COSE
Internet-Draft
Intended status: Standards Track
Expires: 31 October 2026

A. R. Mott
RustyKey

29 April 2026

CBOR Object Signing and Encryption (COSE) and JSON Object Signing and Encryption (JOSE) Registrations for SQIsign
draft-mott-cose-sqisign-03

Abstract

NOTE: This document describes a signature scheme based on an algorithm currently under evaluation in the NIST Post-Quantum Cryptography standardization process. Be aware that the underlying primitive may change as a result of that process.

This document specifies the algorithm encodings and representations for the SQIsign digital signature scheme within the CBOR Object Signing and Encryption (COSE) and JSON Object Signing and Encryption (JOSE) frameworks.

SQIsign is an isogeny-based post-quantum signature scheme that provides exceptionally compact signature and public key sizes compared to lattice-based alternatives currently under NIST evaluation.

The standardization of SQIsign will be helpful to address current infrastructure bottlenecks, specifically the FIDO2 CTAP2 specification used by billions of in-service devices and browser installations. Some deployments of CTAP2-based authenticators enforce limits near 1024 bytes for external key communication, depending on authenticator implementation, transport (USB/NFC/BLE) and message fragmentation support. Some standardized post-quantum signature schemes with larger signature sizes may exceed the message size limits of constrained authenticators, transports or produce surprising and unwanted results further along the authentication ceremony. SQIsign-L1, L2 and L5 signatures are small enough to enable delivery over constrained networks like 802.15.4.

This document clarifies that SQIsign does not expose the auxiliary torsion-point information exploited in the SIDH/SIKE attacks. Consequently, the specific attack techniques of Castryck宴泥ecru do not directly apply. However, the scheme remains subject to ongoing cryptanalysis of isogeny-based constructions. By establishing stable COSE and JOSE identifiers, this document ensures the interoperability required for the seamless integration of post-quantum security into high-density, bandwidth-constrained, and legacy-compatible hardware environments.

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-mott-cose-sqisign/.

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

Source for this draft and an issue tracker can be found at https://github.com/antonymott/quantum-resistant-rustykey.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document registers algorithm identifiers and key type parameters for SQIsign in COSE and JOSE.

1.1. Background and Motivation

Post-quantum cryptography readiness is critical for constrained devices. As of 2026, while FIDO2/WebAuthn supports various COSE algorithms, some hardware authenticators and platform authenticators (like TPMs) have strict memory/storage constraints, effectively limiting public keys to 1024 bytes or less, hindering the adoption of large-key post-quantum algorithms.

1.1.1. Pressing Need for Smaller PQC Signatures

FN-DSA (Falcon) and ML-DSA (Dilithium) have larger signatures that may not fit in constrained environments.

The fundamental differences between ML-DSA, FN-DSA, and SQIsign lie in their underlying hard mathematical problems, implementation complexity, and performance trade-offs.

FN-DSA (Falcon, NIST alternative) uses NTRU lattices to achieve very small signatures and fast verification, but requires complex floating-point math. Dilithium (NIST primary) is a balanced, high-efficiency lattice scheme using Module-LWE/SIS, easy to implement.

SQIsign [SQIsign-Spec] [SQIsign-Analysis] is a non-lattice, isogeny-based scheme that offers the smallest signature sizes but suffers from significantly slower signature generation where even vI may take seconds to minutes, or longer with WASM implementations for browsers of particular relevance to signatures required for WebAuthn PassKeys [WebAuthn-PQC-Signature-size-constraints]. SQIsign is an isogeny-based digital signature scheme participating in NIST's Round 2 Additional Digital Signature Schemes, not yet a NIST standard [NIST-Finalized-Standards].

Speed comparison (approximate, implementation-dependent):

* *Signing:* SQIsign is 100x-1000x slower than ML-DSA

* *Verification:* SQIsign is 2x-10x slower than ML-DSA

* *Key Generation:* SQIsign is 50x-500x slower than ML-DSA

Note: Performance varies significantly based on parameter set and optimization level. WASM implementations may be substantially slower.

Table 1 compares representative parameter sets; note that these schemes are at different stages of standardization and evaluation.

