremely tight bandwidth constraints this could discourage or even prevent their use.

Deploying a hybrid scheme that includes an RSA-based algorithm or includes more than one post-quantum algorithm can be significantly less bandwidth efficient than using a single post-quantum algorithm on its own.

Hybrid protocols can increase the number of messages passed between parties, either by design to avoid certain protocol constraints or as an unintended consequence of the algorithm negotiation process.

EXAMPLE 3: The hybrid IKEv2 protocol specified in IETF RFC 9370 [i.8] derives the session key from an initial key exchange followed by one or more additional key exchanges. The key exchanges are performed in series with one exchange being completed before the next exchange starts. Each exchange requires a separate round trip.

6.2.2 Computation

The performance of post-quantum cryptography can vary significantly depending on the specific choice of algorithm, platform, and implementation approach. This means that the computational overhead of deploying a hybrid scheme that combines an elliptic curve algorithm and a single post-quantum algorithm can vary.

Hybrid schemes will incur a proportionally higher computational overhead when using a fast post-quantum algorithm.

EXAMPLE: The post-quantum key encapsulation mechanism ML-KEM-512 is about twice as fast as the traditional key exchange ECDH-P256. A hybrid key exchange combining ECDH-P256 and ML-KEM-512 requires three times the computational resources of ML-KEM-512 on its own [i.63].

Deploying a hybrid scheme that includes an RSA-based algorithm or includes more than one post-quantum algorithm can require significantly more computational resources than using a single post-quantum algorithm on its own.

6.2.3 Latency

The latency of protocols such as TLS is often determined by the bandwidth required by the handshake rather than the computational cost of the cryptographic algorithms, particularly on network links with long round-trip times or high rates of packet loss.

EXAMPLE: Under realistic network conditions [i.63], a hybrid TLS handshake combining the traditional key exchange ECDH-P256 and the post-quantum key encapsulation mechanism ML-KEM-512 only increases the latency by 1% compared to using ML-KEM-512 on its own.

NOTE: Replacing the traditional digital signature algorithm with a post-quantum digital signature algorithm can have a more significant impact on the latency of TLS than the key exchange due to the increased signature value and certificate sizes [i.62].

6.2.4 Energy

Energy consumption is an important consideration for battery-powered devices. For protocols such as TLS, energy consumption can depend on both the computational and bandwidth requirements of the algorithms. This will vary for different algorithms. Fast and relatively small post-quantum algorithms can be competitive with traditional algorithms.

EXAMPLE 1: On a constrained client [i.66], the energy consumption of a TLS handshake using the post-quantum key encapsulation mechanism ML-KEM-512 and the post-quantum digital signature algorithm ML-DSA-44 is about the same as a TLS handshake using the traditional key exchange ECDH-P256 and the traditional digital signature algorithm ECDSA-P256.

NOTE: The computational and bandwidth costs of the cryptographic algorithms only account for part of the energy consumption of the TLS handshake. This means that a TLS handshake using a hybrid key exchange that combines ML-KEM-512 and ECDH-P256 would require less than twice the energy of a TLS handshake using ML-KEM-512 by itself.

Slow or relatively large post-quantum algorithms can require significantly more energy than traditional algorithms.

EXAMPLE 2: On a constrained client [i.66], verification for the post-quantum digital signature algorithm SLH-DSA-128f takes about 8 times the energy of verification for the traditional digital signature algorithm ECDSA-P256.

6.3 Complexity

6.3.1 Protocol

Adapting existing protocols to use hybrid schemes involves modifying mechanisms for identifying and negotiating algorithms, and extending structures for public keys, ciphertexts, and signature values. Cryptographic agility can help ease the migration to post-quantum cryptography and mitigate potential vulnerabilities in the algorithms.

EXAMPLE 1: The hybrid TLS key exchange proposed in [i.65] uses a fixed combination of algorithms and parameters that is opaque to the protocol. Each combination is represented by its own algorithm identifier; for example, [i.48] includes an identifier for ECDH-P256 with Kyber768 and [i.67] includes a separate identifier for X25519 with Kyber768. A hybrid key exchange is then negotiated using the existing TLS negotiation mechanism. In particular, the list of supported groups can contain identifiers for traditional, post-quantum, and hybrid key exchanges.

NOTE 1: Backwards compatibility can be achieved in TLS at the cost of additional data in the initial client message or an extra round trip in the handshake. If the client sends a hybrid public key, but the server is not post-quantum aware, the server cannot use the traditional component public key on its own even if the client has indicated support for the traditional component key exchange. Instead, either the client needs to send a traditional public key along with the hybrid public key or the server needs to reject the hybrid public key and explicitly request a traditional public key.

Hybrid protocols that use fixed algorithm combinations offer limited cryptographic agility since the only combinations that can be used are ones that have already been assigned algorithm identifiers. It is not feasible to register all possible combinations of algorithms and parameters, particularly if they include more than two component algorithms. For example, [i.48] includes an identifier for the hybrid key exchange ECDH-P256 with Kyber768, but not for ECDH-P384 with Kyber768 or ECDH-P256 with Kyber1024. Adjusting algorithm combinations in response to improvements in cryptanalysis will likely require new identifiers to be registered.

NOTE 2: The order of the component algorithms in a combination can be important. A hybrid key exchange that combines ECDH-P256 with ML-KEM-768 will not necessarily give the same shared secrets as a hybrid key exchange that combines ML-KEM-768 with ECDH-P256.

However, flexibility increases protocol complexity and could lead to new vulnerabilities being introduced at the protocol level.

EXAMPLE 2: The hybrid IKEv2 protocol proposed in IETF RFC 9370 [i.8] negotiates each component key exchange separately. It allows up to seven additional key exchanges during the establishment of the initial IKE security association or a child security association following authentication. This provides flexibility but adds to the complexity of algorithm negotiation and reduces the efficiency of the protocol (see clause 6.2.1).

NOTE 3: Allowing additional key exchanges prior to authentication increases the risk of denial-of-service attacks, particularly as post-quantum key encapsulation mechanisms can be more resource intensive than traditional key exchanges.

