COSE
Internet-Draft
Intended status: Standards Track
Expires: 6 December 2026

A. R. Mott
RustyKey速

4 June 2026

CBOR Object Signing and Encryption (COSE) and JSON Object Signing and Encryption (JOSE) Registrations for SQIsign
draft-mott-cose-sqisign-06

Abstract

NOTE: This document describes a signature scheme based on an algorithm currently under evaluation in the 3rd round [NIST-3rd-round-candidates] 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 the most compact signature and public key sizes of any candidate in the NIST Post-Quantum Cryptography (PQC) standardization and on-ramp-to-standardization processes.

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.

Depending on authenticator implementation, transport (USB/NFC/BLE) and message fragmentation support, some deployments of CTAP2-based authenticators enforce limits near 1024 bytes for external key communication, and some standardized post-quantum signature schemes increase message sizes and may stress constrained authenticators or transports. As a result CBOR-encoded messages may hit 7609-byte limit in some authenticators. SQIsign-L1, L3 and L5 signatures are small enough to enable delivery over constrained networks like 802.15.4 and may be more suitable for constrained networks due to smaller signature sizes.

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寰泥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/https://github.com/antonymott/quantum-resistant-rustykey.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 December 2026.

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.

Falcon (NIST secondary) 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-Standard] [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 3 [NIST-3rd-round-candidates] Additional Digital Signature Schemes, not yet a NIST standard.

Speed: SQIsign is significantly slower at signing (roughly 100x to 1000x) compared to ML-DSA, though the math is changing fast and variants improve this.

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 Fits < 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 ~6.25 billion units.

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. Cryptanalytic Resistance: SIDH/SIKE Attacks Do Not Apply

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 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. 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	Quantum Security (estimated)
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

4.4. SQIsign Variants and the Post-SIKE Landscape

While the SQIsign team initially focused on improving the core algorithm, the 2022 SIKE vulnerability catalyzed broader research into higher-dimensional algebraic geometry, particularly investigating improvements to key and signature generation speed—widely viewed as implementation bottlenecks. This interest has sparked an evolution of SQIsign variants, all still based on the baseline algorithm currently competing in NIST's Round 3. Remarkably, two independent groups published dimension-2 variants on the same day (May 13, 2024), with a third appearing the following day—demonstrating the rapid, simultaneous evolution of the field following the 2022 SIKE breakthrough. Given this dynamic environment, readers interested in SQIsign's future will benefit from this summary, which we intend to update with each revision of this standards-track submission.

The key takeaway is that researchers have repurposed the higher-dimensional techniques from the SIKE cryptanalysis to optimize SQIsign variants with faster signing and potentially smaller sizes, while each group attempts to maintain equivalent post-quantum security levels.

Variants can be classified primarily by the geometric dimensions they employ:

4.4.1. Core SQIsign (Dimension 1)

The baseline algorithm currently competing in NIST's Round 3. The SQIsign team, in cooperation with IBM researchers, actively maintains and tunes this version. Recent updates focus on reducing memory footprints and accelerating core algebraic operations for practical implementation. However, NIST's current process permits only minor "tweaks" rather than substantial algorithmic changes.

4.4.2. Multi-dimensional variants

* SQIsignHD [SQIsignHD] dramatically shrunk signature sizes, simplified verification.

* SQIsign2D-West [SQIsign2D-West] prioritized a rigorous security proof over raw speed.

* SQIsign2D-East [SQIsign2D-East] fast 2D verification using a generalized random isogeny algorithm.

* SQIPrime [SQIPrime]: Offers two sub-variants with different dimension trade-offs:

- SQIPrime2D: Uses only dimension 2 non-smooth challenge isogenies, avoiding the dimension 4 computations required by SQIsignHD. More efficient while remaining highly compact compared to non-isogeny PQC schemes.

- SQIPrime4D: Uses dimension 4 isogenies for response representation, prioritizing maximum compactness at the cost of exponentially higher runtime. Despite the paper's title, this sub-variant represents the authors' exploration before settling on the 2D approach.

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)

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

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___\ _____\ _____wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\ 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-Standard] 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

7.6. Performance testing under real-world scenarios

* public metrics, interoperability and performance testing of the proposed WASM versions can be evaluated on a live testbed [PQC-Testbed].

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. Cryptanalysis and Algorithm Maturity

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

* *NIST Round 3 evaluation*: [NIST-3rd-round-candidates]

* *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

8.4. Implementation Security

8.4.1. Random Number Generation

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

8.4.2. 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

8.4.3. 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

8.5. 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

8.6. 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

* Aerospace and constrained-telemetry engineers/contractors who suggested the idea for pqc.rustykey.me, a public testbed devoted to anyone wishing to testout, evaluate and critique actual working WASM implemented code of all three levels.