Algorithm	Public Key Size	Signature Size	PK + Sig < 1024?
ML-DSA-44	1,312 bytes	2,420 bytes	笑
ML-DSA-65	1,952 bytes	3,293 bytes	
ML-DSA-87	2,592 bytes	4,595 bytes	
FN-DSA-512	897 bytes	666 bytes	(1,563 total)
FN-DSA-1024	1,793 bytes	1,280 bytes	
SQIsign-L1	65 bytes	148 bytes	(213 total)
SQIsign-L3	97 bytes	224 bytes	(321 total)
SQIsign-L5	129 bytes	292 bytes	(421 total)

Table 1

1.1.2. Estimated Constrained Device Footprint

The total addressable market for SQIsign in constrained devices is estimated at more than 6 billion units.

Total Addressable Market: ~6.77 billion devices - Legacy Hardware Security Keys: 120-150 million - Constrained TPMs: ~1.1 billion - Browser Implementations: ~5 billion - Critical Infrastructure: ~300 million - Other IoT: ~250 million

1.1.2.1. Device Category Breakdown

1.1.2.1.1. Legacy Hardware Security Keys: ~120 - 150 million

* Security keys in Service: ~120 - 150 million legacy keys in active circulation (Series 5 and older). Some firmware introduced PQC readiness. Some older keys cannot be updated to increase buffer sizes.

1.1.2.1.2. Constrained TPMs and Platform Modules: ~1.1 billion

Trusted Platform Modules (TPMs) are integrated into PCs and servers, but their WebAuthn implementation often inherits protocol-level constraints. Estimated ~2.5 billion active chips worldwide. Constrained Subset: We estimate ~1.1 billion of these are in older Windows 10/11 or Linux machines where the OS "virtual authenticator" or TPM driver still enforces the 1024-byte message default to maintain backward compatibility with external CTAP1/2 tools.

1.1.2.1.3. Browser and Software Implementations: ~5 billion

This category refers to the "User-Agent" layer that mediates between the web and the hardware. Global Browser Agents: There are over 5 billion active browser instances across mobile and desktop (Chrome, Safari, Edge, Firefox). Legacy Protocols: Even on modern hardware, browsers often use the FIDO2 CTAP2 specification which, unless explicitly negotiated for larger messages, maintains a 1024-byte default for external key communication.

1.1.2.1.4. Critical Infrastructure: ~300 Million includes Energy (electric, nuclear, oil, gas), Water & Wastewater, Transportation Systems, Communications, Government, Emergency Services, Healthcare and Financial Services

Industrial/Government: Agencies like the U.S. Department of Defense rely on high-security FIPS-certified keys that are notoriously slow to upgrade. We estimate ~50 million "frozen" government keys. IoT Security: Of the ~21 billion connected IoT devices in 2026, only a fraction use WebAuthn. However, for those that do (smart locks, secure gateways), approximately 250 million are estimated to use older, non-upgradable secure elements limited to 1024-byte payloads. Recent government-level initiatives highlight the necessity to "...effectively deprecate the use of RSA, Diffie-Hellman (DH), and

elliptic curve cryptography (ECDH and ECDSA) when mandated." [CNSA-2], Page 4.

1.1.3. Pressing need: Limit or Stop 'Harvest now; decrypt later' Attacks

Adversaries are collecting encrypted data today to decrypt when quantum computers become available. The transition to post-quantum cryptography (PQC) is critical for ensuring long-term security of digital communications against adversaries equipped with large-scale quantum computers. The National Institute of Standards and Technology (NIST) has been leading standardization efforts, having selected initial PQC algorithms and continuing to evaluate additional candidates.

CBOR Object Signing and Encryption (COSE) [RFC9052] is specifically designed for constrained node networks and IoT environments where bandwidth, storage, and computational resources are limited. The compact nature of SQIsign makes it an ideal candidate for COSE deployments.

1.2. Scope and Status

This document is published on the *Standards* track rather than Informational Track for the following reasons:

1. *Algorithm Maturity*: SQIsign is currently undergoing evaluation in NIST's on-ramp process

2. *Continued Cryptanalysis*: The algorithm has active ongoing review by the cryptographic research community, including the IRTF CFRG

3. *High anticipated demand*: This specification enables experimentation and early deployment to gather implementation experience

This document does not represent Working Group consensus on algorithm innovation. The COSE and JOSE working groups focus on algorithm _integration_ and _encoding_, not cryptographic algorithm design. The cryptographic properties of SQIsign are being evaluated through NIST's process and academic peer review.

1.3. Relationship to Other Work

This document follows the precedent established by [I-D.ietf-cose-falcon] and [I-D.ietf-cose-dilithium] for integrating NIST PQC candidate algorithms into COSE and JOSE. The structure and approach are intentionally aligned to provide consistency across post-quantum signature scheme integrations.

