Comparative study of Post-Quantum Key-Exchange Mechanisms and its implementations

Summary Report

Aditya Bhardwaj

1. Abstract

The design of cryptographic algorithms that are used in the infrastructure today, is based on assumptions of computational difficulty. These are asymmetrical problems, such as integer factorization: where multiplying two integers is straightforward but factoring a 1000-digit integer is difficult. However, quantum computers can efficiently solve some of these problems, such as integer factorization and discrete logarithms, using algorithms like Shor's algorithm and Grover's algorithm. Post-quantum cryptography, also known as quantum-resistant or quantum-safe cryptography, aims to address these security challenges by designing cryptographic algorithms that remain secure against attacks from both classical and quantum computers. For my work, I chose some of the Post-Quantum Key-Exchange Mechanisms algorithms submitted for the NIST-PQC competition and have done a comparison in terms on speed and performance.

2. The Quantum Threat and NIST PQC competition

Quantum computers can leverage their unique quantum properties, such as superposition and entanglement, to perform certain calculations exponentially faster than classical computers. This allows them to solve problems that would take classical computers an impractical amount of time. It includes breaking the mathematical problems of traditional cryptography systems such as RSA, ECC.

Post-quantum cryptography (PQC) is an active area of research that aims to develop cryptographic algorithms that would remain secure even in the presence of large-scale quantum computers. Researchers are exploring various approaches, including lattice-based cryptography, hash-based cryptography, code-based cryptography,

and multivariate polynomial cryptography, among others. National Institute of Standards and Technology (NIST) launched an initiative and announced a competition in 2016 where they will evaluate the submissions and engage with the cryptographic community to make decisions about the standardization of PQC algorithms.

3. PQC Standardisation

23 signature schemes and 59 encryption/KEM schemes were submitted by the initial submission deadline at the end of 2017 of which 69 total were deemed complete and proper and participated in the first round. After three rounds of evaluation and analysis, NIST selected four algorithms it will standardize as a result of the PQC Standardization Process. The public-key encapsulation mechanism selected was CRYSTALS–KYBER, along with three digital signature schemes: CRYSTALS–Dilithium, FALCON, and SPHINCS+. Since major of the algorithms are lattice-based they also announced alternative schemes. After round 4, from these alternative schemes, the key-encapsulation mechanism (KEM) algorithms are BIKE, Classic McEliece, Falcon, and HQC.

Current status (from official NIST announcement)

NIST plans to hold the 5th NIST PQC Standardization Conference from April 10-12, 2024, in Rockville, Maryland. **The purpose of the conference is to discuss various aspects of the algorithms (both those selected and those being evaluated) and to obtain valuable feedback for informing decisions on standardization**. NIST will invite the submission teams for BIKE, Classic McEliece, Falcon, and HQC to give an update on their algorithms.

4. Methodology

I have selected 4 KEM algorithms for my study. These are: **CRYSTALS–KYBER, NTRUPrime, BIKE, Classic McEliece**. The first two are lattice-based algorithms while the other two are code-based cryptography algorithms. First I have compared them each other in their own group. Then in the end, I compared them all. For each algorithm, I have tried to understand the math behind and read some relevant papers in order to gain insights about the performance. All of these have more than one implementations (some are optimized for certain hardware).

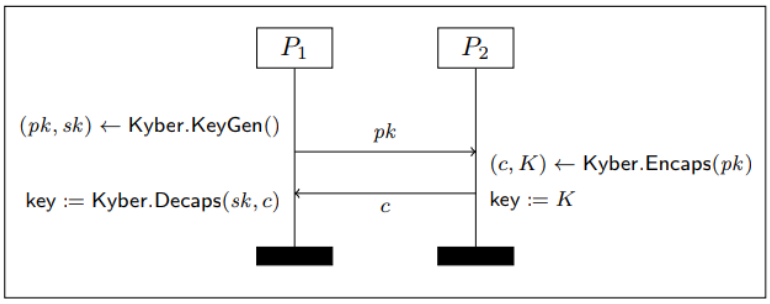


Figure 1. Kyber.KE – Key Exchange protocol using the Kyber = (KeyGen, Encaps, Decaps) key encapsulation mechanism.

