

Post-Quantum Cryptography Algorithm Cheat Sheets

SYMBOLS

General color coding

Best to Worst: ■ ■ ■ ■ ■ TBD: ■

Symbols

- | | |
|--|-----------------------------|
| ⓘ Recommendations differ depending on organization | 🔑 Key Generation |
| ⚙️ Parameter set | 🔒 Decryption |
| 🔒 Encryption / Ciphertext | ❓ Verification |
| ✍️ Signing / Signature | ⌚ CPU Cycles |
| 📅 Not yet standardized by NIST | 🌀 Implementation complexity |
| 🔗 Implementation Code | |
| 📏 Implementation size | |

Security categories of parameter sets

V IV III II I NIST Security Categories V, IV, III, II, I

Higher is more secure.

Implementation complexity and size

L_C M_C H_C Low/Medium/High implementation complexity
L_S M_S H_S Low/Medium/High implementation size

Lower is better.

Rating scales for parameter sizes and performance

Best to Worst: ■ $n \leq 2$, ■ $n \in \{3, 4\}$, ■ $n \in \{5, 6\}$, ■ $n \in \{7, 8\}$, ■ $n \geq 9$

- | | |
|---------------|---|
| $\frac{n}{2}$ | $\mathcal{O}(5^n)$ CPU kilo cycles for key generation |
| $\frac{n}{2}$ | $\mathcal{O}(5^n)$ CPU kilo cycles for signing |
| $\frac{n}{2}$ | $\mathcal{O}(5^n)$ CPU kilo cycles for signature verification |
| $\frac{n}{2}$ | $\mathcal{O}(5^n)$ CPU kilo cycles for encryption / key encapsulation |
| $\frac{n}{2}$ | $\mathcal{O}(5^n)$ CPU kilo cycles for decryption / key decapsulation |
| $\frac{n}{2}$ | $\mathcal{O}(2^n)$ KB of signature size |
| $\frac{n}{2}$ | $\mathcal{O}(2^n)$ KB of ciphertext size |
| $\frac{n}{2}$ | $\mathcal{O}(2^{(n-5)})$ KB of signature algorithm public key size |
| $\frac{n}{2}$ | $\mathcal{O}(2^n)$ KB of encryption algorithm public key size |

HOW TO INTERPRET THIS CHEAT SHEET

The goal of this cheat sheet is to make it as easy as possible to figure out which algorithm to pick for a given use case. Algorithm ID cards break down algorithm parameter sets, their important values and performance characteristics. The cheat sheet is intended to help users primarily in technical roles, such as engineers, architects or software developers working with post-quantum cryptography.

The focus is to avoid giving specific numbers measured in bits, bytes or cycles as this makes comparing numbers across algorithms difficult. Instead, this complexity is simplified by only providing a **color-coded number indicating the order of magnitude of each metric**.

This approach prioritizes easy interpretation and comparability of metrics and in general quick informational gain over absolute precision of data – remember this is a cheat sheet, not a standard! This document is not intended to replace the study of algorithm specifications. It just aims to point you in the right direction quickly.

The approach of focusing on orders of magnitude walks a fine line between treating too many things as “equal” and not simplifying things enough to be easy to read and compare. “In the same order of magnitude” usually refers to “equal up to a factor of at most 10”, which is a very coarse way of comparing numbers. Treating metrics that differ by a factor of e.g. 9.9 as “equal” because $9.9 < 10$ paints a distorted picture. In cryptography, factors of 5 or even 2 can make a significant difference in performance, both in theory and in practice. In order to still tease out the differences in metrics without throwing too many things together that actually differ significantly, this cheat sheet applies different scaling and “orders of magnitude” (i.e., not regarding base 10) for different metrics.

It turns out that for **metrics measured in (kilo) CPU cycle counts, i.e. algorithm performance**, “up to a factor of 5” is a scale that is granular enough to work out the differences between algorithms while maintaining easy comparability. Those cycle counts heavily depend on the CPU used during measurement, hence the numbers need to be taken with a grain of salt, even if given exactly and not in terms of orders of magnitude.