1.4. Constrained Device Applicability

SQIsign is particularly attractive for:

* *IoT sensors* with limited flash memory
* *Firmware updates* over low-bandwidth networks (LoRaWAN, NB-IoT)
* *Embedded certificates* in constrained devices
* *Blockchain and DLT* where transaction size affects fees
* *Satellite communications* with bandwidth constraints

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 the following terms:

* *PQC*: Post-Quantum Cryptography
* *COSE*: CBOR Object Signing and Encryption
* *JOSE*: JSON Object Signing and Encryption
* *JWS*: JSON Web Signature
* *JWK*: JSON Web Key
* *CBOR*: Concise Binary Object Representation [RFC7049]
* *ECDH*: Elliptic Curve Diffie-Hellman
* *IANA*: Internet Assigned Numbers Authority

3. Resistance to "Torsion Point" attack

3.1. SIKE Vulnerability (The "Torsion Point" Attack) of 2022

SIKE (Supersingular Isogeny Key Encapsulation) was a key exchange, more specifically, a Key Encapsulation Mechanism (KEM). In the SIKE protocol, users had to share more than just the target elliptic curve. To make the math work for key exchange, they shared the images of specific points (called torsion points) under the secret isogeny.

* The Info: If the secret isogeny is , SIKE gave away () and () for specific basis points and .

* The Break: In 2022, Castryck and Decru showed that this auxiliary information allowed an attacker to construct a higher-dimensional abelian variety linking the public data. In this setting, the secret isogeny can be recovered efficiently using techniques based on Kani's results on isogenies between products of elliptic curves.

* The Oversight: For years, cryptanalysts thought this extra info was harmless. Related techniques existed in the algebraic geometry literature but had not previously been applied in this cryptographic context.

3.2. Why SQISign appears unaffected by the SIKE Vulnerability

SQIsign is a signature scheme in which the prover demonstrates knowledge of an isogeny through a zero-knowledge protocol. Unlike SIDH/SIKE, it does not publish images of torsion basis points under secret isogenies, specifically:

* SIKE: key exchange with auxiliary structure leakage

* SQIsign: proof/verification of isogeny knowledge without publishing auxiliary isogeny action data

The CastryckDecru attack relies critically on this auxiliary torsion-point information to construct additional structure (e.g., via abelian surfaces) that enables efficient recovery of the secret isogeny. Because SQIsign does not provide such auxiliary data, these techniques do not directly apply. Attacks would instead need to solve instances of the isogeny path problem or related problems in the endomorphism ring, for which no comparable shortcut is currently known.

4. SQIsign Algorithm Overview

4.1. Cryptographic Foundation

SQIsign is based on the hardness of finding isogenies between supersingular elliptic curves over finite fields. The security assumption relies primarily on the difficulty of the *Isogeny Path Problem*

Unlike lattice-based schemes, isogeny-based cryptography offers:

* *Smaller key and signature sizes*

* *Algebraic structure* based on elliptic curve isogenies

* *Different security assumptions* (diversification from lattice-based schemes)

4.2. Security Levels

SQIsign is defined with three parameter sets corresponding to NIST security levels:

Parameter Set	NIST Level	Public Key	Signature	Classical Sec
SQIsign-L1	I	65 bytes	148 bytes	~128 bits
SQIsign-L3	III	97 bytes	224 bytes	~192 bits
SQIsign-L5	V	129 bytes	292 bytes	~256 bits

Table 2

4.3. Performance Characteristics

* *Signing*: Computationally intensive (slower than lattice schemes)

* *Verification*: Moderate computational cost

* *Key Generation*: Intensive computation required

* *Size*: Exceptional efficiency: substantially smaller than many lattice-based alternatives at comparable security levels

Recommended Use Cases: - Sign-once, verify-many scenarios (firmware, certificates) - Bandwidth-constrained environments - Storage-limited devices - Applications where signature/key size dominates performance considerations

5. COSE Integration

This section defines the identifiers for SQIsign in COSE [RFC8152].

5.1. SQIsign Algorithms

The algorithms defined in this document are:

* SQIsign-L1: SQIsign NIST Level I (suggested value -61)

* SQIsign-L3: SQIsign NIST Level III (suggested value -62)

* SQIsign-L5: SQIsign NIST Level V (suggested value -63)

Note: The algorithm identifier values (-61, -62, -63) are requested from IANA upon working group adoption. Early implementations may use temporary values from the Private Use range (-65000 to -65535) for experimentation.

