Post-Quantum Cryptography Standardization

April 21, 2024

Abstract

The article discusses the motivation and goal of the post-quantum standardization process, with a focus on the Module-Lattice-based Key-Encapsulation Standard (FIPS PUB 203). The underlying Module Learning With Errors and Shortest Vector Problem are outlined. Security and performance aspects as well as state-of-the-art attacks and differences from the original CRYSTALS-Kyber scheme are discussed.

1 Standardization process

Over the last decade, there has been a substantial amount of research on quantum computing. Quantum computers should not be seen as better compared to the classical ones, rather they represent a completely a new computing model, where certain problems can be solved far more efficiently.

With the potential advent of large enough quantum computers, the bit security of symmetric cryptography will be halved,[1] but the problems that underpin public cryptography today—integer factorization and discrete logarithms over both finite fields and elliptic curves—will be solved using Shor's algorithm in polynomial time.[8]

Even though the current quantum computers are still toy prototypes and functionally useful quantum computer is somewhere a few years away to impossible, in 2016 the National Institute of Standards and Technology (NIST) started with Post-Quantum Cryptography (PQC) standardization process. The goal is to standardize publicly disclosed digital signature, public-key encryption and key-establishment algorithms that are available worldwide and are capable of protecting sensitive government information well into the foreseeable future, including after the advent of quantum computers.[4]

Apart from the security, performance and adaptation considerations, the ambition is to standardize at least two algorithms for each category—digital signature, public-key encryption/key-establishment algorithms—and these algorithms should be based on different underlying hard problems in order to reduce the risk of leaving the world without a viable standard, in case one of the problems would be broken.[6]

The long time anticipated for standardization, implementation, and adaptation of PQC is the reason we started with the process well in advance of our expectations for the construction of large quantum computer. The lengthy process was expected because of our (1) limited ability to predict the performance characteristics of future quantum computers, such as their cost, speed, and memory size, and (2) hardness to compare post-quantum cryptosystems as they are based on completely different design attributes and mathematical foundation.[5]

Furthermore, the infrastructure needs to be developed well before the quantum computers are there, as we want the secrets compromised by quantum cryptanalysis to be no longer sensitive.[5] This concern is expressed in the attack with a self-explanatory name: The Harvest Now, Decrypt Later. When considering, that it took almost two decades to deploy modern public key cryptography[4] and the transition to post-quantum cryptography will probably not be a drop-in replacement, the companies should already start enforcing cryptographic agility and create migration plans for moving to the post-quantum cryptography.[5][1].

[1] Grover's search algorithm gives a square root time boost for the problem of key search for AES and 3DES[8].

2 Module-Lattice-Based Key-Encapsulation Mechanism

The report starts with a summary of the structure of the standard, followed by an explanation of the underlying problems and a description of the construction. The security and performance characteristics are discussed. Finally, the differences from CRYSTALS-Kyber are stated and our conclusion is given.

2.1 Structure of the standardization document

The preamble sets the document into a context by clearly explaining the motivation and defining the purpose, scope, and subject of the standard. To help standard adoption the patent information and point of contact are provided. The standard briefly revisits the context and describes differences from the 3rd round submission of CRYSTALS-Kyber. Chapter 2 supports clarity by the definition of all used terms, acronyms, and definitions, as well as an explanation of mathematical and pseudocode notation. In Chapter 3, the overall picture and requirements for the ML-KEM are provided and key encapsulation mechanism (KEM) is illustrated. Chapter 4 states the auxiliary algorithms. The essence of the standard is Chapters 5 to 7. Chapter 5 describes the underlying public encryption (K-PKE) consisting of thee routines *Key generation*, *Encryption* and *Decryption*. Finally, in Chapter 6 the ML-KEM is presented in the form of three standardized algorithms *Key Generation*, *Encapsulation*, and *Decapsulation*. The three sets of parameters standardized by the standard – ML-KEM 512, ML-KEM 768, ML-KEM 1024 – along with associated security categories are specified in Chapter 7. The security categories are provided as an appendix.

2.2 Key Encapsulation Mechanism

The key encapsulation mechanism (KEM) is a set of algorithms to establish a shared secret between two communication parties. This shared secret key, can then be used for a symmetric-key cryptography. We illustrate the KEM with two parties Alice and Bob: Alice uses *Key generation* algorithm to generate an encapsulation (public) key and decapsulation (private) key. Then, Alice sends the encapsulation key to Bob. Bob runs *Encapsulation* algorithm, which produces a shared secret S_B (to be used by Bob) and associated ciphertext, which is sent back to Alice. Alice obtains her version of shared secret S_A by running *Decapsulation* with the decapsulation key and obtained ciphertext.