For **signature and ciphertext sizes as well as key sizes, measuring numbers in kilobytes “up to a factor of 2”** is well suited to work out the differences between algorithms while allowing for quick comparison. Specifically for signature public key sizes only, we offset the corresponding color coding by 5 orders of magnitude. This is because SLH-DSA has extremely small public keys compared to all other signature algorithms, which would extend the scale into negative numbers (e.g. for SLH-DSA-SHA2-128s, the public key has $32=2^5$ bytes, which corresponds to an order of magnitude of -5 when measuring in orders magnitude of factor 2 and in kilobytes). This phenomenon of algorithm metrics spanning a very large range of orders of (base 2) magnitudes does not occur to this extent for encryption algorithms, making an offset unnecessary.

All values thus have a lower bound of 0. We do not limit the upper end of scales, but don’t distinguish values greater than 10 anymore in terms of color coding. Please refer to the definitions on the left for symbol explanations, color coding and interpretation of numeric values.

Post-Quantum Cryptography Algorithm Cheat Sheets

ALGORITHM CHOICE GUIDANCE: RULES OF THUMB

The following orders of preference should be considered as general rules of thumb applicable to most use cases. They should however not be considered to be universal truths. Unstandardized algorithms are listed for the sake of completeness.

SIGNATURE ALGORITHMS

Objective: Best All-Around Package **TBV**



Objective: Performance of Key Generation **TBV**



Objective: Performance of Signing **TBV**



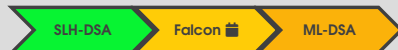
Objective: Performance of Verification **TBV**



Objective: Small Signatures **TBV**



Objective: Small Public Keys **TBV**



* XMSS/LMS is not suitable for general purpose signatures

** Security still unclear, none recommended yet at the moment

ENCRYPTION/KEY ENCAPSULATION ALGORITHMS

Objective: Best All-Around Package **TBV**



Objective: Performance of Key Generation **TBV**



Objective: Performance of Encryption **TBV**



Objective: Performance of Decryption **TBV**














Objective: Small Ciphertext **TBV**



Objective: Small Public Keys **TBV**



SECURITY CATEGORY CHOICES

- First, consider using  as a baseline.
- Use  or  for more security if possible (i.e., if a decrease in performance is not a concern and if no constraints apply).
- Use  or  if and only if  or higher is not an option due to constraints (e.g. performance, memory, etc.).
- Categories and their corresponding classical security level in bits:  = 64 bits,  = 85 bits,  = 96 bits and  =  = 128 bits.

PURE VS. PRE-HASHING

- First, consider using pure (i.e., without pre-hashing) as this is the general recommendation.
- Pre-Hashing may be considered if one or more of the following applies:
 - The message M is too large to be sent to cryptographic module (CM) for hashing without significantly impacting performance. This may be the case e.g. in CMS related use cases such as S/MIME or code signing, or in cases of very narrow communication channels to the CM (e.g. between APDUs exchanged between smartcard and smartcard reader).
 - The hash needs to be signed with different algorithms and would be computed repeatedly without pre-hashing.
 - The specific hash function is not supported in a CM.

PURE PQC vs PQ/T HYBRID

This topic depends on too many factors (e.g. cost of migration, security considerations, risk profile, GRC requirements) to give general advice. Those aspects will differ greatly between different organizations. The main reasons to adopt PQ/T hybrids are to still have traditional algorithms in place in case a new algorithm turns out to be insecure, and that PQ/T might help to avoid a big bang migration as systems could simply ignore the PQC component if they do not support it yet. Consider recommendations from different government agencies: BSI and ANSSI recommend PQ/T hybrid strategies, whereas NIST is more reserved towards PQ/T. **If using PQ/T hybrids, preferably use ECC (e.g. secp256r1, brainpoolP256r1, Curve25519) instead of RSA for the traditional component to keep key sizes as small as possible.**

Post-Quantum Cryptography Algorithm Cheat Sheets

SIGNATURE ALGORITHM OVERVIEW & ID CARDS

ML-DSA (MODULE-LATTICE-BASED DIGITAL SIGNATURE ALGORITHM)