5.2. SQIsign Key Types

A new key type is defined for SQIsign with the name "SQIsign".

5.3. SQIsign Key Parameters

SQIsign keys use the following COSE Key common parameters:

Key Parameter	COSE Label	CBOR Type	Description
kty	1	int	Key type: IETF (SQIsign)
kid	2	bstr	Key ID (optional)
alg	3	int	Algorithm identifier (-61, -62, or -63)
key_ops	4	array	Key operations (sign, verify)

Table 3

5.4. SQIsign-Specific Key Parameters

Key Parameter	Label	CBOR Type	Description
pub	-1	bstr	SQIsign public key
priv	-2	bstr	SQIsign private key (sensitive)

Table 4

5.5. COSE Key Format Examples

NOTE: SQIsign keys are structurally distinct and not representable as existing EC key types due to non-classical representation.

5.5.1. Public Key (COSE_Key)

cbor { 1: IETF, / kty: SQIsign / 3: -61, / alg: SQIsign-L1 / -1: h'[PUBLIC_KEY]' / pub: SQIsign public key bytes / }

5.5.2. Private Key (COSE_Key)

cbor { 1: IETF, / kty: SQIsign / 3: -61, / alg: SQIsign-L1 / -1: h'[PUBLIC_KEY]', / pub: SQIsign public key bytes / -2: h'[PRIVATE_KEY]' / priv: SQIsign private key bytes / }

5.6. COSE Signature Format

SQIsign signatures in COSE follow the standard COSE_Sign1 structure [RFC9052]:

COSE_Sign1 = [protected: bstr .cbor header_map, unprotected: header_map, payload: bstr / nil, signature: bstr]

The signature field contains the raw SQIsign signature bytes.

5.6.1. Protected Headers

The protected header MUST include:

cbor { 1: -61 / alg: SQIsign-L1, -62 for L3, -63 for L5 / }

5.6.2. Example COSE_Sign1 Structure

```
cbor 18( / COSE_Sign1 tag / [ h'A10139003C', / protected: {"alg":
-61} / {}, / unprotected /
h'546869732069732074686520636F6E74656E742E', / payload /
h'[SQISIGN_SIGNATURE_BYTES]' / signature / ] )
```

6. JOSE Integration

6.1. JSON Web Signature (JWS) Algorithm Registration

The following algorithm identifiers are registered for use in the JWS "alg" header parameter for JSON Web Signatures [RFC7515]:

Algorithm Name	Description	Implementation Requirements
SQIsign-L1	SQIsign NIST Level I	Optional
SQIsign-L3	SQIsign NIST Level III	Optional
SQIsign-L5	SQIsign NIST Level V	Optional

Table 5

6.2. JSON Web Key (JWK) Representation

SQIsign keys are represented in JWK [RFC7517] format as follows:

6.2.1. Public Key Parameters

Parameter	Type	Description
kty	string	Key type: "SQIsign"
alg	string	Algorithm: "SQIsign-L1", "SQIsign-L3", or "SQIsign-L5"
pub	string	Base64url-encoded public key
kid	string	Key ID (optional)
use	string	Public key use: "sig"

		(optional)	
key_ops	array	Key operations: [verify] (optional)	

Table 6

6.2.2. Private Key Parameters

Private keys include all public key parameters plus:

Parameter	Type	Description
priv	string	Base64url-encoded private key

Table 7

6.3. JWK Examples

6.3.1. Public Key (JWK) Example

json { "kty": "SQIsign", "alg": "SQIsign-L1", "pub": "KxtQx8s8RcBEU67wr57K37fdPEztN4M8NUC_\ 5xZuqgMwkaeJhM94YHi_-2UsQllbnmm-W4XFSLm2hUwiMylrAh0", "kid": "2027-01-device-key", "use": "sig", "key_ops": ["verify"] }