* 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>.

[NIST-3rd-round-candidates] NIST, "Nine Candidates Advance to the Third Round of the Additional Digital Signatures for the PQC Standardization Process", May 2026, <https://csrc.nist.gov/News/2026/nist-advances-9-candidates-to-the-3rd-round-of-pqc>.

[PQC-Testbed] RustyKey, "PQC RustyKey Testbed", June 2026, <https://pqc.rustykey.me>.

[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>.

[SQIPrime] Max Duparc, Tako Boris Fouotsa, "SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies", May 2024, <https://eprint.iacr.org/2024/773>.

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

[SQIsign-Standard]
SQIsign team, "Algorithmspecifications andsupporting documentation Version 2.0.1", July 2025, <https://sqisign.org/spec/sqisign-20250707.pdf>.

[SQIsign2D-East]
Kohei Nakagawa, Hiroshi Onuki, "SQIsign2D-East: A New Signature Scheme Using 2-dimensional Isogenies", May 2024, <https://eprint.iacr.org/2024/771>.

[SQIsign2D-West]
Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, Benjamin Wesolowski, "SQIsign2D-West: The Fast, the Small, and the Safer", May 2024, <https://eprint.iacr.org/2024/760>.

[SQIsignHD]
Pierrick Dartois, Antonin Leroux, Damien Robert, Benjamin Wesolowski, "SQISignHD: New Dimensions in Cryptography", May 2023, <https://eprint.iacr.org/2023/436>.

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

Appendix A. Test Vectors

Vectors use NIST KAT count = 0 from upstream SQIsign response files (PQCsignKAT_*_SQIsign_lvl*.rsp). The same 32-byte message appears at each security level so implementers can compare keys and signatures. Algorithm identifiers -61, -62, and -63 map to SQIsign-L1, SQIsign-L3, and SQIsign-L5 respectively.

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

A.2.1. Example 1: Simple Message Signing

The following test vector exhibits a SQIsign Level III signature over a short message (NIST KAT count = 0; COSE/JOSE algorithm -62).

Message (hex): \
D81C4D8D734FCBFBEADE3D3F8A039FAA2A2C9957E835AD55B22E75BF \ 57BB556AC8 \
Message (ASCII): MsO=?*,W5U.uWUj

```
   Public Key (hex):
   C32377D6F6D70729884A7F6877EF4791E35D21F751A3E96DE23F9 \
   A7A3C01BCD8A5F146DC19E4E2AC63007457F97D8A40EE84AEE7564CA9A7FBE6200FD3E5
   \
   E55901BFC60EB25C50D39F5C91C96510556BAA22028DF76360841721A601D65E8D0F06
   Public Key (Base64url): wyN31vbXBymISn9od-9HkeNdIfdRo-lt4j-
   aejwBvNil8Ub \
   cGeTirGMAdFf5fYpA7oSu51ZMqaf75iAP0-XlWQG_xg6yXFDTn1yRyWUQVWuqIgKN92NghB
   \ chpgHWXo0PBg

   Signature (hex):
   0868CFBF275B8E7B19BF597D658D62CC913B9B2933E30A297288FB \
   E687F6F6B8AC8AF7AA007F191386BB1A203CDDBC2BDB42792D05DA69A4507073D12B0BD
   \
   C47E2B36BC4BA45C68791918281E578F2DC14294504726DCD4CA4C4565FBB89A1280004
   \
   8C7B84746A2CBD8247248E248B70B51AE91994957857692A028D8F5CABABFC91E4BF1C5
   \
   D350219A0189C57DE4A7710D29E0364C79B2188449EC0397359430D594C7B5980CC6755
   \
   1933A902D3C11F0FBD6DC39711D3E1F501159EE7FB85CE81B4CE24E1016006567DF4693
   \ 15D513E73F69F6301664E6449AF9DCEB4000D15 Signature (Base64url):
   CGjPvydbjnsZv1l9ZY1izJE7mykz4wopcoj75of29risiveq \
   AH8ZE4a7GiA83bwr20J5LQXaaaRQcHPRKwvcR-
   Kza8S6RcaHkZGCgeV48twUKUUEcm3NTKT \ EVl-
   7iaEoAASMe4R0aiy9gkckjiSLcLUa6RmUlXhXaSoCjY9cq6v8keS_HF01AhmgGJxX3k \
   p3ENKeA2THmyGIRJ7AOXNZQw1ZTHtZgMxnVRkzqQLTwR8PvW3DlxHT4fUBFZ7n-
   4XOgbTOJ \ OEBYAZWffRpMV1RPnP2n2MBZk5kSa-dzrQADRU
```