Further, some protocols have been designed to use a traditional Diffie-Hellman key exchange and rely on functionality that might not hold for post-quantum or hybrid key encapsulation. For example, authenticated modes of HPKE [i.12] cannot be instantiated using the post-quantum key encapsulation mechanism ML-KEM or a hybrid key encapsulation scheme that includes ML-KEM as a component. In those cases, the protocol would need to be adapted to use a different mechanism that does not rely on the same functionality.

Similarly, the security analysis for some protocols might assume that a traditional Diffie-Hellman key exchange is being used; for example, the cryptographic analysis of the TLS 1.3 handshake in [i.40]. Post-quantum key encapsulation mechanisms and hybrid key encapsulation schemes can have subtly different security properties and they can be integrated into protocols in subtly different ways. This could require additional analysis to ensure that using a post-quantum or hybrid key exchange provides the same protocol security guarantees.

6.3.2 Implementation

Post-quantum algorithms can be more complicated to implement than traditional algorithms. For example, the digital signature algorithm Falcon [i.41] uses floating-point arithmetic in signature generation. Stateful hash-based digital signature algorithms such as XMSS [i.10] and LMS [i.11] require careful state management by the signer [i.2].

Hybrid schemes and hybrid protocols require implementations to support both traditional and post-quantum algorithms. Most hybrid proposals have been designed so that standalone implementations of the component algorithms can be reused without change (see, however, clause B.3.3). Implementations will need to include mechanisms for combining shared secrets in hybrid key exchanges or additional verification logic in hybrid digital signature schemes. Errors in the implementation of either of these can lead to vulnerabilities that weaken the security of the hybrid scheme.

Deploying hybrid schemes that include more than two component algorithms can increase the complexity of implementation testing and validation due to the number of potential algorithm combinations and code paths.

Traditional and post-quantum algorithms can have different interfaces and security considerations that impact the way they need to be handled by the implementation. This applies similarly to hybrid schemes.

EXAMPLE 1: In the traditional digital signature algorithm ECDSA , message hashing can be separated from the rest of the signing process. If a cryptographic security module is being used to protect the ECDSA private key, then only the hash of the message needs to be sent to the module for signing.

In the post-quantum digital signature algorithm SLH-DSA, message hashing includes a random value generated by the signer and the security of the algorithm depends on this value being unpredictable. If a cryptographic security module is being used to protect the SLH-DSA private key, then the full message needs to be sent to the module for signing.

NOTE: It is possible to avoid sending the full message to the module by pre-hashing the message before signing and verification; for example, as is done for the CMS signed-data content type [i.5]. However, the hash function used for pre-hashing needs to be chosen carefully so that it does not weaken the security of the hybrid digital signature scheme.

EXAMPLE 2: In a protocol with a traditional Diffie-Hellman key exchange such as TLS 1.3, the server can cache their public key for a short period of time and use it in key exchanges with several different clients. This will not preserve forward secrecy, but is otherwise secure provided that the server validates the client public keys.

With a post-quantum key encapsulation mechanism or a hybrid key encapsulation scheme, the server sends a ciphertext that depends on the client public key so caching is not possible. Further, if a server attempts to reuse the same random seed when computing ciphertexts for two different client public keys, this can reveal enough information for a passive adversary to recover the session keys.

7 Deployment considerations

7.1 Algorithm selection

7.1.1 Number of components

It is important to consider the number of component algorithms to include when planning to deploy hybrid schemes.

The simplest approach is to combine a single traditional algorithm with a single post-quantum algorithm. This can minimise bandwidth and computational overheads, provide backwards compatibility in schemes intended for hybrid interoperability, and mitigate classical weaknesses in the post-quantum algorithm in schemes intended for hybrid security.

A more conservative approach would be to combine a single traditional algorithm with two or more post-quantum algorithms. This can provide additional flexibility in schemes intended for hybrid interoperability, or additional reassurance in schemes intended for hybrid security. However, each additional post-quantum algorithm will significantly increase the bandwidth and computational overheads.

Where there is enough confidence in the security of a post-quantum algorithm it might not be necessary to deploy it in a hybrid scheme. For example, hash-based digital signature algorithms such as SLH-DSA are well enough understood that they could be used without a traditional digital signature algorithm in some applications.

7.1.2 Choice of components

It is important to consider the choice of component algorithms carefully.

In general, it is preferable to use component algorithms that have been selected for standardisation following a public review process, as they are likely to have received greater scrutiny. Component algorithms that are less mature and have not had the same level of analysis will not increase confidence in the security of the hybrid scheme. In some cases, weak component algorithms can compromise the security of the hybrid scheme (see clause 6.1.2).

Including two or more component algorithms constructed from the same underlying mathematical problem is unlikely to increase confidence in the security of the hybrid scheme, as a sufficiently general attack could apply to all algorithms constructed from that problem.

EXAMPLE: An earlier version of the "composite keys" proposal [i.55] listed explicit algorithm identifiers for hybrid digital signature schemes that combined a traditional digital signature algorithm, the post-quantum digital signature algorithm Dilithium-5, and the post-quantum digital signature algorithm Falcon-1024. Dilithium and Falcon both use structured lattices so could both be affected by an attack that exploits the structure.

In particular, including more than one traditional algorithm in a hybrid scheme will provide minimal security benefit because they are all vulnerable to a quantum adversary.

If there are concerns about the precise level of security provided by a post-quantum algorithm, these could be mitigated by increasing the size of its parameters; for example, by deploying a hybrid that combines ECDH-P256 with ML-KEM-768 or ML-KEM-1024 rather than ML-KEM-512.

7.2 Key management

7.2.1 Component key reuse

When generating a hybrid key, it is essential that the component keys are generated independently from each other. If the component keys are related - for example, if they are derived from the same seed - then the compromise of one could affect the security of the others. However, some proposals allow multiple different hybrid keys to contain the same component key or allow a component of a hybrid key to be used in a non-hybrid mode.

EXAMPLE 1: The TLS hybrid key exchange proposed in [i.65] can provide backwards compatibility by sending the server the client public keys for both the traditional key exchange and the hybrid key exchange at the same time (see clause 6.3.1). If the traditional key exchange is one of the component algorithms for the hybrid key exchange, the client might want to reuse the traditional public key as a component of the hybrid public key to improve efficiency. This type of component key reuse is permitted by [i.65].

