ipsecme
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

T. Reddy
Nokia
V. Smyslov
ELVIS-PLUS
S. Fluhrer
Cisco Systems

20 October 2025

Signature Authentication in the Internet Key Exchange Version 2 (IKEv2) using PQC
draft-ietf-ipsecme-ikev2-pqc-auth-06

Abstract

Signature-based authentication methods are utilized in IKEv2 [RFC7296]. The current version of the Internet Key Exchange Version 2 (IKEv2) protocol supports traditional digital signatures.

This document specifies a generic mechanism for integrating post-quantum cryptographic (PQC) digital signature algorithms into the IKEv2 protocol. The approach allows for seamless inclusion of any PQC signature scheme within the existing authentication framework of IKEv2. Additionally, it outlines how Module-Lattice-Based Digital Signatures (ML-DSA) and Stateless Hash-Based Digital Signatures (SLH-DSA), can be employed as authentication methods within the IKEv2 protocol, as they have been standardized by NIST.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ipsecme-ikev2-pqc/.

Discussion of this document takes place on the ipsecme Working Group mailing list (mailto:ipsecme@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ipsec/. Subscribe at https://www.ietf.org/mailman/listinfo/ipsecme/.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 23 April 2026.

Copyright Notice

Table of Contents

1. Introduction

The Internet Key Exchange, or IKEv2 [RFC7296], is a key agreement and security negotiation protocol; it is used for key establishment in IPsec. In the IKE_AUTH exchange, the initiator and responder independently select and use their preferred authentication method, which may differ between peers. The most common authentication method is digital signatures using asymmetric cryptography. Currently, traditional digital signatures are defined for use within IKE_AUTH: RSA signatures, Elliptic Curve Digital Signature Algorithm (ECDSA) [RFC4754], and Edwards-curve Digital Signature Algorithm (EdDSA) [RFC8420].

The existence of a Cryptographically Relevant Quantum Computer (CRQC) would render traditional asymmetric algorithms obsolete and insecure. This is because the assumptions about the intractability of the mathematical problems these algorithms rely on, which offer confident levels of security today, no longer apply in the existence of a CRQC. Consequently, there is a requirement to update protocols and infrastructure to use post-quantum algorithms. Post-quantum algorithms are asymmetric algorithms designed to be secure against CRQCs as well as classical computers. The traditional cryptographic primitives that need to be replaced by PQC algorithms are discussed in [I-D.ietf-pquip-pqc-engineers].

This document defines a general approach to incorporating PQC digital signature algorithms into IKEv2 while maintaining interoperability and backward compatibility, as it does not change the IKEv2 protocol but adds negotiable PQC signature algorithms. Additionally, it outlines how Module-Lattice-Based Digital Signatures (ML-DSA) [FIPS204] and Stateless Hash-Based Digital Signatures (SLH-DSA) [FIPS205] can be employed as authentication methods within IKEv2, as they have been standardized the US National Institute of Standards and Technology (NIST) PQC project.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terms defined in [I-D.ietf-pquip-pqt-hybrid-terminology]. For the purposes of this document, it is helpful to be able to divide cryptographic algorithms into two classes:

"Asymmetric Traditional Cryptographic Algorithm": An asymmetric cryptographic algorithm based on integer factorisation, finite field discrete logarithms, elliptic curve discrete logarithms, or related mathematical problems.

"Post-Quantum Algorithm": An asymmetric cryptographic algorithm that is believed to be secure against attacks using quantum computers as well as classical computers. Post-quantum algorithms can also be called quantum-resistant or quantum-safe algorithms. Examples of quantum-resistant digital signature schemes include ML-DSA and SLH-DSA.

3. General Framework for PQC Authentication in IKEv2

IKEv2 authentication commonly relies on digital signatures to verify the identity of communicating peers. The mechanism described in this document enables the use of any PQC digital signature algorithm without modifying core IKEv2 operations.