6.3.2. Private Key (JWK) Example

json { "kty": "SQIsign", "alg": "SQIsign-L1", "pub": "KxtQx8s8RcBEU67wr57K37fdPEztN4M8NUC_\ 5xZuqgMwkaeJhM94YHi_-2UsQllbnmm-W4XFSLm2hUwiMylrAh0", "priv": "KxtQx8s8RcBEU67wr57K37fdPEztN4M8NUC_5xZuqgMwkaeJhM94YHi_\ -2UsQllbnmm-W4XFSLm2hUwiMylrAh1VwP9vNkBZH0Bjj2wc-\ p7sUgQAAAAAAAAAAAAAAAAAN68tviJbcCpQ84fh-4IJB4-\ _____P38m3fKOhfhMspQU9GmA4CD5___\ _____\ _____wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\ AAAAAAA5cP9aha40v-8mFd_bdAgpR93Ug2iPhu4_NxG97C7\ 8wBvVMGOrQTCli7NxrR2KlPZR1AC5VddGf4p-ZjCzrWfAJv\ xhEh4uOKXq1MmuS9TwZGuz1YIYMIguu1wqjdmfaQAfOmK2g\ WWO3vcld5s7GR2AcrTv65ocK_pVUWY8eJDcQA", "kid": "2027-01-device-key", "use": "sig", "key_ops": ["sign"] }

6.4. JWS Compact Serialization

A JWS using SQIsign follows the standard compact serialization:

BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload) || '.' || BASE64URL(JWS Signature)

6.4.1. Example JWS Protected Header

json { "alg": "SQIsign-L1", "typ": "JWT" }

Base64url-encoded: eyJhbGciOiJTUUlzaWduLUwxIiwidHlwIjoiSldUIn0

6.4.2. Complete JWS Example

eyJhbGciOiJTUUlzaWduLUwxIiwidHlwIjoiSldUIn0 . [BASE64URL_PAYLOAD] . [BASE64URL_SQISIGN_SIGNATURE]

7. Implementation Considerations

7.1. Signature and Key Generation

Implementations MUST follow the SQIsign specification [SQIsign-Spec] for:

* Key pair generation
* Signature generation
* Signature verification

7.2. Randomness Requirements

SQIsign signature generation requires high-quality randomness. Implementations MUST use a cryptographically secure random number generator (CSRNG) compliant with [RFC4086] or equivalent.

7.3. Side-Channel Protections

Implementations SHOULD implement protections against:

* Timing attacks
* Power analysis
* Fault injection attacks

Particularly for constrained devices deployed in physically accessible environments.

7.4. Performance Trade-offs

Implementers should be aware:

* *Signing is computationally expensive*: Consider pre-signing or batch operations
* *Verification is moderate*: Suitable for resource-constrained verifiers
* *Size is exceptional*: Minimizes bandwidth and storage

7.5. Interoperability Testing

Early implementations SHOULD participate in interoperability testing to ensure:

* Consistent signature generation and verification
* Proper encoding in COSE and JOSE formats
* Cross-platform compatibility

8. Security Considerations

8.1. Algorithm Security

The security of SQIsign relies primarily on the hardness of finding isogenies between supersingular elliptic curves.

These assumptions are *different from lattice-based schemes*, providing cryptographic diversity in the post-quantum landscape.

8.2. Quantum Security

SQIsign is designed to resist attacks by large-scale quantum computers. The three parameter sets provide security equivalent to AES-128, AES-192, and AES-256 against both classical and quantum adversaries.

8.3. Forward-Looking Quantum Risk: isogeny schemes vs lattice schemes

Isogeny schemes are not yet trusted in the same way as lattice schemes, this section is here to avoid even the appearance of tone drift toward endorsement of isogeny schemes.

No publicly known attacks applicable to the standardized parameter sets that recover full private keys under stated assumptions, but the field is evolving. SQISign's security is based on hardness assumptions different from factoring/discrete log problems, leading to the expectation it will resist known quantum attacks using Shor's algorithm [Shor-Algorithm]. Ongoing research is exploring new types of quantum algorithms that could weaken certain underlying assumptions in isogeny-based cryptography.

Organizations considering deployment should plan for long-term flexibility, including the ability to update or replace cryptographic algorithms as new information becomes available.

8.3.1. Quantum algorithms for isogeny path problems

8.3.1.1. Tani's algorithm

Tani's algorithm [Tani-Algorithm] achieves (O(N^{1/3})\ query complexity for claw-finding / quantum walk problems instead of (O(N^{1/2})). However, as analyzed by Jaques and Schanck [Jaques-Schanck-Cryptanalysis], practical implementation faces significant barriers:

* *Quantum RAM requirements:* Exponentially large qRAM with sub-nanosecond access

* *Parallelization limits:* Circuit depth constraints reduce theoretical speedup

* *Error correction overhead:* Physical qubit requirements may exceed 10^9 qubits

These constraints suggest that even with fault-tolerant quantum computers, Tani-style attacks remain impractical for security parameters used in SQIsign-L3 and SQIsign-L5.