EXAMPLE 2: If an organisation already has a long-lived traditional certificate that is widely trusted, they might want to deploy a hybrid certificate [i.21] that includes a new post-quantum public key along with the existing traditional public key for backwards compatibility.

There are risks to allowing the same component key to be used in different contexts. In a hybrid digital signature scheme, the components of the hybrid signature value are often valid signature values for the component digital signature algorithms. In a hybrid key encapsulation scheme, the components of the hybrid ciphertext are often valid ciphertexts for the component key encapsulation algorithms. If the hybrid scheme does not tightly bind together the component algorithms, this could lead to active attacks that use signing or decapsulation oracles (see clause B.2). At the very least, it can invalidate the assumptions made by any security proofs.

7.2.2 Component key compromise

Hybrid schemes are typically designed with the expectation that the traditional component key will eventually be compromised by a quantum adversary. Post-quantum component keys could be compromised through a weakness in the algorithm, vulnerabilities in the implementation, or other misuse of the keys.

Hybrid security is intended to ensure that the compromise of a single component key is not enough to break the hybrid scheme; all component keys need to be compromised to break the hybrid security. However, compromise of a component key can still impact the security of the scheme (see clause 6.1.1) and it might be necessary to revoke the hybrid key, depending on the use case.

EXAMPLE: If the compromise of a post-quantum component key in a hybrid key encapsulation scheme means that the hybrid key is now vulnerable to the store-and-decrypt threat, then the hybrid scheme can no longer be safely used to protect long-lived sensitive information.

Hybrid interoperability can often mean that the compromise of a single component key is enough to break the hybrid scheme (see clause 6.1.2). In that case it will be necessary to revoke the hybrid key.

Key management policy will dictate whether the revocation of a hybrid key means that all of its component keys are also considered compromised [i.55]. Allowing the same component key to be used in multiple hybrid keys can complicate this. Revoking a single hybrid key could lead to the revocation of further hybrid keys if they share a common component key.

7.3 Migration and forwards compatibility

The approach to migration will depend on a number of factors.

If a protocol does not allow on-line algorithm negotiation, it might be necessary to deploy post-quantum algorithms in a hybrid scheme that provides hybrid interoperability. If a protocol will be used to protect long-lived sensitive information or involves a long-lived root of trust, it will be important to provide hybrid security. Achieving both hybrid interoperability and hybrid security in the same protocol can be difficult. In some cases, the requirements for confidentiality and for authentication can be different.

EXAMPLE 1: The TLS hybrid key exchange specified in [i.65] is intended to provide hybrid security as a TLS session could be used to protect long-lived sensitive data. Hybrid interoperability can be achieved through the existing TLS key exchange negotiation mechanism and is protected against downgrade attacks by the authentication.

However, [i.57] argues that hybrid security is less of a concern for TLS authentication as it only needs to be secure at the time the TLS session is being established. Instead, the existing TLS digital signature algorithm negotiation mechanism can be used to negotiate independent traditional or post-quantum certificate chains.

EXAMPLE 2: The hybrid key encapsulation schemes specified in [i.52] are intended to provide hybrid security for S/MIME as message encryption could be used to protect the confidentiality of long-lived sensitive data. Algorithm negotiation is not possible, but hybrid interoperability can be achieved by having both traditional and hybrid encryption certificates for the recipient. A sender that is post-quantum aware can use the hybrid encryption certificate, otherwise they can use the traditional encryption certificate.

S/MIME authentication could be used to provide non-repudiation of long-lived data, even when confidentiality does not need to be protected. Hybrid interoperability is important in this case as the sender might not know whether the recipient is post-quantum aware. The approach from [i.31] can be applied to sign the message with both a traditional and a post-quantum digital signature algorithm. A recipient that is post-quantum aware can verify both signature values, otherwise they can verify the traditional signature value.

EXAMPLE 3: Firmware signing can involve a long-lived root of trust that is difficult to update so hybrid security is critical, but hybrid interoperability is less of a concern as the firmware developer will likely know whether a device is post-quantum aware. It is straightforward to use the "concatenation only" hybrid digital signature scheme [i.54] in this case (see clause A.3.1). Alternatively, a stateful hash-based digital signature algorithm such as LMS [i.11] or XMSS [i.10] can be used if safe state management is possible.

In the future, there will be enough confidence in the security of post-quantum algorithms and their implementations that hybrid schemes will no longer be necessary for hybrid security. Keeping the traditional component algorithms will incur overheads without any security benefit and could lead to downgrade attacks on some hybrid schemes once a CRQC is available. Forward compatibility is the notion that protocols can migrate away from hybrid schemes to use post-quantum algorithms by themselves.

EXAMPLE 4: The TLS key exchange negotiation can be used to negotiate purely post-quantum algorithms. If a post-quantum aware client sends a post-quantum public key but the server only supports hybrid schemes, then the server can reject the post-quantum public key and explicitly request a hybrid public key. Downgrade protection is provided by using a post-quantum certificate chain for the TLS authentication.

EXAMPLE 5: Post-quantum algorithms are currently too large for the initial key exchange in IKEv2 so the hybrid protocol in IETF RFC 9370 [i.8] will likely need to keep a traditional key exchange algorithm until the fragmentation issue is resolved.

NOTE: In some situations, there could be a reason to continue to use hybrid schemes that only involve post-quantum component algorithms. This is out of scope of the present document.

8 Conclusions

Deploying post-quantum cryptography alongside existing traditional algorithms in a hybrid scheme or hybrid protocol can mitigate potential vulnerabilities in post-quantum implementations or provide backwards compatibility during a post-quantum migration. Bandwidth, computation, and latency overheads can be minimised by pairing a post-quantum algorithm with a traditional elliptic curve algorithm, but the complexity of protocols, implementations, and key management will increase.

Hybrid schemes and hybrid protocols need to be designed carefully. Post-quantum algorithms can have subtly different functionality and security properties compared to existing traditional algorithms. It is important to understand the impact of using post-quantum algorithms or hybrid schemes on the security analysis of a protocol and to understand the security guarantees that remain if one of the component algorithms is broken. Inappropriate hybrid schemes can be less secure than using post-quantum algorithms in a non-hybrid mode.