2.3 Underlying Security Problem

The schema is essentially a Lyubashevsky, Peikert, Regev (LPR) encryption within the Module Learning With Errors (MLWE) setting.[14]

The Learning With Errors (LWE) problem can be stated in two variants (1) search and (2) decision version. The setting for both is the same. Then, given a modulus q, uniformly distributed matrix $\mathbf{A} \in \mathbb{Z}_q^{k \times l}$ and χ. Sample $\mathbf{s} \in \mathbb{Z}_q^l, \mathbf{e} \in \mathbb{Z}_q^k$ from χ and compute $\mathbf{As} + \mathbf{e} = \mathbf{t}$. When given the $\mathbf{A}$ and $\mathbf{t}$, the goal is to find the secret value $\mathbf{s}$ in the search version and distinguish $\mathbf{t}$ from uniformly random in the decision version.

The plain LWE works with vectors of independent integers. As a consequence, $O(n^2)$ of memory is needed to store the matrix $\mathbf{A}$ and the computation costs are $O(n^2)$.[9] High storage and computation requirements led to the development of Ring Learning With Error (RLWE), where we only need to store the first column of the matrix, as for the second column we shift all elements a position down and negate the last element of previous column (now on top). The added structure results in the storage requirement decreasing to $O(n)$

and as the columns are no longer independent the Number-Theoretic-Transform can be used for the multiplication with the final computational complexity of $O(n \log n)$.[13][9]

The Module Learning With Errors (MLWE) further builds on the RLWE by introducing the module rank parameter k. In MLWE the matrix $\mathbf{A}$ can be thought of as built from $k \times k$ matrices with the structure corresponding to RLWE. Therefore the natural requirements are $O(k^2 n)$ for storage and $O(k^2 n \log n)$. The reason to move from RLWE to MLWE is motivated by more flexibility as the higher security level is achieved by increasing the module rank k without changes in the underlying arithmetic. This enables more efficient implementations.[9]

The LWE problems can be translated to problems on lattices (therefore the name of the standard *Module-Lattice-based*). Lattice $\mathcal{L}$ is defined as a discrete additive subgroup of $\mathbb{R}^n$ and the natural way of thinking about lattices is as periodic patterns of points in a grid. The ML-KEM is specifically connected to the approximate version of the Shortest Vector Problem (SVP). In the non-approximate version, the task is: Given lattice $\mathcal{L}$, find a non-zero $v \in \mathcal{L}$, such that v is the shortest vector in the lattice. In the α-approximate version, our goal is to find $v \in \mathcal{L}$, which is maximally α-time larger than shortest vector.[11]

2.4 Construction

The construction is based on the IND-CPA[2]public key encryption K-PKE, which is then turned into the IND-CCA2[3] key encapsulation mechanism ML-KEM using the Fujisaki Okamoto transform (FO).[16]

The to-be-established shared secret is fixed to 256 bits. The underlying ring is is $R = \mathbb{Z}[X]/(X^{256}+1)$ and the module ranks are $k = 2, 3, 4$ corresponding to NIST security categories $1, 3, 5$. The integer modulus is fixed to $q = 3329$. The matrix $\mathbf{A} \in R_q^{k \times k}$ is pseudorandomly generated from random 256-bit string.[4] Two secret vectors of polynomials $\mathbf{s}, \mathbf{e} \in R_q^k$ are sampled independently from the central binomial distributions χ with parameters η_1 and η_2 to set appropriately to produce numbers relatively small to q with high probability. The $\mathbf{s}$ represents the private key and the vector $\mathbf{e}$ is the error term. The corresponding public key is $pk = (\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$.[6][16]

Next, the three top-level algorithms of ML-KEM will be explained together with the K-PKE subroutines. We start with key generation. Both *ML-KEM.KeyGen()* and *K-PKE.KeyGen()* takes no input and requires randomness. Firstly, the encryption (public) and decryption (private) keys are generated (as explained in the previous paragraph) by *K-PKE.KeyGen()*, which is invoked as a core subroutine in *ML-KEM.KeyGen()*. *ML-KEM.KeyGen()* returns two keys (1) encapsulation key, which is exactly the encryption key returned by *K-PKE.KeyGen()* and (2) decapsulation key, which is comprised from the decryption key, encapsulation key, hash of the encapsulation key, and a random value used for implicit rejection, in case decapsulation would not hold the correct result.