3.1. Specifying PQC Signature Algorithms

* IKEv2 can use arbitrary signature algorithms as described in [RFC7427], where the "Digital Signature" authentication method supersedes previously defined signature authentication methods. Any PQC digital signature algorithm can be incorporated using the "Signature Algorithm" field in authentication payloads, as defined in [RFC7427].

* DER encoded AlgorithmIdentifier ASN.1 objects will be used to uniquely identify PQC signature algorithm scheme and the parameter set associated with it.

3.2. Signature Generation and Verification

PQC signatures may be generated in either deterministic or hedged modes. The terms deterministic and hedged used in this document are in accordance with [FIPS204] and [FIPS205], which define the ML-DSA and SLH-DSA algorithms. Future PQC signature algorithms may adopt different nomenclature, but will be expected to follow the same principles.

In the deterministic mode, the signature is derived entirely from the message and the signer’ s private key, without introducing fresh randomness at signing time. While this eliminates reliance on an external random number generator (RNG), it increases susceptibility to side-channel attacks, particularly fault injection attacks.

The hedged mode provides some resistance against this risk by including precomputed randomness in the signer’s private key and incorporating fresh randomness generated at signing time. This foils some side channel attack approaches, while adding no additional strength against others. If protection against side-channel attacks are required, an ML-DSA implementation that implements side-channel resistance should be used.

In the context of signature-based authentication in IKEv2, the data used for generating a digital signature is unique for each session, as it includes session-specific information such as nonces. PQC signature algorithms can leverage the hedged variant within IKEv2 to enhance security against side-channel attacks. The choice between deterministic and hedged signing modes does not impact interoperability because the verification process remains the same for both variants.

If the PQC signature algorithm uses a ’context’ input parameter, it MUST be set to an empty string.

Certain digital signature algorithms support two modes: "pure" mode and "pre-hash" mode. For example, ML-DSA and SLH-DSA support both modes. In pure mode, the content is signed directly along with some domain separation information. In contrast, pre-hash mode involves signing a digest of the message. This document specifies the use of pure mode for signature-based authentication in IKEv2, where the message is signed directly along with domain separation information. The data used for authentication in IKEv2, as described in Section 2.15 of [RFC7296], consists of elements such as nonces, SPIs, and initial exchange messages (messages preceding IKE_AUTH), which are typically within device memory constraints. While pre-hash mode can help in scenarios with memory constraints, the IKEv2 authentication data is generally small, and combined with other practical challenges (discussed in Appendix A), this document only specifies pure mode.

3.2.1. Handling PQC Signatures in IKEv2

As specified in [RFC7427], both the initiator and responder MUST send the SIGNATURE_HASH_ALGORITHMS notify payload in the IKE_SA_INIT exchange to indicate the set of hash algorithms they support for signature generation and verification. The SIGNATURE_HASH_ALGORITHMS notify payload contains a list of 2-octet hash algorithm identifiers, defined in the IANA "IKEv2 Hash Algorithms" registry.

For PQC signature algorithms that inherently operate directly on the raw message without hashing, such as ML-DSA and SLH-DSA, only the 'Identity' hash function is applicable. The 'Identity' hash function (value 5) is defined in Section 2 of [RFC8420] and indicates that the input message is used as-is, without any hash function applied. Therefore, implementations supporting such PQC signature algorithms MUST include the 'Identity' hash (5) in the SIGNATURE_HASH_ALGORITHMS notify. Furthermore, PQC signature algorithms requiring the 'Identity' hash MUST NOT be used with a peer that has not indicated support for the Identity hash in its notify payload.

For additional background on design alternatives that were considered and the rationale for adopting the [RFC8420] approach as the cleanest and most secure method, see Appendix A.

When generating a signature with a PQC signature algorithm, the IKEv2 implementation takes the InitiatorSignedOctets string or the ResponderSignedOctets string (as appropriate), logically sends it to the identity hash (which leaves it unchanged), and then passes it into the PQC signer as the message to be signed (with empty context string, if applicable). The resulting signature is placed into the Signature Value field of the Authentication Payload.