Unusable

Poor

Good

Perfect

1-3

4-5

6-8

9-10

6.55

Algorithm Overall Usability Score

PREVIOUS NAME

CRYSTALS-DILITHIUM

SPECIFICATION

[FIPS 204](#)

TYPE

Signature

FAMILY

Lattice

SUITABLE FOR GENERAL USE

Yes

STANDARDIZATION STATUS

Standardized

RECOMMENDED BY

NIST, BSI, ANSSI

HASHING

Pure, Pre-Hashing

NAMING

by $k \times l$ matrix A
(e.g. $6 \times 5 \rightarrow$ ML-DSA-65)

IMPLEMENTATION

?

C

CP

?

C

CP

?

C

CP

COMPLEXITY

?

C

CP

?

C

CP

?

C

CP

SIZE

?

S

CP

?

S

CP

?

S

CP



PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE			SIGNATURE SIZE	PUBLIC KEY SIZE	SUITABLE PRE-HASHING
ML-DSA-44	2.16.840.1.101.3.4.3.17	II	3	3	2	1	5	SHA-256, SHA3-256
ML-DSA-65	2.16.840.1.101.3.4.3.18	III	3	3	3	1	5	SHA-384, SHA3-384
ML-DSA-87	2.16.840.1.101.3.4.3.19	V	4	4	3	2	6	SHA-512, SHA3-512

👍 / Pros / Use If:

- You need a general-purpose signature algorithm with decent specs in all categories

👎 / Cons / Don't Use If:

- You don't want a lattice-based algorithm

© MARIO SCHIENER  

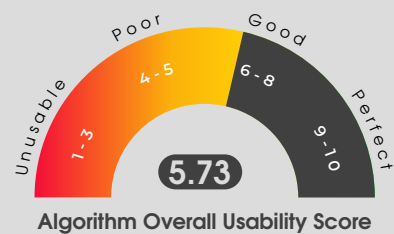
01 SEPTEMBER, 2024 – v0.2 – DRAFT

NO RESPONSIBILITY IS TAKEN FOR THE CORRECTNESS OF THIS INFORMATION.

3

Post-Quantum Cryptography Algorithm Cheat Sheets

FALCON (FAST-FOURIER LATTICE-BASED COMPACT SIGNATURES OVER NTRU)



PREVIOUS NAME
SPECIFICATION
TYPE
FAMILY
SUITABLE FOR GENERAL USE
STANDARDIZATION STATUS
RECOMMENDED BY
HASHING
NAMING

Falcon
[Project Page](#)
Signature
Lattice
Yes
Pending
TBD
TBD
TBD

IMPLEMENTATION			
COMPLEXITY			
SIZE			

PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE	SIGNATURE SIZE	PUBLIC KEY SIZE	SUITABLE PRE-HASHING
FALCON-512	TBD					TBD
FALCON-1024	TBD					TBD

👍 / Pros / Use If:

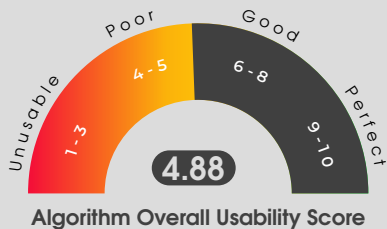
- Falcon has smaller signature sizes than ML-DSA:
 vs. resp.
- Falcon has smaller public key sizes than ML-DSA:
 vs. on Level I, vs. on Level V.

👎 / Cons / Don't Use If:

- You need a medium security category between and
- The algorithm is not yet standardized
- The algorithm requires expensive floating point arithmetic
- Key generation and signing are slower than for ML-DSA

Post-Quantum Cryptography Algorithm Cheat Sheets

SLH-DSA (STATELESS HASH-BASED DIGITAL SIGNATURE STANDARD)



PREVIOUS NAME SPHINCS+
SPECIFICATION [FIPS 205](#)
TYPE Signature
FAMILY Hash (stateless)
SUITABLE FOR GENERAL USE Yes
STANDARDIZATION STATUS Standardized
RECOMMENDED BY NIST, BSI, ANSSI
HASHING Pure, Pre-Hashing
NAMING based on various characteristics (*s=small signatures, *f=fast)