A.2.2. COSE_Sign1 Complete Example

```
   cbor 18( [ h’a10139003d’, / protected: {"alg": -62} / {}, /
   unprotected / h’d81c4d8d734fcbfbeade3d3f8a039faa2a2c995 \
   7e835ad55b22e75bf57bb556ac8’, / payload /
   h’0868cfbf275b8e7b19bf597d658d62cc913b9b2 \
   933e30a297288fbe687f6f6b8ac8af7aa007f191386bb1a203cddbc2bdb42792 \
   d05da69a4507073d12b0bdc47e2b36bc4ba45c68791918281e578f2dc1429450 \
   4726dcd4ca4c4565fbb89a12800048c7b84746a2cbd8247248e248b70b51ae91 \
   994957857692a028d8f5cababfc91e4bf1c5d350219a0189c57de4a7710d29e0 \
   364c79b2188449ec0397359430d594c7b5980cc67551933a902d3c11f0fbd6dc \
   39711d3e1f501159ee7fb85ce81b4ce24e1016006567df469315d513e73f69f6 \
   301664e6449af9dceb4000d15’, / signature / ] )
```

A.2.3. JWS Complete Example

```
eyJhbGciOiJTUUlzaWduLUwzIiwidHlwIjoiSldUIn0 .
2BxNjXNPy_vq3j0_igOfqiosmVfoNa1Vsi51v1e7VWrI .
CGjPvydbjnsZv1l9ZY1izJE7mykz4wopcoj75of29risiveqAH8ZE4a7GiA83bwr20J5LQXa
\ aaRQcHPRKwvcR-Kza8S6RcaHkZGCgeV48twUKUUEcm3NTKTEVl-
7iaEoAASMe4R0aiy9gkck \
jiSLcLUa6RmUlXhXaSoCjY9cq6v8keS_HF01AhmgGJxX3kp3ENKeA2THmyGIRJ7AOXNZQw1Z
\ THtZgMxnVRkzqQLTwR8PvW3DlxHT4fUBFZ7n-
4XOgbTOJOEBYAZWffRpMV1RPnP2n2MBZk5k \ Sa-dzrQADRU
```

A.3. SQIsign-L5 Test Vectors

A.3.1. Example 1: Simple Message Signing

The following test vector exhibits a SQIsign Level V signature over a short message (NIST KAT count = 0; COSE/JOSE algorithm -63).

Message (hex):

```
D81C4D8D734FCBFBEADE3D3F8A039FAA2A2C9957E835AD55B22E75BF \ 57BB556AC8
```

Message (ASCII): `MsO=?*,W5U.uWUj`

Public Key (hex):

```
86FFA3B0F73D55A64D13C6F89F28D75FD17C5E2368E1D451127C1 \
6D1A97CDB440E20333A233AD2F8E4D70187C8AE31602049ADE949A87F95E79DA4C456F5
\
D400B2485A96D04708A2F30046812B8D65A3BFBFDED0DD6563462F9E2BCE760CD753CAE
\ 8471BEC7049EF28FFEFE859C15DAC49DB959AEE99842D97A380A70DD7330106
```

Public Key (Base64url):

```
hv-
jsPc9VaZNE8b4nyjXX9F8XiNo4dRREnwW0al820QOIDM \ 6IzrS-
OTXAYfIrjFgIEmt6Umof5XnnaTEVvXUALJIWpbQRwii8wBGgSuNZaO_v97Q3WVjRi \ -
eK852DNdTyuhHG-xwSe8o_-_oWcFdrEnblZrumYQtl6OApw3XMwEG
```