When verifying a signature with a PQC signature algorithm, the IKEv2 implementation takes the InitiatorSignedOctets string or the ResponderSignedOctets string (as appropriate), logically sends it to the identity hash (which leaves it unchanged), and then passes it into the PQC signature verifier as the message to be verified (with empty context string, if applicable).

IKEv2 peers supporting the PQC authentication mechanism defined in this specification SHOULD implement IKEv2 message fragmentation as specified in [RFC7383], unless IKEv2 runs over a reliable transport (e.g., [RFC9329]) or the underlying network is known to support sufficiently large MTUs without fragmentation issues, since PQC public keys and signatures can be significantly larger than those used in traditional algorithms. For example, ML-DSA-44 requires a public key of 1,312 bytes and a signature of 2,420 bytes, while even the smallest SLH-DSA signature is around 7,856 bytes. As guidance, IKEv2 peers should assume a minimum PMTU of 1280 bytes for IPv6 (per [RFC8200]) and, where legacy IPv4 networks are a consideration, an effective MTU of 576 bytes for IPv4 (per [RFC1122]).

3.3. Mechanisms for Signaling Supported Key Pair Types

The following mechanisms can be used by peers to signal the types of digital signature algorithms and parameters they support:

* Certificate Request Payload: One method to ascertain that the key pair type the initiator wants the responder to use is through a Certificate Request payload (defined in Section 3.7 of [RFC7296]) sent by the initiator. For example, the initiator can specify that it trusts certificates issued by a certificate authority (CA) that signs with a particular post-quantum cryptographic (PQC) signature algorithm. This implies that the initiator can process signatures generated using that algorithm, thereby allowing the responder to authenticate itself using a key pair associated with the specified PQC signature scheme.

* Authentication Method Announcement: Another method is to utilize [RFC9593],
which enables peers to declare their supported authentication methods. This improves interoperability when IKEv2 peers are configured with multiple credential types of different type to authenticate each other. The responder includes a SUPPORTED_AUTH_METHODS notification in the IKE_SA_INIT response message, listing the signature scheme(s) it supports under the Digital Signature authentication method. The initiator includes the SUPPORTED_AUTH_METHODS notification in either the IKE_AUTH request message or in the IKE_INTERMEDIATE request. This notification lists the digital signature scheme(s) supported by the initiator, ordered by preference.

In traditional IKEv2 deployments, peers often implicitly know the signature algorithms in use based on pre-configured certificates, trusted CAs, and IKEv2 policies. However, cryptographic agility, the ability to negotiate and use different cryptographic algorithms is gaining increased attention for ensuring long-term security and interoperability. This requirement becomes even more relevant with the introduction of PQC algorithms, where multiple signature algorithms with varying security levels and performance characteristics may need to be supported over time.

4. Specifying ML-DSA within IKEv2

ML-DSA [FIPS204] is a digital signature algorithm based on the hardness lattice problems over module lattices (i.e., the Module Learning with Errors problem (MLWE)). The design of the algorithm is based on the "Fiat-Shamir with Aborts" [Lyu09] framework introduced by Lyubashevsky that leverages rejection sampling to render lattice-based FS schemes compact and secure. ML-DSA uses a uniform distribution over small integers for computing coefficients in error vectors, which simplifies implementation compared to schemes requiring discrete Gaussian sampling.

ML-DSA is instantiated with three parameter sets for the security categories 2, 3, and 5 (see Table 2 in Section 10 of [I-D.ietf-pquip-pqc-engineers]). Security properties of ML-DSA are discussed in Section 9 of [I-D.ietf-lamps-dilithium-certificates]. This document specifies the use of the ML-DSA algorithm in IKEv2 at three security levels: ML-DSA-44, ML-DSA-65, and ML-DSA-87. The DER encodings of the AlgorithmIdentifier objects for ML-DSA-44, ML-DSA-65, and ML-DSA-87 are listed in Appendix B.