Hybrid protocols can provide both hybrid security and hybrid interoperability through algorithm negotiation, but this needs to be protected against downgrade attacks. The requirements will depend on the specific protocol or use case and might be different for confidentiality and authentication. The choice of post-quantum algorithm or hybrid algorithm combination also needs careful consideration. It is not advisable to deploy post-quantum algorithms that have not been through a standardisation process or have not received sufficient analysis, even when used in a hybrid scheme or hybrid protocol.

Eventually, there will be enough confidence in the security of post-quantum algorithms and their implementations that hybrid schemes and hybrid protocols will no longer be necessary. Migrating to purely post-quantum algorithms and protocols at this point will avoid the overheads and additional complexity associated with hybrids and will minimise any potential risk from continuing to use traditional component algorithms that are known to be vulnerable to quantum adversaries.

Annex A:
Key encapsulation mechanisms

A.1 Introduction

A Key Encapsulation Mechanism (KEM) consists of a triple of functions:

- **Key generation:** Returns a new public and private key pair.
- **Encapsulation:** Takes a public key as input and returns a randomly selected session key and a ciphertext that encapsulates the session key.
- **Decapsulation:** Takes a private key and a ciphertext as input and returns a session key.

NOTE 1: Some KEMs can have decapsulation failures where decapsulation either returns an error or a session key that does not match the encapsulated session key.

In practice, KEMs usually involve two parties: a sender and recipient. The sender encapsulates a session key for the recipient, using the recipient's public key. KEMs differ from public-key encryption schemes in that the session key is an output of encapsulation rather than an input to encryption.

The two main security notions for KEMs are Indistinguishability under Chosen-Plaintext Attack (IND-CPA) and Indistinguishability under Chosen-Ciphertext Attack (IND-CCA). Informally, these are defined as follows:

- **IND-CPA security:** Given a public key P, ciphertext C, and session key K, the adversary tries to determine whether K was encapsulated by C or was chosen randomly.
- **IND-CCA security:** Given a public key P, ciphertext C, and session key K, the adversary tries to determine whether K was encapsulated by C or was chosen randomly, when they are also given access to a decapsulation oracle that returns the session key encapsulated by any other ciphertext $C' \neq C$.

NOTE 2: This version of Indistinguishability under Chosen-Ciphertext Attack is usually referred to as IND-CCA2 security in the academic literature. A more restricted version, IND-CCA1, only allows the adversary to query the decapsulation oracle before they have been given the challenge ciphertext C.

IND-CPA security roughly corresponds to a passive adversary who can only observe ciphertexts sent from an honest sender to the recipient. IND-CCA security roughly corresponds to an active adversary who can construct malformed ciphertexts and observe the recipient's responses in order to learn information about the session key.

Two related security notions are One-Way under Chosen-Plaintext Attack (OW-CPA) and One-Way under Chosen-Ciphertext Attack (OW-CCA), Informally, these are defined as follows:

- **OW-CPA security:** Given a public key P and ciphertext C, the adversary tries to recover the session key K encapsulated by C.
- **OW-CCA security:** Given a public key P and ciphertext C, the adversary tries to recover the session key K encapsulated by C, when they are also given access to a decapsulation oracle that returns the session key encapsulated by any other ciphertext $C' \neq C$.

NOTE 3: IND-CPA security implies OW-CPA security since an adversary cannot tell when they have recovered the correct session key for the given ciphertext. Similarly, IND-CCA security implies OW-CCA security.

EXAMPLE: Bleichenbacher's attack on RSA encryption with PKCS#1 v1.5 padding [i.33] exploits the malleability of RSA decryption. By carefully modifying the ciphertext, an adversary can use the result of the padding checks to learn information about, and eventually recover, the original message. This directly led to vulnerabilities in RSA-based sessions for early versions of SSL and TLS 1.0 [i.46], and later resurfaced in the cross-protocol attack DROWN on TLS 1.2 [i.25].

A.2 Traditional KEMs

A.2.1 RSA-KEM

Basic RSA encryption can be converted into an IND-CCA secure KEM (see [i.64] or section 7.2.1 in [i.19]) using a key derivation function KDF:

- **Key generation:**
  - Choose primes p, q and compute $N = pq$.
  - Choose an encryption exponent e with $\gcd(e, (p-1)(q-1)) = 1$.
  - Compute the decryption exponent d such that $ed \equiv 1 \bmod (p-1)(q-1)$.
  - Return the public key (N, e) and private key (N, d).
- **Encapsulation:** Given the public key (N, e):
  - Choose a random value $x \in \{1, \ldots, N-1\}$.
  - Compute $C \equiv x^e \bmod N$.
  - Return the ciphertext C and session key $K = \mathrm{KDF}(x)$.
- **Decapsulation:** Given the private key (N, d) and ciphertext C:
  - Compute $x \equiv C^d \bmod N$.
  - Return the session key $K = \mathrm{KDF}(x)$

NOTE: The use of a KDF is necessary for the IND-CPA security of RSA-KEM. If the adversary is given the ciphertext C and a value x instead of $K = \mathrm{KDF}(x)$, they could determine whether C encapsulates x by trial encrypting $C' \equiv x^e \bmod N$ and checking whether $C' = C$.

A.2.2 DHKEM

Let G be the generator of a group $\mathcal{G}$ of prime order p, written additively. The Diffie-Hellman key agreement in $\mathcal{G}$ can be converted into an IND-CCA secure KEM [i.12] using a key derivation function KDF:

- **Key generation:**
  - Choose a random value $x \in \{1, \ldots, p-1\}$.
  - Compute $P = [x]G$.
  - Return the public key P and private key x.
- **Encapsulation:** Given the public key P:
  - Choose a random value $y \in \{1, \ldots, p-1\}$.
  - Compute $Q = [y]G$.
  - Compute $R = [y]P$.
  - Return the ciphertext Q and session key $K = \mathrm{KDF}(R \,||\, Q)$.
- **Decapsulation:** Given the private key x and ciphertext Q:
  - Compute $R = [x]Q$.
  - Return the session key $K = \mathrm{KDF}(R \,||\, Q)$.