8.4. Current Cryptanalysis Status

As of this writing, SQIsign is undergoing active cryptanalytic review:

* *NIST Round 2 evaluation*: [NIST-Finalized-Standards]

* *Academic research*: Ongoing analysis of isogeny-based cryptography

* *Known attacks*: No attacks are currently known that recover private keys for the standardized parameter sets within their claimed security levels. However, the scheme and its underlying assumptions remain under active study.

Implementers are advised: - Monitor NIST announcements and updates - Follow academic literature on isogeny cryptanalysis - Be prepared to deprecate or update as cryptanalysis evolves

8.5. Implementation Security

8.5.1. Random Number Generation

Poor randomness can completely compromise SQIsign security. Implementations MUST use robust CSRNGs, especially on constrained devices with limited entropy sources.

8.5.2. Side-Channel Resistance

Constrained devices may be physically accessible to attackers. Implementations SHOULD:

* Use constant-time algorithms where possible

* Implement countermeasures against DPA/SPA

* Consider fault attack mitigations

8.5.3. Key Management

* Private keys MUST be protected with appropriate access controls

* Consider hardware security modules (HSMs) or secure elements for key storage

* Implement key rotation policies appropriate to the deployment

8.6. Cryptographic Agility

Organizations deploying SQIsign SHOULD:

* Maintain hybrid deployments with classical algorithms during transition

* Plan for algorithm migration if cryptanalysis reveals weaknesses

* Monitor NIST and IRTF guidance on PQC deployment

8.7. Constrained Device Specific Risks

IoT devices face unique challenges:

* *Physical access*: Devices may be deployed in hostile environments

* *Limited update capability*: Firmware updates may be infrequent or impossible

* *Long deployment lifetimes*: Devices may operate for 10+ years

Design systems with: - Defense in depth (multiple security layers) - Remote update capability when possible - Graceful degradation if algorithm is compromised

9. IANA Considerations

9.1. Additions to Existing Registries

IANA is requested to add the following entries to the COSE and JOSE registries. The following completed registration actions are provided as described in [RFC9053] and [RFC9054].

9.1.1. New COSE Algorithms

IANA is requested to register the following entries in the "COSE Algorithms" registry:

Name	Value	Description	Capabilities	Change Cont	Ref	Rec'd
SQIsign-L1	-61	SQIsign NIST L I	kty	IETF	THIS-RFC	No
SQIsign-L3	-62	SQIsign NIST L III	kty	IETF	THIS-RFC	No
SQIsign-L5	-63	SQIsign NIST L V	kty	IETF	THIS-RFC	No

Table 8

9.1.2. New COSE Key Types

IANA is requested to register the following entry in the "COSE Key Types" registry:

Name	Value	Description	Capabilities	Change Cont	Ref
SQIsign	IETF	SQIsign pub key	sign, verify	IETF	THIS-RFC

Table 9

9.1.3. New COSE Key Type Parameters

IANA is requested to register the following entries in the "COSE Key Type Parameters" registry:

Key Type	Name	Label	CBOR Type	Desc	Change Cont	Reference
SQIsign	pub	-1	bstr	Public key	IETF	THIS-RFC
SQIsign	priv	-2	bstr	Private key	IETF	THIS-RFC

Table 10

9.1.4. New JWS Algorithms

IANA is requested to register the following entries in the "JSON Web Signature and Encryption Algorithms" registry:

Algorithm Name	Desc	Impl Req	Change Cont	Ref	Recommended
SQIsign-L1	SQIsign NIST L I	Optional	IETF	THIS-RFC	No
SQIsign-L3	SQIsign NIST L III	Optional	IETF	THIS-RFC	No
SQIsign-L5	SQIsign NIST L V	Optional	IETF	THIS-RFC	No

Table 11

9.1.5. New JSON Web Key Types

IANA is requested to register the following entry in the "JSON Web Key Types" registry:

"kty" Param Value	Key Type Desc	Change Cont	Reference
SQIsign	SQIsign public key	IETF	THIS-RFC

Table 12

9.1.6. New JSON Web Key Parameters

IANA is requested to register the following entries in the "JSON Web Key Parameters" registry:

Param Name	Desc	Used with "kty" Val	Change Cont	Reference
pub	Public key	SQIsign	IETF	THIS-RFC
priv	Private key	SQIsign	IETF	THIS-RFC

Table 13

10. Acknowledgments

The authors would like to thank:

* Luca DeFeo for reviewing draft-00 and providing valuable feedback. Any remaining errors are solely the responsibility of the authors.