5. Specifying SLH-DSA within IKEv2

SLH-DSA [FIPS205] utilizes the concept of stateless hash-based signatures. In contrast to stateful signature algorithms such as XMSS [RFC8391] or HSS/LMS [RFC8554], SLH-DSA eliminates the need for maintaining state information during the signing process. SLH-DSA is designed to sign up to 2^64 messages and it offers three security levels. The parameters for security levels 1, 3, and 5 were chosen to provide AES-128, AES-192, and AES-256 bits of security respectively (see Table 2 in Section 10 of [I-D.ietf-pquip-pqc-engineers]). This document specifies the use of the SLH-DSA algorithm in IKEv2 at each level.

Each security level (1, 3, and 5) defines two variants of the algorithm: a small (S) version and a fast (F) version. The small version prioritizes smaller signature sizes, making them suitable for resource-constrained IoT devices. Conversely, the fast version prioritizes speed over signature size, minimizing the time required to generate signatures. However, signature verification with the small version is faster than with the fast version. For hash function selection, the algorithm uses SHA-256 ([FIPS180]) for security level 1 and both SHA-256 and SHA-512 ([FIPS180]) for security levels 3 and 5. Alternatively, SHAKE256 ([FIPS202]) can be used across all security levels. Those hash function selections are internal to SLH-DSA implementations, and are not to be confused with those in the SIGNATURE_HASH_ALGORITHMS notification payload.

ML-DSA outperforms SLH-DSA in both signature generation and validation time, as well as signature size. SLH-DSA, in contrast, offers smaller key sizes but larger signature sizes.

The following combinations are defined in SLH-DSA [FIPS205]:

* SLH-DSA-128S-SHA2
* SLH-DSA-128F-SHA2
* SLH-DSA-192S-SHA2
* SLH-DSA-192F-SHA2
* SLH-DSA-256S-SHA2
* SLH-DSA-256F-SHA2
* SLH-DSA-128S-SHAKE
* SLH-DSA-128F-SHAKE

* SLH-DSA-192S-SHAKE

* SLH-DSA-192F-SHAKE

* SLH-DSA-256S-SHAKE

* SLH-DSA-256F-SHAKE

SLH-DSA does not introduce a new hardness assumption beyond those inherent to the underlying hash functions. It builds upon established foundations in cryptography, making it a reliable and robust digital signature scheme in the face of a CRQC. While attacks on lattice-based schemes like ML-DSA are currently hypothetical at the time of writing this document, such attacks, if realized, could compromise their security. SLH-DSA would remain unaffected by these attacks due to its distinct mathematical foundations. This ensures the continued security of systems and protocols that utilize SLH-DSA for digital signatures.

The DER encodings of the AlgorithmIdentifier objects for each SLH-DSA variant are listed in Appendix B.

6. Implementation Alternatives for ML-DSA

With ML-DSA, there are two different approaches to implementing the signature process. The first one is to simply hand the SignedOctets string to the cryptographic library to generate the full signature; this works for SLH-DSA as well.

The second approach involves using the Externalμ-ML-DSA API defined in [FIPS204]. In this method, the implementation calls the External μ pre-hashing mode with the SignedOctets string and the ML-DSA public key, which externalizes the message pre-hashing originally performed inside the signing operation (see Appendix D of [I-D.ietf-lamps-dilithium-certificates]). This hash is then passed to the cryptographic library to execute the Externalμ-ML-DSA.Sign API, which uses the hash and the ML-DSA private key to produce the signature. This document specifies only the use of ML-DSA's External μ mode, and not HashML-DSA.

Both approaches are considered "pure" mode and produce the same ML-DSA signature and are fully interoperable. The choice between them depends on implementation preferences, such as whether the External μ pre-hashing step is handled internally by the cryptographic module or performed explicitly by the IKEv2 implementation.