NOTE 1: The use of a KDF is necessary for the OW-CCA and IND-CCA security of DHKEM. If the adversary is given the ciphertext Q and can query a decapsulation oracle with ciphertexts $Q' \neq Q$ to obtain the value $R' = [x]Q'$ instead of $K' = KDF(R' \,||\, Q')$, then querying the oracle with the ciphertext $Q' = -Q$ would give $R' = [x]Q' = [-x]Q = -R$ which allows recovery of R.

NOTE 2: The inclusion of the ciphertext Q in the KDF input is necessary to protect against instantiations of Diffie-Hellman where it is possible for an adversary to find a different ciphertext Q' that gives the same shared secret R as Q since, in this case, querying the decapsulation oracle with the ciphertext Q' would give the same session key $K = \mathrm{KDF}(R)$ instead of $K' = \mathrm{KDF}(R \,||\, Q')$.

EXAMPLE 1: X25519 [i.9] is not OW-CCA or IND-CCA secure if the KDF does not include the ciphertext Q as an input because it permits the use of public keys that lie outside the cryptographic subgroup $\mathcal{G}$. Given a ciphertext Q, an adversary can choose the different ciphertext $Q' = Q + T$, for a point T of small order, and the shared secret will be $R' = [x]Q' = [x]Q + [x]T = [x]Q = R$ due to private key clamping.

EXAMPLE 2: Compact representation ECDH-P256 [i.6] is not OW-CCA or IND-CCA if the KDF does not include the ciphertext Q as an input because it only uses the x-coordinate of R as the shared secret rather than the full point. Given a ciphertext Q, an adversary can choose the different ciphertext $Q' = -Q$ and $R' = [x]Q' = [-x]Q = -R$ will have the same x-coordinate as R.

A.3 Hybrid KEM examples

A.3.1 Concatenation only

The simplest approach to constructing a hybrid KEM from a pair of component KEMs, KEM_1 and KEM_2, is concatenation:

- **Key generation:**
  - Call $\mathrm{KEM}_1.\mathrm{KeyGen}()$ to obtain the component public key P_1 and component private key S_1.
  - Call $\mathrm{KEM}_2.\mathrm{KeyGen}()$ to obtain the component public key P_2 and component private key S_2.
  - Return the hybrid public key $P = P_1 \,||\, P_2$ and the hybrid private key $S = S_1 \,||\, S_2$.
- **Encapsulation:** Given the hybrid public key P:
  - Call $\mathrm{KEM}_1.\mathrm{Encaps}(P_1)$ to obtain the component ciphertext C_1 and component session key K_1.
  - Call $\mathrm{KEM}_2.\mathrm{Encaps}(P_2)$ to obtain the component ciphertext C_2 and component session key K_2.
  - Return the hybrid ciphertext $C = C_1 \,||\, C_2$ and final session key $K = K_1 \,||\, K_2$.
- **Decapsulation:** Given the hybrid private key S and hybrid ciphertext C:
  - Call $\mathrm{KEM}_1.\mathrm{Decaps}(S_1, C)$ to obtain the component session key K_1.
  - Call $\mathrm{KEM}_2.\mathrm{Decaps}(S_2, C)$ to obtain the component session key K_2.
  - Return the final session key $K = K_1 \,||\, K_2$.

NOTE 1: The hybrid KEM will be IND-CPA secure if both component KEMs are IND-CPA secure. This follows from the observation that the final session key $K = K_1 \,||\, K_2$ will be indistinguishable from $K' = K_1' \,||\, K_2$ for a random K_1' by the IND-CPA security of KEM_1 and $K' = K_1' \,||\, K_2$ will be indistinguishable from $K'' = K_1' \,||\, K_2'$ for a random K_2' by the IND-CPA security of KEM_2.

NOTE 2: The hybrid KEM will not be IND-CPA secure if one of the component KEMs is not IND-CPA secure. Suppose that KEM_1 is not IND-CPA secure. With high probability, the adversary can determine whether the session key $K = K_1 \,||\, K_2$ was encapsulated by the hybrid ciphertext $C = C_1 \,||\, C_2$ by checking whether the corresponding component session key K_1 was encapsulated by the component ciphertext C_1.

In particular, the hybrid KEM will not be IND-CPA secure once a CRQC is available since a quantum adversary will be able to break the IND-CPA security of the traditional component KEM.

NOTE 3: The hybrid KEM will not be IND-CCA secure even if both component KEMs are IND-CCA secure. The adversary can choose a different component ciphertext $C_1' \neq C_1$ for KEM_1 and request the decapsulation $K' = K_1' \,||\, K_2'$ of the new hybrid ciphertext $C' = C_1' \,||\, C_2$. With high probability, the adversary can then determine whether the session key $K = K_1 \,||\, K_2$ was encapsulated by the hybrid ciphertext $C = C_1 \,||\, C_2$ by checking whether $K_2' = K_2$.

EXAMPLE 1: The "concatenation only" hybrid KEM has been proposed for use in TLS [i.65], [i.48], [i.67]. Although the hybrid KEM is not IND-CCA secure by itself, the TLS key schedule includes the hybrid ciphertext when deriving the traffic keys so these will still be indistinguishable from random provided that at least one of the component KEMs is OW-CCA secure (see clause A.3.3).

EXAMPLE 2: The "concatenation only" hybrid KEM has been proposed for use in HPKE [i.68]. In contrast to TLS, the HPKE key schedule does not include the hybrid ciphertext when deriving the encryption keys so the general construction will not be IND-CCA secure. Nevertheless, [i.68] claims that it will be secure when used with DHKEM(X25519) and Kyber768.