IMPLEMENTATION			
COMPLEXITY			
SIZE			

PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE	SIGNATURE SIZE	PUBLIC KEY SIZE	SUITABLE PRE-HASHING
SLH-DSA-SHA2-128s	2.16.840.1.101.3.4.3.20					SHA-256, SHA3-256
SLH-DSA-SHA2-128f	2.16.840.1.101.3.4.3.21					SHA-256, SHA3-256
SLH-DSA-SHA2-192s	2.16.840.1.101.3.4.3.22					SHA-384, SHA3-384
SLH-DSA-SHA2-192f	2.16.840.1.101.3.4.3.23					SHA-384, SHA3-384
SLH-DSA-SHA2-256s	2.16.840.1.101.3.4.3.24					SHA-512, SHA3-512
SLH-DSA-SHA2-256f	2.16.840.1.101.3.4.3.25					SHA-512, SHA3-512
SLH-DSA-SHAKE-128s	2.16.840.1.101.3.4.3.26					SHA-256, SHA3-256
SLH-DSA-SHAKE-128f	2.16.840.1.101.3.4.3.27					SHA-256, SHA3-256
SLH-DSA-SHAKE-192s	2.16.840.1.101.3.4.3.28					SHA-384, SHA3-384
SLH-DSA-SHAKE-192f	2.16.840.1.101.3.4.3.29					SHA-384, SHA3-384
SLH-DSA-SHAKE-256s	2.16.840.1.101.3.4.3.30					SHA-512, SHA3-512
SLH-DSA-SHAKE-256f	2.16.840.1.101.3.4.3.31					SHA-512, SHA3-512

👍 / Pros / Use If:

- Alternative to ML-DSA and Falcon that is not based on lattices
- Very small public keys

👎 / Cons / Don't Use If:

- Poor key generation and signing performance compared to other algorithms
- High complexity of the algorithm and the implementation
- Possible interoperability issues due to the many variants that may not all be supported everywhere

Post-Quantum Cryptography Algorithm Cheat Sheets

XMSS / XMSS-MT (eXTENDED MERKLE SIGNATURE SCHEME / eXTENDED MERKLE SIGNATURE SCHEME MULTI TREE)



PREVIOUS NAME XMSS/XMSSMT
SPECIFICATION [SP 800-208](#), [RFC 8391](#)
TYPE Signature
FAMILY Merkle Trees (stateful hash trees)
SUITABLE FOR GENERAL USE No
STANDARDIZATION STATUS Standardized
RECOMMENDED BY NIST, BSI, ANSSI
HASHING TBD
NAMING XMSS-[Hashfamily]_[h]_[n]
XMSSMT-[Hashfamily]_[h]/[d]_[n]
where h is the tree height,
d is the number of layers, and
n is the message length in bits

IMPLEMENTATION			
COMPLEXITY			
SIZE			

PARAMETER SET	NUMERIC IDENTIFIER	SECURITY CATEGORY	PERFORMANCE	SIGNATURE SIZE	MAXIMUM SIGNATURES	NUMBER OF LAYERS
XMSS-SHA2_10_256	0x00000001				2 ¹⁰	1
XMSS-SHA2_16_256	0x00000002				2 ¹⁶	1
XMSS-SHA2_20_256	0x00000003				2 ²⁰	1
XMSSMT-SHA2_20/2_256	0x00000001				2 ²⁰	2
XMSSMT-SHA2_20/4_256	0x00000002				2 ²⁰	4
XMSSMT-SHA2_40/2_256	0x00000003				2 ⁴⁰	2
XMSSMT-SHA2_40/4_256	0x00000004				2 ⁴⁰	4
XMSSMT-SHA2_40/8_256	0x00000005				2 ⁴⁰	8
XMSSMT-SHA2_60/3_256	0x00000006				2 ⁶⁰	3
XMSSMT-SHA2_60/6_256	0x00000007				2 ⁶⁰	6
XMSSMT-SHA2_60/12_256	0x00000008				2 ⁶⁰	12

NOTE:
[SP 800-208](#) defines further parameter sets not listed in [RFC 8391](#) using other hash functions (SHA256/192, SHAKE256/256, SHAKE256/192). Furthermore, [RFC 8391](#) lists optional parameter sets that are not approved in [SP 800-208](#). All of those variants are omitted here as they are not likely to be widely used, in particular not after ML-DSA and SLH-DSA have been standardized in the meantime.