7. Use of ML-DSA and SLH-DSA

Both ML-DSA and SLH-DSA offer deterministic and hedged signing modes. By default, ML-DSA uses a hedged approach, where the random value rnd is a 256-bit string generated by an Random Bit Generator (RBG). The signature generation function utilizes this randomness along with the private key and the preprocessed message. In the deterministic variant, rnd is instead set to a constant 256-bit zero string. Similarly, SLH-DSA can operate in either deterministic or hedged mode. The mode is determined by the value of opt_rand, when opt_rand is set to a fixed value (e.g., the public seed from the public key), SLH-DSA generates deterministic signatures, ensuring that signing the same message twice produces the same signature. In hedged mode, opt_rand is a fresh random value, introducing additional entropy to enhance security and mitigate potential side-channel risks.

IKEv2 peers can use either the hedged or deterministic variants of ML-DSA and SLH-DSA for authentication in IKEv2, with a preference for using the hedged mode (Section 3.2).

The three security levels of ML-DSA are identified via AlgorithmIdentifier ASN.1 objects, as specified in NIST [CSOR] and referenced in [I-D.ietf-lamps-dilithium-certificates]. [FIPS204] defines both a pure and a pre-hash variant of ML-DSA, but [I-D.ietf-lamps-dilithium-certificates] specifies only the pure variant.

The different combinations of SLH-DSA are identified via AlgorithmIdentifier ASN.1 objects, as specified in NIST [CSOR] and referenced in [I-D.ietf-lamps-x509-slhdsa]. [FIPS205] defines two signature modes: pure mode and pre-hash mode. [I-D.ietf-lamps-x509-slhdsa] specifies the use of both Pure SLH-DSA and HashSLH-DSA in Public Key Infrastructure X.509 (PKIX) certificates and Certificate Revocation Lists (CRLs).

8. Security Considerations

PQC signature algorithms are generally modeled to achieve strong unforgeability under adaptive chosen-message attacks (SUF-CMA; see Section 10.1.1 of [I-D.ietf-pquip-pqc-engineers]). For example, ML-DSA provides SUF-CMA security. However, some algorithms, such as SLH-DSA, achieve existential unforgeability under chosen-message attacks (EUF-CMA; see Section 10.1.1 of [I-D.ietf-pquip-pqc-engineers]). This distinction does not impact IKEv2, as the signed data in each session is unique due to the inclusion of nonces. Consequently, the oracle-based forgery attack scenarios in the EUF-CMA model do not arise in IKEv2.

Different PQC signature schemes are designed to provide security levels comparable to well-established cryptographic primitives. For example, some schemes align with the NIST post-quantum security categories (Categories 1 through 5) as discussed in [FIPS204] and [FIPS205]. These categories specify target security strengths that correspond approximately to exhaustive key-search resistance for AES-128, AES-192, and AES-256, and collision-search resistance for SHA-256, SHA-384, and SHA-512. The choice of a PQC signature algorithm should be guided by the desired security level and performance requirements.

ML-DSA-44, ML-DSA-65, and ML-DSA-87 are designed to offer security comparable with the SHA-256/SHA3-256, AES-192, and AES-256 respectively. Similarly, SLH-DSA-128{S,F}-{SHA2,SHAKE}, SLH-DSA-192{S,F}-{SHA2,SHAKE}, and SLH-DSA-256{S,F}-{SHA2,SHAKE} are designed to offer security comparable with the AES-128, AES-192, and AES-256 respectively.

The Security Considerations section of [I-D.ietf-lamps-dilithium-certificates] and [I-D.ietf-lamps-x509-slhdsa] apply to this specification as well.

SLH-DSA keys are limited to 2^64 signatures. This upper bound is so large that even a IKEv2 server establishing IKEv2 sessions at an extremely high rate could not realistically reach it (at 10 billion signatures per second, it would still take over 58 years). The limit is therefore of theoretical interest only, but implementations may still track signature usage as a precautionary security measure. ML-DSA does not have a built-in signature limit, allowing for an arbitrary number of signatures to be made with the same key.