A.3.2 Concatenation with simple KDF

The IND-CPA and IND-CCA attacks on the "concatenation only" hybrid KEM in clause A.3.1 can be prevented by using a key derivation function KDF to produce the final session key:

- **Key generation:**
  - Call $\mathrm{KEM}_1.\mathrm{KeyGen}()$ to obtain the component public key P_1 and component private key S_1.
  - Call $\mathrm{KEM}_2.\mathrm{KeyGen}()$ to obtain the component public key P_2 and component private key S_2.
  - Return the hybrid public key $P = P_1 \,||\, P_2$ and the hybrid private key $S = S_1 \,||\, S_2$.
- **Encapsulation:** Given the hybrid public key P:
  - Call $\mathrm{KEM}_1.\mathrm{Encaps}(P_1)$ to obtain the component ciphertext C_1 and component session key K_1.
  - Call $\mathrm{KEM}_2.\mathrm{Encaps}(P_2)$ to obtain the component ciphertext C_2 and component session key K_2.
  - Return the hybrid ciphertext $C = C_1 \,||\, C_2$ and final session key $K = \mathrm{KDF}(K_1 \,||\, K_2)$.
- **Decapsulation:** Given the hybrid private key S and hybrid ciphertext C:
  - Call $\mathrm{KEM}_1.\mathrm{Decaps}(S_1, C)$ to obtain the component session key K_1.
  - Call $\mathrm{KEM}_2.\mathrm{Decaps}(S_2, C)$ to obtain the component session key K_2.
  - Return the final session key $K = \mathrm{KDF}(K_1 \,||\, K_2)$.

NOTE 1: The hybrid KEM will be IND-CPA secure if at least one of the component KEMs is OW-CPA secure, when KDF is modelled as a random oracle. This follows from the same argument as Theorem 1 in [i.36].

NOTE 2: While it is plausible that the hybrid KEM might be IND-CCA secure if both of the component KEMs are OW-CPA or IND-CCA secure, there is currently no proof of this in the academic literature.

NOTE 3: There are examples where the hybrid KEM is not IND-CCA secure even though one of the component KEMs is IND-CCA secure. Suppose that KEM_1 is IND-CCA secure, but that given any ciphertext C_2 for KEM_2 the adversary can find a different ciphertext C_2' that encapsulates the same session key K_2. In this case, the adversary can determine whether the hybrid ciphertext $C = C_1 \,||\, C_2$ encapsulates a final session key K by querying the decapsulation oracle with the different hybrid ciphertext $C = C_1 \,||\, C_2'$ to obtain the key $K' = \mathrm{KDF}(K_1 \,||\, K_2)$ and checking whether $K' = K$.

EXAMPLE 1: The "concatenation with simple KDF" hybrid KEM will not be IND-CCA secure if the traditional component KEM is X25519 or compact representation ECDH-P256 without applying the full DHKEM conversion (see clause A.2.2).

EXAMPLE 2: The "concatenation with simple KDF" hybrid KEM has been proposed for use in SSH [i.42], [i.44]. These specify either X25519 or compact representation ECDH for the traditional component key, but the SSH key schedule includes the hybrid ciphertext when deriving the encryption and integrity keys so these will still be indistinguishable from random (see clause A.3.3).

A.3.3 Concatenation with full KDF

Adjusting the key derivation function slightly can prevent the IND-CCA attack on the "concatenation with simple KDF" hybrid KEM from clause A.3.2:

- **Key generation:**
  - Call $\mathrm{KEM}_1.\mathrm{KeyGen}()$ to obtain the component public key P_1 and component private key S_1.
  - Call $\mathrm{KEM}_2.\mathrm{KeyGen}()$ to obtain the component public key P_2 and component private key S_2.
  - Return the hybrid public key $P = P_1 \,||\, P_2$ and the hybrid private key $S = S_1 \,||\, S_2$.
- **Encapsulation:** Given the hybrid public key P:
  - Call $\mathrm{KEM}_1.\mathrm{Encaps}(P_1)$ to obtain the component ciphertext C_1 and component session key K_1.
  - Call $\mathrm{KEM}_2.\mathrm{Encaps}(P_2)$ to obtain the component ciphertext C_2 and component session key K_2.
  - Return the hybrid ciphertext $C = C_1 \,||\, C_2$ and final session key $K = \mathrm{KDF}(K_1 \,||\, K_2 \,||\, C_1 \,||\, C_2)$.
- **Decapsulation:** Given the hybrid private key S and hybrid ciphertext C:
  - Call $\mathrm{KEM}_1.\mathrm{Decaps}(S_1, C)$ to obtain the component session key K_1.
  - Call $\mathrm{KEM}_2.\mathrm{Decaps}(S_2, C)$ to obtain the component session key K_2.
  - Return the final session key $K = \mathrm{KDF}(K_1 \,||\, K_2 \,||\, C_1 \,||\, C_2)$.

NOTE 1: The hybrid KEM will be IND-CCA secure if at least one of the component KEMs is IND-CCA secure and KDF satisfies certain pseudo-random function security assumptions [i.43].

NOTE 2: Traditional Diffie-Hellman key exchanges are not IND-CCA secure KEMs when used without a DHKEM-like conversion (see clause A.2.2).

EXAMPLE 1: The "concatenation with full KDF" hybrid KEM is proposed as a general construction in [i.56] and for use in OpenPGP in [i.47].

EXAMPLE 2: The CatKDF hybrid KEM specified in ETSI TS 103 744 [i.3] is a more general version of the "concatenation with full KDF" hybrid KEM. This is flexible enough to cover the use of "concatenation only" hybrid KEMs in the TLS and SSH key schedules [i.58] and has been shown to be IND-CCA secure if at least one of the component KEMs is OW-CCA secure, when KDF is modelled as a random oracle [i.58].

EXAMPLE 3: The X-Wing hybrid KEM proposed in [i.38] is an optimised version of the "concatenation with full KDF" hybrid KEM for the specific combination of X25519 with ML-KEM-768. This keeps the X25519 ciphertext as an input to the KDF, but omits the ML-KEM-768 ciphertext. It is shown to be IND-CCA secure if ML-KEM-768 is IND-CCA secure or if both ML-KEM-768 is Ciphertext Collision Resistant and X25519 satisfies the Strong Diffie-Hellman assumption, when KDF is modelled as a random oracle [i.28].

EXAMPLE 4: An alternative approach has been proposed by Poettering and Rastikian in [i.59]. They consider a generalised form of KEM, called a KDF Encapsulation Mechanism (KDFEM), which splits out the final KDF step; that is, encapsulation produces a ciphertext C, handle HD, and intermediate state ST, and the session key is derived via a separate evaluation function as $K = \mathrm{KDF}(ST \,||\, HD)$. In the hybrid KEM, the final session key is derived as $K = \mathrm{KDF}(ST_1 \,||\, HD_2) \oplus \mathrm{KDF}(ST_2 \,||\, HD_1)$ where the two handles have been switched. They show that this is secure against active adversaries if one of the component KDFEMs is secure against active adversaries and the other is correct.