The encapsulation routine by *ML-KEM.Encaps(ek)*, where the ek is the encapsulation key, must firstly check (1) the length of the encryption key ek and (2) whether coefficients of ek are from $\mathbb{Z}_q$. If both checks pass, the *K-PKE.Encrypt(ek_{PKE}, m, r)* is called with encryption key ek_{PKE} and explicit randomness r to encrypt random value $m \in R_q$. The encryption consists of sampling two vectors of polynomials $\mathbf{r}, \mathbf{e_1} \in R_q^k$ and polynomial $e_2 \in R_q$ with all coefficients of each polynomial chosen independently from χ. Then the

[2]Indistinguishability under chosen plaintext attack.
[3]Indistinguishability under chosen ciphertext attack.
[4]Therefore only the string needs to be transmitted as opposed to the whole matrix

ciphertext c is computed as (remember, that $\mathbf{t}$ is the part of public key):

$$c = (\mathbf{u}, v) = (\mathbf{rA} + \mathbf{e_1}, \mathbf{rt} + e_2 + \lfloor q/2 \rceil \cdot m) \in R_q^k \times R_q$$

The shared secret K is derived from m and the encapsulation key ek_{PKE} via hashing. The *ML-KEM.Encaps* returns the shared secret together with the resulting ciphertext c.

The last missing piece is the decapsulation routine *ML-KEM.Decaps(c, dk)*, taking ciphertext c and decapsulation key dk as input. The routine must start with two checks (1) the length of the ciphertext c and (2) the length of the decapsulation key dk. If both checks pass, the core consists of parsing the components of the decapsulation key, specifically the decryption key, encryption key, hash of encryption key, and implicit rejection value. The ciphertext is decrypted into m' using the *K-PKE.Decrypt(dk_{PKE}, c)* with dk_{PKE} being the decryption key and c the ciphertext. *K-PKE.Decrypt* starts by decompressing the ciphertext, then computes the intermediate value

$$w = v - \mathbf{us} = [\mathbf{rAs} + \mathbf{re} + e_2 + m] - [\mathbf{rAs} + \mathbf{e_1 s}] = m + \mathbf{re} + \mathbf{e_1 s} + e_2$$

The values of $\mathbf{r}, \mathbf{s}, \mathbf{e}, \mathbf{e_1}, e_2$ are small relative to the transmitted message m (where all coefficients are scaled up). Therefore the message m can be recovered from w by rounding each coefficient of the polynomial w module 2. Finally the m is re-encrypted into ciphertext c' and compared with received ciphertext c. If the values of c and c' differ the algorithm returns a value corresponding to the hash of ciphertext together with implicit rejection value. Otherwise, the shared secret derived by hashing from message m is returned.

2.5 Security

The security of ML-KEM, more precisely the K-PKE is based on lattice cryptography, which has been studied for decades. It is essentially connected to the approximate versions of SVP with larger factors, which we expected not to be NP-hard (as is the case for smaller factors), but still believed to be computationally hard.[11]

The security proofs for the employed FO transform hold tightly in the Random Oracle Model (ROM) and non-tightly in the Quantum Random Oracle Model (QROM). But under other natural assumptions, ML-KEM may also achieve tight security reduction in the Quantum Random Oracle Model (QROM).[6]

Currently, the best known way to break the ML-KEM is by breaking the underlying MLWE problem. Therefore finding the solution for the noisy linear system. The attacks are classified into three categories primal, dual, and hybrid (a combination of the previous two).[3]

The primal attack runs the lattice reduction algorithm. The algorithm employed for this is BKZ, which calls a subroutine to solve the instances of the Shortest Vector Problem. For solving the shortest vector problem, two approaches – enumeration and sieving – are possible. Currently, sieves achieve better results, with the state-of-the-art sieve being $bgj1-sieve$.[7] By analyzing the heuristically selected attack parameters, we can conclude, that for ML-KEM-512 the resulting complexity is the same as solving the SVP problem in lattice of dimension 400.[3]

The dual attacks leverage the structure of the dual lattice[5] to solve the underlying lattice problem. These attacks are far more complex to understand, but it turns out, that there are tricks, which make these attacks currently a little bit faster than the primary attacks.[3][15]

[5]For a given lattice, its dual lattice consists of all vectors, that are orthogonal to all vectors in the original lattice. Essentially, it's a lattice of all the vectors perpendicular to the original lattice.