Acknowledgements

Thanks to Stefaan De Cnodder, Loganaden Velvindron, Paul Wouters, Andreas Steffen, Dan Wing, Wang Guilin, Rebecca Guthrie, Jonathan Hammell, John Mattsson and Daniel Van Geest for the discussion and comments.

References

Normative References

[CSOR] NIST, "Computer Security Objects Register", 20 August 2024, <https://csrc.nist.gov/projects/computer-security-objects-register/algorithm-registration>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, DOI 10.17487/RFC4754, January 2007, <https://www.rfc-editor.org/rfc/rfc4754>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/rfc/rfc7296>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <https://www.rfc-editor.org/rfc/rfc7383>.

[RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <https://www.rfc-editor.org/rfc/rfc7427>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8420] Nir, Y., "Using the Edwards-Curve Digital Signature Algorithm (EdDSA) in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8420, DOI 10.17487/RFC8420, August 2018, <https://www.rfc-editor.org/rfc/rfc8420>.

[RFC9593] Smyslov, V., "Announcing Supported Authentication Methods in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 9593, DOI 10.17487/RFC9593, July 2024, <https://www.rfc-editor.org/rfc/rfc9593>.

Informative References

[FIPS180] "NIST, Secure Hash Standard (SHS), FIPS PUB 180-4, August 2015", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[FIPS202] "NIST, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, August 2015.", <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf>.

[FIPS204] "FIPS 204: Module-Lattice-Based Digital Signature Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf>.

[FIPS205] "FIPS 205: Stateless Hash-Based Digital Signature Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf>.

[I-D.ietf-lamps-dilithium-certificates] Massimo, J., Kampanakis, P., Turner, S., and B. Westerbaan, "Internet X.509 Public Key Infrastructure - Algorithm Identifiers for the Module-Lattice-Based Digital Signature Algorithm (ML-DSA)", Work in Progress, Internet-Draft, draft-ietf-lamps-dilithium-certificates-13, 30 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-dilithium-certificates-13>.

[I-D.ietf-lamps-x509-slhdsa] Bashiri, K., Fluhrer, S., Gazdag, S., Van Geest, D., and S. Kousidis, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for SLH-DSA", Work in Progress, Internet-Draft, draft-ietf-lamps-x509-slhdsa-09, 30 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-x509-slhdsa-09>.

[I-D.ietf-pquip-pqc-engineers] Banerjee, A., Reddy.K, T., Schoinianakis, D., Hollebeek, T., and M. Ounsworth, "Post-Quantum Cryptography for Engineers", Work in Progress, Internet-Draft, draft-ietf-pquip-pqc-engineers-14, 25 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqc-engineers-14>.

[I-D.ietf-pquip-pqt-hybrid-terminology] D, F., P, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-ietf-pquip-pqt-hybrid-terminology-06, 10 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqt-hybrid-terminology-06>.

[Lyu09] "V. Lyubashevsky, “Fiat-Shamir With Aborts: Applications to Lattice and Factoring-Based Signatures “, ASIACRYPT 2009", <https://www.iacr.org/archive/asiacrypt2009/59120596/59120596.pdf>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/rfc/rfc1122>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/rfc/rfc8200>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/rfc/rfc8391>.

[RFC8554] McGrew, D., Curcio, M., and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC 8554, DOI 10.17487/RFC8554, April 2019, <https://www.rfc-editor.org/rfc/rfc8554>.

[RFC9329] Pauly, T. and V. Smyslov, "TCP Encapsulation of Internet Key Exchange Protocol (IKE) and IPsec Packets", RFC 9329, DOI 10.17487/RFC9329, November 2022, <https://www.rfc-editor.org/rfc/rfc9329>.

Appendix A. Discussion of ML-DSA and SLH-DSA and Prehashing

This section discusses various approaches for integrating ML-DSA and SLH-DSA into IKEv2 other than those proposed above.