👍 / Pros / Use If:

- You can predict the maximum number of signatures that are going to be required
- Firmware signing use cases
- You want a signature scheme where the security only relies on the security of the hash function used without assuming the hardness of another mathematical problem.
- Cf. [SP 800-208, Section 1.1](#) for additional explanations

👎 / Cons / Don't Use If:

- You require an algorithm for general use
- You cannot predict the maximum number of signatures that are going to be required, or the number of required signatures exceeds the maximum number of signatures enabled through the approved parameter sets
- Your application does not allow for the careful state management and tracking of signatures performed that is required with this algorithm

Post-Quantum Cryptography Algorithm Cheat Sheets

LMS/HSS (LEIGHTON-MICALI SIGNATURE / HIERARCHICAL SIGNATURE SYSTEM)



PREVIOUS NAME LMS, LMS/HSS
SPECIFICATION [SP 800-208](#), [RFC 8554](#)
TYPE Signature
FAMILY Merkle Trees (stateful hash trees)
SUITABLE FOR GENERAL USE No
STANDARDIZATION STATUS Standardized
RECOMMENDED BY NIST, BSI, ANSSI
HASHING TBD
NAMING LMS_SHA256_M32_[h]
where h is the tree height

IMPLEMENTATION			
COMPLEXITY			
SIZE			

PARAMETER SET	NUMERIC IDENTIFIER	SECURITY CATEGORY	PERFORMANCE	SIGNATURE SIZE	MAXIMUM SIGNATURES
LMS_SHA256_M32_H5	0x00000005				2 ⁵
LMS_SHA256_M32_H10	0x00000006				2 ¹⁰
LMS_SHA256_M32_H15	0x00000007				2 ¹⁵
LMS_SHA256_M32_H20	0x00000009				2 ²⁰
LMS_SHA256_M32_H25	0x00000009				2 ²⁵

NOTE: [SP 800-208](#) defines further parameter sets not listed in [RFC 8554](#) using other hash functions (SHA256/192, SHAKE256/256, SHAKE256/192). Those variants are omitted here as they are not likely to be widely used, in particular not after ML-DSA and SLH-DSA have been standardized in the meantime. Similar to XMSS multi-tree variants, a comparable structure exists with Hierarchical Signature System (HSS) variants of LMS that involves a number of LMS instances.

👍 / Pros / Use If:

- You can predict the maximum number of signatures that are going to be required
- Firmware signing use cases
- You want a signature scheme where the security only relies on the security of the hash function used without assuming the hardness of another mathematical problem.
- Cf. [SP 800-208, Section 1.1](#) for additional explanations

👎 / Cons / Don't Use If:

- You require an algorithm for general use
- You cannot predict the maximum number of signatures that are going to be required, or the number of required signatures exceeds the maximum number of signatures enabled through the approved parameter sets
- Your application does not allow for the careful state management and tracking of signatures performed that is required with this algorithm

Post-Quantum Cryptography Algorithm Cheat Sheets

🔒 ENCRYPTION ALGORITHM OVERVIEW & ID CARDS

ML-KEM (MODULE-LATTICE-BASED KEY-ENCAPSULATION MECHANISM STANDARD)

Unusable

Poor

Good

Perfect

1-3

4-5

6-8

9-10

8.28

Algorithm Overall Usability Score

PREVIOUS NAME

CRYSTALS-KYBER

SPECIFICATION

[FIPS 203](#)

TYPE

Encryption/KEM

FAMILY

Lattice

SUITABLE FOR GENERAL USE

Yes

STANDARDIZATION STATUS

Standardized

RECOMMENDED BY

NIST, BSI, ANSSI

NAMING

TBD

IMPLEMENTATION

🔑

✎

?