Annex B: Digital signature algorithms

B.1 Introduction

A digital signature algorithm consists of a triple of functions:

- **Key generation:** Returns a new public and private key pair.
- **Signature generation:** Takes a private key and message as inputs, and returns a signature value.
- **Verification:** Takes a public key, message and signature value as inputs, and returns either accept or reject.

Digital signature algorithms usually involve two parties: a signer and a verifier. The signer produces a signature value for a message using their private key. The verifier checks that the signature value for the message is valid using the signer's public key.

The main security notion for digital signature algorithms is Existential Unforgeability under Chosen-Message Attack (EUF-CMA).

- **EUF-CMA security:** Given a public key P and access to a signing oracle that will return a signature value σ for any message M, the adversary tries to construct a valid signature value σ' for a message M' that was not included in a request to the signing oracle.

B.2 Separability

In addition to EUF-CMA security, there are also several notions of separability for hybrid digital signature schemes [i.29]; that is, the ability to extract a valid signature value for one of the components from the hybrid signature value:

- **Separable:** Given a hybrid signature value, an adversary can produce a signature value that will be accepted by the verification algorithm for one of the component algorithms and contains no evidence that it was derived from a hybrid signature value.
- **Weakly non-separable:** Given a hybrid signature value, an adversary cannot produce a signature value that will be accepted by the verification algorithm for one of the component algorithms without it containing some evidence that it was derived from a hybrid signature value.
- **Strongly non-separable:** Given a hybrid signature value, an adversary cannot produce a signature value that will be accepted by the verification algorithm for one of the component algorithms.

Non-separable hybrid signature schemes reduce the risk that an adversary can manipulate the verifier into accepting part of a hybrid signature value as a purely traditional or purely post-quantum signature value. Weak non-separability allows the verifier to detect when they are presented with part of a hybrid signature value. Strong non-separability prevents the verifier from accepting any partial signature value as valid.

EXAMPLE 1: The chameleon certificate proposal [i.34] gives a method for including a second delta certificate in a non-critical extension of a base X.509 certificate. The outer base signature value is computed over the base certificate which contains the delta certificate extension, but the inner delta signature value is only computed over the delta certificate. This means that the inner delta certificate can be fully separated from the base certificate since it will contain no evidence of the existence of the base certificate.

EXAMPLE 2: Recommendation ITU-T X.509 [i.21] hybrid certificates give a method of including a second alternative signature value in a non-critical extension of a base X.509 certificate. The outer base signature value and the inner alternative signature value are computed over messages that contain identifiers for both digital signature algorithms. This provides weak non-separability since it is not possible to remove the identifier for one of the digital signature algorithms and still have a certificate that can be verified using the other digital signature algorithm.

NOTE: Chameleon certificates [i.34] and Recommendation ITU-T X.509 [i.21] hybrid certificates are both designed to provide backwards compatibility since they can be verified by a relying party that only understands the outer digital signature algorithm and ignores the inner delta certificate or alternative signature value. This would not be possible with a strongly non-separable scheme.

B.3 Hybrid signature examples

B.3.1 Concatenation only

The simplest approach to constructing a hybrid digital signature scheme from a pair of component digital signature algorithms, Sig_1 and Sig_2, is concatenation:

- **Key generation:**
  - Call $\mathrm{Sig}_1.\,\mathrm{KeyGen}()$ to obtain the component public key P_1 and component private key S_1.
  - Call $\mathrm{Sig}_2.\,\mathrm{KeyGen}()$ to obtain the component public key P_2 and component private key S_2.
  - Return the hybrid public key $P = P_1 \,||\, P_2$ and the hybrid private key $S = S_1 \,||\, S_2$.
- **Signature generation:** Given the hybrid private key S and message M:
  - Call $\mathrm{Sig}_1.\,\mathrm{Sign}(S_1, M)$ to obtain the component signature value σ_1.
  - Call $\mathrm{Sig}_2.\,\mathrm{Sign}(S_2, M)$ to obtain the component signature value σ_2.
  - Return the hybrid signature value $\sigma = \sigma_1 \,||\, \sigma_2$.
- **Verification:** Given the hybrid public key P, message M and hybrid signature value σ:
  - Call $\mathrm{Sig}_1.\,\mathrm{Verify}(P_1, M, \sigma_1)$.
  - Call $\mathrm{Sig}_2.\,\mathrm{Verify}(P_2, M, \sigma_2)$.
  - Accept the hybrid signature value if both component signature values are valid, otherwise reject.

NOTE 1: The hybrid signature scheme will be EUF-CMA secure provided that at least one component digital signature algorithm is EUF-CMA secure [i.31].

NOTE 2: The hybrid signature scheme is trivially separable since each component signature value σ_i will be accepted by the component digital signature algorithm Sig_i as valid for the original message M.

EXAMPLE: The "concatenation only" hybrid signature scheme was proposed for general use in [i.35] and in the context of public-key infrastructures in [i.31].

B.3.2 Concatenation with identifiers

Weak non-separability can be achieved by including algorithm identifiers for the component digital signature algorithms in the message that is to be signed:

- **Key generation:**
  - Call $\mathrm{Sig}_1.\,\mathrm{KeyGen}()$ to obtain the component public key P_1 and component private key S_1.
  - Call $\mathrm{Sig}_2.\,\mathrm{KeyGen}()$ to obtain the component public key P_2 and component private key S_2.
  - Return the hybrid public key $P = P_1 \,||\, P_2$ and the hybrid private key $S = S_1 \,||\, S_2$.
- **Signature generation:** Given the hybrid private key S and a message M:
  - Call $\mathrm{Sig}_1.\,\mathrm{Sign}(S_1, \mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, M)$ to obtain the component signature value σ_1.

- Call $\mathrm{Sig}_2.\mathrm{Sign}(S_2, \mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, M)$ to obtain the component signature value σ_2.
- Return the hybrid signature value $\sigma = \sigma_1 \,||\, \sigma_2$.