* The SQIsign design team for groundbreaking work on isogeny-based signatures

* The NIST PQC team for managing the standardization process

* The COSE and JOSE working groups for guidance on integration

* The IRTF Crypto Forum Research Group for ongoing cryptanalytic review

* Early implementers who provide valuable feedback

This work builds upon the template established by [I-D.ietf-cose-falcon] and similar PQC integration efforts.

11. References

11.1. Normative References

Populated automatically from metadata

11.2. Informative References

Populated automatically from metadata

12. References

12.1. Normative References

[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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/rfc/rfc8152>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9054] Schaad, J., "CBOR Object Signing and Encryption (COSE): Hash Algorithms", RFC 9054, DOI 10.17487/RFC9054, August 2022, <https://www.rfc-editor.org/rfc/rfc9054>.

12.2. Informative References

[CNSA-2] National Security Agency, "Commercial National Security Algorithm Suite 2.0", May 2025, <https://media.defense.gov/2025/May/30/ 2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF>.

[I-D.ietf-cose-dilithium] Prorock, M. and O. Steele, "ML-DSA for JOSE and COSE", Work in Progress, Internet-Draft, draft-ietf-cose-dilithium-11, 15 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-dilithium-11>.

[I-D.ietf-cose-falcon] Prorock, M., Steele, O., and H. Tschofenig, "FN-DSA for JOSE and COSE", Work in Progress, Internet-Draft, draft-ietf-cose-falcon-04, 15 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-falcon-04>.

[Jaques-Schanck-Cryptanalysis] Jaques, S. and J. M. Schanck, ""Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE"", June 2019, <https://eprint.iacr.org/archive/2019/103/20190619:191423>.

[NIST-Finalized-Standards]
NIST, ""NIST Releases First 3 Finalized Post-Quantum Encryption Standards"", August 2024, <https://www.nist.gov/news-events/news/2024/08/ nist-releases-first-3-finalized-post-quantum-encryption-standards>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/rfc/rfc7049>.

[Shor-Algorithm]
Shor, P. W., ""Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer"", January 1996, <https://arxiv.org/abs/quant-ph/9508027>.

[SQIsign-Analysis]
IACR ePrint Archive, ""SQIsign: Compact Post-Quantum Signatures from Quaternions and Isogenies"", January 2021, <https://eprint.iacr.org/2020/1240>.

[SQIsign-Spec]
SQIsign team, "SQIsign: Compact Post-Quantum Signatures from Quaternions and Isogenies (Round 2)", February 2025, <https://sqisign.org/spec/sqisign-20250205.pdf>.

[Tani-Algorithm]
Tani, S., "Claw Finding Algorithms Using Quantum Walk", August 2007, <https://arxiv.org/abs/0708.2584>.

[WebAuthn-PQC-Signature-size-constraints]
University of Quantum Science, "WebAuthn PQC Signature size constraints", April 2026, <https://www.npmjs.com/package/quantum-resistant-rustykey>.

Appendix A. Test Vectors

A.1. SQIsign-L1 Test Vectors

A.1.1. Example 1: Simple Message Signing

The following test vector exhibits a SQIsign Level I signature over a short message.

Message (hex): d81c4d8d734fcbfbeade3d3f8a039faa2a2c9957e835ad55b2 \
2e75bf57bb556ac8 Message (ASCII): MsO=?*,W5U.uWUj

Public Key (hex): 07CCD21425136F6E865E497D2D4D208F0054AD81372066E \
817480787AAF7B2029550C89E892D618CE3230F23510BFBE68FCCDDAEA51DB1436 \
B462ADFAF008A010B Public Key (Base64url): \
B8zSFCUTb26GXkl9LU0gjwBUrYE3IGboF0gHh6r3s \
gKVUMieiS1hjOMjDyNRC_vmj8zdrqUdsUNrRirfrwCKAQs