Signature (hex):

```
6B8EF5D7689A1EA1CFCE9C6F7495E309E9D1D1B03E61CD97088E67 \
9C4901D0B6B6D38217F4AED6C44949B41F9AF80B43E84D0C91BDB1D00E06957BEBF30A5
\
8012AD01E52CF7906CE197AD06696F7FCF756908EA980549E7C215D089BDE7117799F62
\
8817A1B9C8FB7FEBFF7E9D9B776142460CFAAFC97D48A57E09E0DA378401000229CC8E1
\
B94E1F2F8AFDC42066BEACE076E3E70DD01F90C4D01DAC17BEC58743532848D438A87A5
\
74D9DB940C17236AE3566281E27A99EFE5EE26E05B88A1D610A80B3AF38267D845C7FE3
\
30F199B43794A9B2E14846924127366B8F6A1F0F24D3C4B54D79DBB61B098BF32D98EA8
\
819F7BE4A5FFBA29E88B1A996C6CDFD32B048BC2ACFFA28870181447FCC8B6F97B63C47
```

```
   \ CB013C6F3D84CBD07619A5C355B000911 Signature (Base64url):
   a47112iaHqHPzpxvdJXjCenR0bA-Yc2XCI5nnEkB0La204IX \ 9K7WxElJtB-
   a-AtD6E0Mkb2x0A4GlXvr8wpYASrQHlLPeQbOGXrQZpb3_PdWkI6pgFSefCF \
   dCJvecRd5n2KIF6G5yPt_6_9-nZt3YUJGDPqvyX1IpX4J4No3hAEAAinMjhuU4fL4r9xCBm
   \
   vqzgduPnDdAfkMTQHawXvsWHQ1MoSNQ4qHpXTZ25QMFyNq41ZigeJ6me_l7ibgW4ih1hCoC
   \ zrzgmfYRcf-
   Mw8Zm0N5SpsuFIRpJBJzZrj2ofDyTTxLVNedu2GwmL8y2Y6ogZ975KX_uino \
   ixqZbGzf0ysEi8Ks_6KIcBgUR_zItvl7Y8R8sBPG89hMvQdhmlw1WwAJEQ
```

A.3.2. COSE_Sign1 Complete Example

```
   cbor 18( [ h'a10139003e', / protected: {"alg": -63} / {}, /
   unprotected / h'd81c4d8d734fcbfbeade3d3f8a039faa2a2c995 \
   7e835ad55b22e75bf57bb556ac8', / payload /
   h'6b8ef5d7689a1ea1cfce9c6f7495e309e9d1d1b \
   03e61cd97088e679c4901d0b6b6d38217f4aed6c44949b41f9af80b43e84d0c9 \
   1bdb1d00e06957bebf30a58012ad01e52cf7906ce197ad06696f7fcf756908ea \
   980549e7c215d089bde7117799f628817a1b9c8fb7febff7e9d9b776142460cf \
   aafc97d48a57e09e0da378401000229cc8e1b94e1f2f8afdc42066beace076e3 \
   e70dd01f90c4d01dac17bec58743532848d438a87a574d9db940c17236ae3566 \
   281e27a99efe5ee26e05b88a1d610a80b3af38267d845c7fe330f199b43794a9 \
   b2e14846924127366b8f6a1f0f24d3c4b54d79dbb61b098bf32d98ea8819f7be \
   4a5ffba29e88b1a996c6cdfd32b048bc2acffa28870181447fcc8b6f97b63c47 \
   cb013c6f3d84cbd07619a5c355b000911', / signature / ] )
```

A.3.3. JWS Complete Example

```
   eyJhbGciOiJTUUlzaWduLUw1IiwidHlwIjoiSldUIn0 .
   2BxNjXNPy_vq3j0_igOfqiosmVfoNa1Vsi51v1e7VWrI .
   a47112iaHqHPzpxvdJXjCenR0bA-Yc2XCI5nnEkB0La204IX9K7WxElJtB-
   a-AtD6E0Mkb2x \
   0A4GlXvr8wpYASrQHlLPeQbOGXrQZpb3_PdWkI6pgFSefCFdCJvecRd5n2KIF6G5yPt_6_9-
   \
   nZt3YUJGDPqvyX1IpX4J4No3hAEAAinMjhuU4fL4r9xCBmvqzgduPnDdAfkMTQHawXvsWHQ1
   \ MoSNQ4qHpXTZ25QMFyNq41ZigeJ6me_l7ibgW4ih1hCoCzrzgmfYRcf-
   Mw8Zm0N5SpsuFIRp \
   JBJzZrj2ofDyTTxLVNedu2GwmL8y2Y6ogZ975KX_uinoixqZbGzf0ysEi8Ks_6KIcBgUR_zI
   \ tvl7Y8R8sBPG89hMvQdhmlw1WwAJEQ
```

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 Standards Action range (-255 to -1) per RFC 9053

* Sequential for the three parameter sets

* Not conflicting with existing registrations (verified against IANA COSE registry)

* 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-06

* fixed typos

D.2. draft-mott-cose-sqisign versions prior to -06

* added section "SQIsign Variants and the Post-SIKE Landscape"

* Incorporated technical corrections and feedback from Luca De Feo

* 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

* Added SQIsign-L3 and SQIsign-L5 COSE_Sign1 and JWS test vectors (algorithms -62 and -63)

* Documented NIST KAT count = 0 byte values for cross-implementation checks

* added informational resource for interactive working code public testbed

* updated after SQISign advances to NIST round 3 with 8 other candidates

Author’s Address

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