- **Verification:** Given the hybrid public key P, message M and hybrid signature value σ
  - Call $\mathrm{Sig}_1.\mathrm{Verify}(P_1, \mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, M, \sigma_1)$.
  - Call $\mathrm{Sig}_2.\mathrm{Verify}(P_2, \mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, M, \sigma_2)$.
  - Accept the hybrid signature value if both component signature values are valid, otherwise reject.

NOTE 1: The hybrid signature scheme will be EUF-CMA secure provided that at least one component digital signature algorithm is EUF-CMA secure [i.31].

NOTE 2: The hybrid signature scheme is weakly non-separable since each composite signature value σ_i will be accepted by the component digital signature algorithm Sig_i as valid, but only for a message $M' = \mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, M$ that contains information about both component algorithms.

EXAMPLE: The "concatenation with identifiers" hybrid signature scheme is proposed for use in certificates and CMS in [i.54]. This uses a single identifier for a hybrid algorithm combination rather than two identifiers for individual component algorithms. Earlier versions of [i.54] specified the "concatenation only" hybrid signature scheme, but this was changed to provide non-separability.

B.3.3 Strong non-separability

Strong non-separability cannot be achieved by hybrid constructions that can only make black-box calls to the key generation, signature generation and verification functions of the underlying component algorithms: if the verification process for the hybrid digital signature scheme makes a black-box verification call to one of the component digital signature algorithms, then the inputs to this call necessarily give a message and signature value that will be valid for that component. Instead, strongly non-separable hybrid signature schemes need to modify the component signing and verification algorithms. This means that the hybrid construction will depend on the details of the component algorithms.

As an example, consider the following general framework for a Fiat-Shamir digital signature algorithm constructed from a 3-pass identification scheme with commitment function Comm, challenge function H, response function Resp, and verification function Ver:

- **Key generation:**
  - Return a public key P and private key S.
- **Signature generation:** Given the private key S and message M:
  - Choose a random seed r.
  - Compute the commitment $w = \mathrm{Comm}(S, r)$.
  - Compute the challenge $c = H(w, M)$.
  - Compute the response $d = \mathrm{Resp}(S, w, c, r)$.
  - Return the signature value $\sigma = w \,||\, d$.
- **Verification:** Given the public key P, message M and signature value σ:
  - Recover the challenge $c = H(w, M)$.
  - Accept if the verification $\mathrm{Ver}(P, w, c, d)$ passes, otherwise reject.

A separable "concatenation only" hybrid signature scheme that combines two Fiat-Shamir component digital signature algorithms, Sig_1 and Sig_2, can be modified to give a strongly non-separable hybrid signature scheme by changing the way that the challenges are generated:

- **Key generation:**
  - Call $\mathrm{Sig}_1.\,\mathrm{KeyGen}()$ to obtain the component public key P_1 and component private key S_1.
  - Call $\mathrm{Sig}_2.\,\mathrm{KeyGen}()$ to obtain the component public key P_2 and component private key S_2.
  - Return the hybrid public key $P = P_1 \,||\, P_2$ and the hybrid private key $S = S_1 \,||\, S_2$.
- **Signature generation:** Given the private key S and message M:
  - Choose random seeds r_1 and r_2.
  - Compute the commitments $w_1 = \mathrm{Sig}_1.\,\mathrm{Comm}(S_1, r_1)$ and $w_2 = \mathrm{Sig}_2.\,\mathrm{Comm}(S_2, r_2)$.
  - Compute the challenges $c_1 = \mathrm{Sig}_1.\,H(w_1 \,||\, w_2, M)$ and $c_2 = \mathrm{Sig}_2.\,H(w_1 \,||\, w_2, M)$.
  - Compute the responses $d_1 = \mathrm{Sig}_1.\,\mathrm{Resp}(S_1, w_1, c_1, r_1)$ and $d_2 = \mathrm{Sig}_2.\,\mathrm{Resp}(S_2, w_2, c_2, r_2)$.
  - Return the signature value $\sigma = w_1 \,||\, w_2 \,||\, d_1 \,||\, d_2$.
- **Verification:** Given the public key P, message M and signature value σ:
  - Recover the challenges $c_1 = \mathrm{Sig}_1.\,H(w_1 \,||\, w_2, M)$ and $c_2 = \mathrm{Sig}_2.\,H(w_1 \,||\, w_2, M)$.
  - Accept if both verifications $\mathrm{Sig}_1.\,\mathrm{Ver}(P_1, w_1, c_1, d_1)$ and $\mathrm{Sig}_2.\,\mathrm{Ver}(P_2, w_2, c_2, d_2)$ pass, otherwise reject.

NOTE: Given a hybrid signature value, it is not possible to extract a signature value that would be valid for one of the Fiat-Shamir component digital signature algorithms because the recovered challenge values will be different for the unmodified component algorithm.

EXAMPLE: Strongly non-separable hybrid signatures have been proposed in [i.51]. However, the construction in [i.51] that combines the post-quantum digital signature algorithm Dilithium and the traditional digital signature algorithm ECDSA does not include the rejection sampling for the Dilithium component algorithm. This means that the hybrid signature values will leak information about the Dilithium component private key, allowing an adversary to forge a Dilithium signature value and potentially breaking strong non-separability.

The need for strong non-separability will depend on the specific use case. Weak non-separability could be sufficient if the evidence of the original hybrid signature value can be detected by the protocol or causes the protocol to fail. Careful key management can also mitigate the risk of separable hybrid signature schemes by preventing any component of a hybrid key from being used by the component algorithm in a non-hybrid mode. For further discussion and examples, see [i.29].

Annex C:
Change History

Date	Version	Information about changes
April 2022	0.0.1	Initial skeleton document with scope and clause headings.
September 2022	0.0.2	Scope updated. Content added to clauses 4 (introduction), 5 (motivations) and 6 (design considerations).
December 2022	0.0.3	Revisions to clause 5 (motivations). Clause 6 (design considerations) completed.
February 2022	0.0.4	Minor editorial changes to clause 6 (design considerations). Content added to clause 7 (deployment considerations).

History

Document history		
V1.1.1	October 2024	Publication