Signature (hex): 84228651f271b0f39f2f19f2e8718f31ed3365ac9e5cb303 \
afe663d0cfc11f0455d891b0ca6c7e653f9ba2667730bb77befe1b1a3182840428 \
4af8fd7baacc010001d974b5ca671ff65708d8b462a5a84a1443ee9b5fed721876 \
7c9d85ceed04db0a69a2f6ec3be835b3b2624b9a0df68837ad00bcacc27d1ec806 \
a44840267471d86eff3447018adb0a6551ee8322ab30010202 Signature \
(Base64url): hCKGUfJxsPOfLxny6HGPMe0zZayeXLMDr-Zj0M_BHw \ RV2JGwymx-\
ZT-bomZ3MLt3vv4bGjGChAQoSvj9e6rMAQAB2XS1ymcf9lcI2LRipahK \
FEPum1_tchh2fJ2Fzu0E2wppovbsO-g1s7JiS5oN9og3rQC8rMJ9HsgGpEhAJnRx2G \
7_NEcBitsKZVHugyKrMAECAg

A.1.2. COSE_Sign1 Complete Example

cbor 18([h’a10139003c’, / protected: {"alg": -61} / {}, / \
unprotected / \
h’d81c4d8d734fcbfbeade3d3f8a039faa2a2c9957e835ad55b22e75bf57bb \
556ac8’, / payload / \
h’84228651f271b0f39f2f19f2e8718f31ed3365ac9e5cb303afe663d0cfc1 \
1f0455d891b0ca6c7e653f9ba2667730bb77befe1b1a31828404284af8fd7b \
aacc010001d974b5ca671ff65708d8b462a5a84a1443ee9b5fed7218767c9d \
85ceed04db0a69a2f6ec3be835b3b2624b9a0df68837ad00bcacc27d1ec806 \
a44840267471d86eff3447018adb0a6551ee8322ab30010202’])

A.1.3. JWS Complete Example

eyJhbGciOiJTUUlzaWduLUwxIiwidHlwIjoiSldUIn0 . \
2BxNjXNPy_vq3j0_igOfqiosmVfoNa1Vsi51v1e7VWrI . \
hCKGUfJxsPOfLxny6HGPMe0zZayeXLMDr-Zj0M_BHwRV2JGwymx-ZT-bomZ3MLt3vv \
4bGjGChAQoSvj9e6rMAQAB2XS1ymcf9lcI2LRipahKFEPum1_tchh2fJ2Fzu0E2wpp \
ovbsO-g1s7JiS5oN9og3rQC8rMJ9HsgGpEhAJnRx2G7_NEcBitsKZVHugyKrMAECAg

A.2. SQIsign-L3 Test Vectors

[PLACEHOLDER FOR L3 TEST VECTORS]

A.3. SQIsign-L5 Test Vectors

[PLACEHOLDER FOR L5 TEST VECTORS]

Appendix B. Implementation Status

[RFC Editor: Please remove this section before publication]

This section records the status of known implementations at the time of writing.

B.1. Open Source Implementations

B.1.1. Reference Implementation

* *Organization*: SQIsign team
* *Repository*: https://github.com/SQISign/the-sqisign
* *Language*: C
* *License*: MIT
* *Status*: Active development
* *COSE/JOSE Support*: Not yet integrated

B.1.2. Rust Implementation

* *Organization*: IETF - Community implementation
* *Repository*: IETF
* *Language*: Rust
* *License*: IETF
* *COSE Support*: Planned
* *Status*: Development

B.2. Commercial Implementations

[RFC EDITOR: To be populated as vendors implement]

B.3. Interoperability Testing

* *Test Suite Location*: IETF

* *Participating Organizations*: IETF

Appendix C. Design Rationale

C.1. Algorithm Identifier Selection

The requested algorithm identifiers (-61, -62, -63) are:

* In the range designated for experimental/informational use

* Sequential for the three parameter sets

* Not conflicting with existing registrations

* Consistent with the approach used for other PQC algorithms

C.2. Key Type Design

The SQIsign key type is intentionally simple:

* Only two parameters (pub, priv) following minimalist design

* Binary encoding (bstr) for efficiency

* No algorithm-specific encoding—raw bytes from SQIsign spec

This approach: - Minimizes CBOR encoding overhead (critical for constrained devices) - Simplifies implementation - Provides future flexibility for parameter set evolution

Appendix D. Change Log

[RFC Editor Note:** Please remove this section before publication]

D.1. draft-mott-cose-sqisign-03

* Incorporated technical corrections and feedback from Luca De Feo on draft-00

* Updated the Abstract and Introduction to utilize more neutral, objective language

* Removed vendor-specific branding in favor of generic cryptographic terminology

* fixed various formatting issues

Author's Address

Antony R. Mott
RustyKey
United States of America
Email: antony@rustykey.io