The signature architecture within IKE was designed around RSA (and later extended to ECDSA). In this architecture, the actual message (the SignedOctets) are first hashed (using a hash that the verifier has indicated support for), and then passed for the remaining part of the signature generation processing. That is, it is designed for signature algorithms that first apply a hash function to the message and then perform processing on that hash. Neither ML-DSA nor SLH-DSA fits cleanly into this architecture.

We see three ways to address this mismatch.

The first consideration is that both ML-DSA and SLH-DSA provide prehashed parameter sets, which are designed to sign messages that have already been hashed by an external source. At first glance, this might seem like an ideal solution. However, several practical challenges arise:

1. The prehashed versions of ML-DSA and SLH-DSA appear to be rarely used, making it likely that support for them in cryptographic libraries is limited or unavailable.

2. The public keys for the prehashed variants use different OIDs, which means that certificates for IKEv2 would differ from those used in other protocols. This not only complicates certificate management but also adds protocol complexity if a peer needs to support both pure and prehashed variants. Additionally, some certificate authorities (CAs) may not support issuing certificates for prehashed ML-DSA or SLH-DSA due to their limited adoption.

3. Some users have explicitly indicated a preference not to use the prehashed parameter sets.

The second is to note that, while IKEv2 normally follows the ’hash and then sign’ paradigm, it doesn’t always. EdDSA has a similar constraint on not working cleanly with the standard ’hash and then sign’ paradigm, and so the existing [RFC8420] provides an alternative method, which ML-DSA would cleanly fit into. We could certainly adopt this same strategy; our concern would be that it might be more difficult for IKEv2 implementors which do not already have support for EdDSA.

The third way is what we can refer to as ’fake prehashing’; IKEv2 would generate the hash as specified by the pre-hash modes in [FIPS204] and [FIPS205], but instead of running ML-DSA or SLH-DSA in prehash mode, the hash is signed as if it were the unhashed message, as is done in pure mode. This is a violation of the spirit, if not the letter of FIPS 204, 205. However, it is secure (assuming the hash function is strong), and fits in cleanly with both the existing IKEv2 architecture, and what crypto libraries provide. Additionally, for SLH-DSA, this means that we’re now dependent on collision resistance (while the rest of the SLH-DSA architecture was carefully designed not to be).

After analysis, the IPsecME working group selected the approach defined in [RFC8420], the second method discussed above as the most cryptographically secure way to integrate PQC algorithms into IKEv2. The details are specified in Section 3.2.1.

Appendix B. ASN.1 Objects

This section lists ASN.1 objects for algorithms mentioned in the document in binary form.
This section is not normative, and these values should only be used as examples. If the ASN.1 object listed in this section and the ASN.1 object specified by the algorithm differ, then the algorithm specification must be used.

B.1. ML-DSA-44

```
id-ml-dsa-44 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) sigAlgs(3)
id-ml-dsa-44(17) }
```

Parameters are absent.

```
Name = id-ml-dsa-44, oid = 2.16.840.1.101.3.4.3.17
Length = 13
0000: 300b 0609 6086 4801 6503 0403 11
```

B.2. ML-DSA-65

```
id-ml-dsa-65 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) sigAlgs(3)
id-ml-dsa-65(18) }
```

Parameters are absent.

```
Name = id-ml-dsa-65, oid = 2.16.840.1.101.3.4.3.18
Length = 13
0000: 300b 0609 6086 4801 6503 0403 12
```

B.3. ML-DSA-87

```
id-ml-dsa-87 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) sigAlgs(3)
id-ml-dsa-87(19) }
```

Parameters are absent.

```
Name = id-ml-dsa-87, oid = 2.16.840.1.101.3.4.3.19
Length = 13
0000: 300b 0609 6086 4801 6503 0403 13
```

B.4. SLH-DSA-128S-SHA2

```
id-slh-dsa-sha2-128s OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-sha2-128s(20) }

Parameters are absent.

Name = id-slh-dsa-sha2-128s, oid = 2.16.840.1.101.3.4.3.20
Length = 13
0000: 300b 0609 6086 4801 6503 0403 14
```