COMPLEXITY

?

?

?

SIZE

?

?

?

PARAMETER SET

OID

SECURITY CATEGORY

PERFORMANCE

CIPHERTEXT SIZE

PUBLIC KEY SIZE

ML-KEM-512

2.16.840.1.101.3.4.4.7

I

2

2

2

0

0

ML-KEM-768

2.16.840.1.101.3.4.4.2

III

2

2

3

0

0

ML-KEM-1024

2.16.840.1.101.3.4.4.3

V

3

3

3

0

0

👍 / Pros / Use If:

👎 / Cons / Don't Use If:

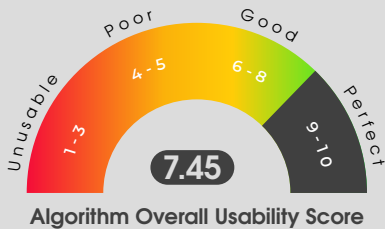
Currently the only post-quantum encryption /key encapsulation algorithm standardized by NIST

Need a general-purpose encryption / key-encapsulation algorithm with decent specs in all categories

You don't want a lattice-based algorithm

Post-Quantum Cryptography Algorithm Cheat Sheets

BIKE (BIT FLIPPING KEY ENCAPSULATION)



PREVIOUS NAME BIKE
SPECIFICATION [Project Page](#)
TYPE Encryption/KEM
FAMILY Codes
SUITABLE FOR GENERAL USE Yes
STANDARDIZATION STATUS 4th round candidate
RECOMMENDED BY NAMING TBD
after security categories I, III and V

IMPLEMENTATION			
COMPLEXITY			
SIZE			

PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE			CIPHERTEXT SIZE	PUBLIC KEY SIZE
BIKE-L1	TBD						
BIKE-L3	TBD						
BIKE-L5	TBD						

NOTE:
No data available for BIKE-L5 for cycle counts. The algorithm overall usability score is computed over BIKE-L1 and BIKE-L3 only.

/ Pros / Use If:

- Possible alternative to ML-KEM not based on lattices.

/ Cons / Don't Use If:

- Not yet standardized. Note that NIST intends to standardize at most one of the algorithms HQC and BIKE.
- Metrics not as good as the ones of ML-KEM.

Post-Quantum Cryptography Algorithm Cheat Sheets

HQC (HAMMING-QUASI-CYCLIC)

Unusable

Poor

Good

Perfect

1-3

4-5

6-8

9-10

6.95

Algorithm Overall Usability Score

PREVIOUS NAME

SPECIFICATION

TYPE

FAMILY

SUITABLE FOR GENERAL USE

STANDARDIZATION STATUS

RECOMMENDED BY

NAMING

HQC

[Project Page](#)

Encryption/KEM

Codes

Yes

4th round candidate

TBD

128, 192 and 256 bits of security in reference to security categories I, III and V

IMPLEMENTATION

?

?

?

COMPLEXITY

?

?

?

SIZE

?

?

?

PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE			CIPHERTEXT SIZE	PUBLIC KEY SIZE
HQC-128	TBD						
HQC-192	TBD						
HQC-256	TBD						

👍 / Pros / Use If:

- Possible alternative to ML-KEM not based on lattices.

👎 / Cons / Don't Use If:

- Not yet standardized. Note that NIST intends to standardize at most one of the algorithms HQC and BIKE.
- Metrics not as good as the ones of ML-KEM.

Post-Quantum Cryptography Algorithm Cheat Sheets

FRODOKEM

Unusable

Poor

Good

Perfect

1-3

4-5

6-8

9-10

5.42

Algorithm Overall Usability Score

PREVIOUS NAME

SPECIFICATION

TYPE

FAMILY

SUITABLE FOR GENERAL USE

STANDARDIZATION STATUS

RECOMMENDED BY

NAMING

FrodoKEM

[Project Page](#)

Encryption/KEM

Lattice

Yes

Disregarded by NIST

BSI, ANSSI

FrodoKEM-[n]-[AES | SHAKE]

where n is a matrix dimension

IMPLEMENTATION

COMPLEXITY

SIZE

?