Recently an article stating a substantial improvement in the problem of solving the SVP on quantum computers was published.[10] The article explicitly states, that the current version is not able to break the ML-KEM implementation. Also, the article is very recent and was not opposed by security experts.[6]

2.6 Performance

The ML-KEM provides good performance on both x86-64 with AVX2 extension and ARM Cortex-M. The ML-KEM is even comparably fast (if not faster) when compared with today's elliptic curve cryptography.[2] The ML-KEM is also suitable for use on constrained devices as can be observed from the PQM4 results.[12] When implemented without protections to side channels, the implementation uses less than $4KiB$ of RAM and less than $20KiB$ of storage for code.[6]

Both public key and ciphertext sizes are on the order of thousand bytes, therefore a few magnitudes larger, than sizes used by today's elliptic curve cryptography.[2] However the sizes should still be acceptable for most applications.[6] Standardization of both the ML-KEM based on CRYSTALS-Kyber and ML-DSA based on CRYSTALS-Dilithium, which share the same computational platform, provides an important advantage for the software and hardware implementations. Also, the ML-KEM performs only the integer arithmetic, with floating point arithmetic explicitly disallowed. This makes the implementation less prone to errors.[16]

The NIST also concludes, that all structured lattice finalists, can be substituted in existing protocols with relatively small or no cost.[6] This can be also observed as the $PQXDH$ version of Signal protocol, as well as $PQ3$ introduced by Apple (based on Signal protocol) are based on the same CRYSTALS-Kyber scheme. The $PQ3$ also shows the feasibility of forward secrecy using periodic re-keying, which is done about every 50 messages due to the size of the keys.

2.7 Differences from CRYSTALS-Kyber

The main differences from the 2018 proposal for CRYSTAL-Kyber compared to the FIPS 203 proposal can be summarized in four parts. First, the integer modulus q has been changed from 7681 to 3329. Secondly the *ML-KEM.Encaps* and *ML-KEM.Decaps* algorithms in the current specification use a different variant of the Fujisaki-Okamoto transform than the third-round specification. Specifically, *ML-KEM.Encaps* no longer includes a hash of the ciphertext in the derivation of the shared secret, and *ML-KEM.Decaps* has been adjusted to match this change. As this standard requires the use of NIST-approved randomness generation, this step is unnecessary and is not performed in *ML-KEM*. The current specification includes explicit input validation steps that were not part of the third-round specification.

3 Conclusion

The ML-KEM is a very fast lattice-based scheme for establishing secret over a public channel. The security is based on the hardness of the MLWE problem which is in turn based on the hardness of the SVP problem. Its keys are bigger than those of pre-quantum schemes, but small enough to be used in real-world systems.

[6] We personally cannot asses, the validity of the article, as we are missing a substantial amount of knowledge about both lattice cryptography and quantum computers

References

[1] Achieving Crypto Agility - with Rebecca Krauthamer and Greg Bullard from QuSecure, August 2021.

[2] Kyber and post-quantum crypto, December 2021.

[3] Attacking lattice-based cryptography with martin albrecht, November 2023.

[4] NIST Computer Security Resource Center. Post-Quantum Cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography`, April 2024.

[5] NIST Computer Security Resource Center. Post-Quantum Cryptography. Call for Proposals. `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals`, April 2024.

[6] Gorjan Alagic, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

[7] Martin R Albrecht, Vlad Gheorghiu, Eamonn W Postlethwaite, and John M Schanck. Estimating quantum speedups for lattice sieves. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 583–613. Springer, 2020.

[8] Ritik Bavdekar, Eashan Jayant Chopde, Ankit Agrawal, Ashutosh Bhatia, and Kamlesh Tiwari. Post quantum cryptography: A review of techniques, challenges and standardizations. In *2023 International Conference on Information Networking (ICOIN)*, pages 146–151, 2023.

[9] Mojtaba Bisheh-Niasar. Introduction to lattice-based cryptography. the case study of kyber, September 2022.

[10] Yilei Chen. Quantum algorithms for lattice problems. *Cryptology ePrint Archive*, 2024.

[11] Joel GÃ¤rdner. Lattice cryptography, 2024.

[12] Matthias J Kannwischer, Markus Krausz, Richard Petri, and Shang-Yi Yang. pqm4: Benchmarking nist additional post-quantum signature schemes on microcontrollers. *Cryptology ePrint Archive*, 2024.

[13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. Cryptology ePrint Archive, Paper 2012/230, 2012. `https://eprint.iacr.org/2012/230`.

[14] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013.

[15] MATZOV. Report on the security of lwe: Improved dual lattice attack. 2022.

[16] National Institute of Standards and Technology (2023) (Department of Commerce, Washington, D.C.). Module-lattice-based key-encapsulation mechanism standard. 2023.