B.5. SLH-DSA-128F-SHA2

```
id-slh-dsa-sha2-128f OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-sha2-128f(21) }

Parameters are absent.

Name = id-slh-dsa-sha2-128f, oid = 2.16.840.1.101.3.4.3.21
Length = 13
0000: 300b 0609 6086 4801 6503 0403 15
```

B.6. SLH-DSA-192S-SHA2

```
id-slh-dsa-sha2-192s OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-sha2-192s(22) }

Parameters are absent.

Name = id-slh-dsa-sha2-192s, oid = 2.16.840.1.101.3.4.3.22
Length = 13
0000: 300b 0609 6086 4801 6503 0403 16
```

B.7. SLH-DSA-192F-SHA2

```
id-slh-dsa-sha2-192f OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-sha2-192f(23) }

Parameters are absent.

Name = id-slh-dsa-sha2-192f, oid = 2.16.840.1.101.3.4.3.23
Length = 13
0000: 300b 0609 6086 4801 6503 0403 17
```

B.8. SLH-DSA-256S-SHA2

```
id-slh-dsa-sha2-256s OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-sha2-256s(24) }

Parameters are absent.

Name = id-slh-dsa-sha2-256s, oid = 2.16.840.1.101.3.4.3.24
Length = 13
0000: 300b 0609 6086 4801 6503 0403 18
```

B.9. SLH-DSA-256F-SHA2

```
id-slh-dsa-sha2-256f OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-sha2-256f(25) }

Parameters are absent.

Name = id-slh-dsa-sha2-256f, oid = 2.16.840.1.101.3.4.3.25
Length = 13
0000: 300b 0609 6086 4801 6503 0403 19
```

B.10. SLH-DSA-128S-SHAKE

```
id-slh-dsa-shake-128s OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-shake-128s(26) }

Parameters are absent.

Name = id-slh-dsa-shake-128s, oid = 2.16.840.1.101.3.4.3.26
Length = 13
0000: 300b 0609 6086 4801 6503 0403 1a
```

B.11. SLH-DSA-128F-SHAKE

```
id-slh-dsa-shake-128f OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-shake-128f(27) }

Parameters are absent.

Name = id-slh-dsa-shake-128f, oid = 2.16.840.1.101.3.4.3.27
Length = 13
0000: 300b 0609 6086 4801 6503 0403 1b
```

B.12. SLH-DSA-192S-SHAKE

```
id-slh-dsa-shake-192s OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-shake-192s(28) }

Parameters are absent.

Name = id-slh-dsa-shake-192s, oid = 2.16.840.1.101.3.4.3.28
Length = 13
0000: 300b 0609 6086 4801 6503 0403 1c
```

B.13. SLH-DSA-192F-SHAKE

```
id-slh-dsa-shake-192f OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-shake-192f(29) }

Parameters are absent.

Name = id-slh-dsa-shake-192f, oid = 2.16.840.1.101.3.4.3.29
Length = 13
0000: 300b 0609 6086 4801 6503 0403 1d
```

B.14. SLH-DSA-256S-SHAKE

```
id-slh-dsa-shake-256s OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-shake-256s(30) }

Parameters are absent.

Name = id-slh-dsa-shake-256s, oid = 2.16.840.1.101.3.4.3.30
Length = 13
0000: 300b 0609 6086 4801 6503 0403 1e
```

B.15. SLH-DSA-256F-SHAKE

```
id-slh-dsa-shake-256f OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4)
sigAlgs(3) id-slh-dsa-shake-256f(31) }

Parameters are absent.

Name = id-slh-dsa-shake-256f, oid = 2.16.840.1.101.3.4.3.31
Length = 13
0000: 300b 0609 6086 4801 6503 0403 1f
```

Authors' Addresses

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: kondtir@gmail.com

Valery Smyslov
ELVIS-PLUS
Russian Federation
Email: svan@elvis.ru

Scott Fluhrer
Cisco Systems
Email: sfluhrer@cisco.com