?

?

?

?

?

?

?

?

👍 / Pros / Use If:

👎 / Cons / Don't Use If:

• More conservative than ML-KEM since it is based on unstructured grids

• Not standardized by NIST because ML-KEM is more efficient.

PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE			CIPHERTEXT SIZE	PUBLIC KEY SIZE
			🔑	🔒	🔐		
FRODOKEM-640-AES	TBD	I	4	4	4	3	3
FRODOKEM-640-SHAKE	TBD	I	5	5	5	3	3
FRODOKEM-976-AES	TBD	III	4	4	4	3	3
FRODOKEM-976-SHAKE	TBD	III	6	6	6	3	3
FRODOKEM-1344-AES	TBD	V	5	5	5	4	4
FRODOKEM-1344-SHAKE	TBD	V	6	6	6	4	4

NOTE:

There are also ephemeral versions of FrodoKEM (eFrodoKEM-640, etc.), but BSI only recommends FrodoKEM-976 and FrodoKEM-1344 versions for long-term protection. The algorithm score also only includes the non-ephemeral versions listed above.

Post-Quantum Cryptography Algorithm Cheat Sheets

CLASSIC McEliece



PREVIOUS NAME Classic McEliece
SPECIFICATION [Project Page](#)
TYPE Encryption/KEM
FAMILY Codes
SUITABLE FOR GENERAL USE No / With Limitations
STANDARDIZATION STATUS 4th round candidate (NIST)
RECOMMENDED BY BSI
NAMING mceliece[n][deg(F(y))][[f]]
where n, F(y) are parameters
non-f versions: $(\mu, \nu) = (0, 0)$
f versions: $(\mu, \nu) = (32, 64)$

IMPLEMENTATION			
COMPLEXITY			
SIZE			

👍 / Pros / Use If:

- Possible alternative to ML-KEM not based on lattices.
- Considered safe and recommended by BSI even though not standardized by NIST.
- Use cases that allow for pre-sharing of keys where the large public key size is not a concern.

👎 / Cons / Don't Use If:

- Not yet standardized by NIST.
- Not suitable for general use due to the very large public keys and poor performance of key generation.
- Due to aforementioned point, likely not suitable for embedded devices.

PARAMETER SET	OID	SECURITY CATEGORY	PERFORMANCE			CIPHERTEXT SIZE	PUBLIC KEY SIZE
MCELIECE348864	TBD						
MCELIECE348864F	TBD						
MCELIECE460896	TBD						
MCELIECE460896F	TBD						
MCELIECE6688128	TBD						
MCELIECE6688128F	TBD						
MCELIECE6960119	TBD						
MCELIECE6960119F	TBD						
MCELIECE8192128	TBD						
MCELIECE8192128F	TBD						

NOTE:
No data available for the f versions for encryption/decryption cycle counts. The algorithm overall usability score is computed over non-f versions only. Non-f and f versions are interoperable, with f versions offering faster key generation and non-f versions having simpler key generation.

Post-Quantum Cryptography Algorithm Cheat Sheets

ALGORITHM OVERALL USABILITY SCORE

We try to measure an algorithm's overall usability for a general use case without any special characteristics by calculating a single number between 0 (worst) and 10 (best) for the algorithm. This calculation is taking into account all parameter sets, performance metrics, public key size, signature/ciphertext size, the number of security categories provided, whether or not it is suitable for general use, and the complexity and size of its implementation. We define an algorithm's overall score as

$$\text{score}_{\text{algorithm}} = \max \left\{ 0, \text{avg} \left\{ \text{score}_{\text{parameterSet}} \mid \text{parameterSet is a parameter set of algorithm} \right\} - \frac{1}{8} \cdot (5 - \text{categories}_{\text{algorithm}}) - \frac{1}{4} \cdot \text{impl}_{\text{algorithm}} - \text{generality}_{\text{algorithm}} \right\}$$

where $\text{score}_{\text{parameterSet}}$ is a score for an individual algorithm parameter set. Depending on the type of algorithm, $\text{score}_{\text{parameterSet}}$ is defined as

$$\text{score}_{\text{signature-parameterSet}} = 10 - \text{avg} \left\{ \text{no}, \text{no}, \text{no}, \text{no}, \text{no} \right\}$$

respectively

$$\text{score}_{\text{encryption-parameterSet}} = 10 - \text{avg} \left\{ \text{no}, \text{no}, \text{no}, \text{no}, \text{no} \right\}.$$

Furthermore, $1 \leq \text{categories}_{\text{algorithm}} \leq 5$ denotes the number of different NIST security categories offered by *algorithm*. By assigning numeric values of $0 = \text{L}_{\text{cp}}^{\text{C}} = \text{L}_{\text{cp}}^{\text{S}}$, $1 = \text{M}_{\text{cp}}^{\text{C}} = \text{M}_{\text{cp}}^{\text{S}}$ and $2 = \text{H}_{\text{cp}}^{\text{C}} = \text{H}_{\text{cp}}^{\text{S}}$, we set

$$\text{impl}_{\text{algorithm}} = \text{avg} \left\{ \text{L}_{\text{cp}}^{\text{C}}, \text{M}_{\text{cp}}^{\text{C}}, \text{H}_{\text{cp}}^{\text{C}}, \text{L}_{\text{cp}}^{\text{S}}, \text{M}_{\text{cp}}^{\text{S}}, \text{H}_{\text{cp}}^{\text{S}} \right\}.$$

Finally, we define

$$\text{generality}_{\text{algorithm}} = \begin{cases} 0 & \text{if } \text{algorithm} \text{ is a general purpose algorithm} \\ 2 & \text{else} \end{cases}$$

to take into account if the algorithm is suitable for general use.

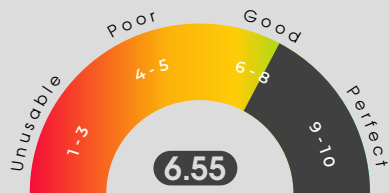
TBD: In the algorithm scores given in ID cards, we currently use $\text{impl}_{\text{algorithm}} = 0$ in the respective computations because the necessary values for implementation complexity and size are still TBD. This will be corrected later.

EXAMPLE: ML-DSA OVERALL USABILITY SCORE

We calculate

$$\text{score}_{\text{ML-DSA-44}} = 10 - \text{avg} \left\{ 3, 3, 2, 1, 5 \right\} = 10 - 2.8 = 7.2$$

Similarly, we obtain $\text{score}_{\text{ML-DSA-65}} = 7.0$ and $\text{score}_{\text{ML-DSA-87}} = 6.2$. Furthermore, $\text{categories}_{\text{ML-DSA}} = 3$ since ML-DSA offers the three security categories III_{cp} , III_{sp} , and V_{cp} , and $\text{generality}_{\text{ML-DSA}} = 0$ since ML-DSA is a general purpose signature algorithm. This results in an overall usability score of 6.55:



ML-DSA Overall Usability Score

$$\begin{aligned} \text{score}_{\text{ML-DSA}} &= \max \left\{ 0, \text{avg} \left\{ \text{score}_{\text{ML-DSA-44}}, \text{score}_{\text{ML-DSA-65}}, \text{score}_{\text{ML-DSA-87}} \right\} - \frac{1}{8} \cdot (5 - \text{categories}_{\text{ML-DSA}}) - \frac{1}{4} \cdot \text{impl}_{\text{ML-DSA}} - \text{generality}_{\text{ML-DSA}} \right\} \\ &= \max \left\{ 0, \text{avg} \{ 7.2, 7.0, 6.2 \} - \frac{1}{8} \cdot (5-3) - 0 - 0 \right\} \\ &= \max \{ 0, 6.8 - 0.25 \} \\ &= \max \{ 0, 6.55 \} \\ &= 6.55 \end{aligned}$$

TBD: We use $\text{impl}_{\text{ML-DSA}} = 0$ because the necessary values to compute it are still TBD, cf. ML-DSA ID card. This will be corrected later.