Figure 1: KEM functions

For comparing the speed and performance of the KEMs, we can focus on the functions *KeyGen()*, *Encaps()*, *Decaps()*. Figure 1 shows that public key *pk* and secret key *sk* are generated from KeyGen(), then a sharing secret *K* is generated which is encrypted using the function Encaps(). With the correct secret key *sk*, the ciphertext *c* can be decrypted to get the shared secret by the other party.

I will compare the sizes of the keys generated along with ciphertext and shared secret. For performance, we can compare the running time needed to successfully execute these three functions in a KEM. I will use the Open Quantum Safe project to run these algorithms and generate a shared secret as done in a KEM.

5. Lattice-based cryptography

Lattices are algebraic structures that consist of set of points in an n-dimension space. They can be defined by a set of n vectors B = (b_1, b_2, …, b_n), with B as the basis of the lattice. Below is an example from [1]

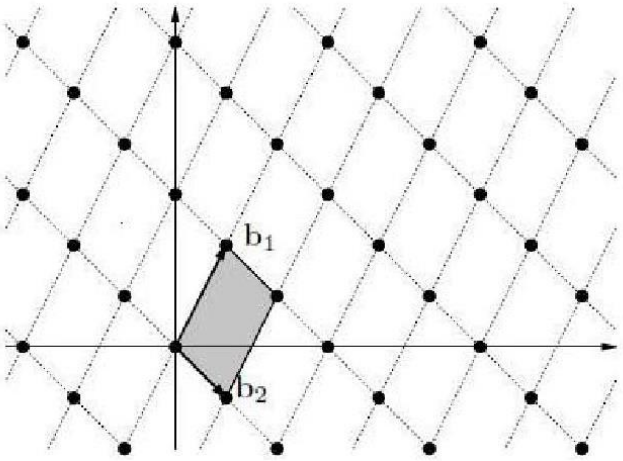


Figure 2: A 2-D lattice generated by basis B = [b_1,b_2]

Some problems on lattices, are hard to solve even for a quantum computer. **Shortest Vector Problem (SVP)**: given a set of basis B of an n-dimensional lattice L, find the shortest non-zero vector in the lattice. There is also **Closest Vector Problem (CVP)**: given a set of basis B of an n-dimensional lattice L and a target point x, find the lattice point that is closest to the target. Variants of these problems exists but the base idea remains the same.

The hardness problem at the core of lattice-based cryptography is often related to the Learning With Errors (LWE) problem. LWE is a mathematical problem associated with the computational hardness of extracting a secret from linear equations with errors.

The Learning With Errors (LWE) Problem: Given a set of random linear equations in the form $a_1 \cdot s + e_1$, $a_2 \cdot s + e_2$, … $a_n \cdot s + e_n$, where s is a secret vector, a_i are random vectors, and e_i are small random errors, the task is to learn the secret vector s based on the information provided. Their security can be proven by finding a theoretical reduction to a computationally hard lattice problem.

One downside of cryptographic schemes based on LWE is that they require quite large keys, usually in the order of n^2, and therefore, their runtime performance is also affected. One way to reduce the size of the keys is by assuming that the lattice has a specific structure. This can

be achieved by interchanging the group Z_q^n with a ring, such as the polynomial ring $R_q = Z_q[x]/(x^n + 1)$, where $Z_q[x]$ is the set of all polynomials that have coefficients in Z_q. One can then sample instances in (R^k_q , R_q) where k is the rank of the lattice. If k = 1, then we call the problem Ring-LWE, if k >= 2 we call it Module-LWE.

CRYSTALS–KYBER is based on Module-LWE with the polynomial ring $Z_q[x]/(x^n + 1)$. **NTRU Prime** substitutes the commonly used ring $R = Z_q[x]/(x^n +- 1)$ with the field $F = Z_q[x]/(x^n – x - 1)$. But the hardness is still based on the SVP or CVP (or the variants). NIST advanced NTRU Prime to the final round as an alternate finalist because of the different choice for the ring, which could be an alternative to the cyclotomic ring used by most of the other schemes.

	KeyGen	**Encap**	**Decap**
Kyber-1	122684	154524	187960
NTRU-1	12506668	761236	1940870
Kyber-3	199408	235260	274900
NTRU-3	21833048	1313454	3399726
Kyber-5	307148	346648	396584
NTRU-5	31835958	1856936	4920436

Table 1: Kyber and NTRU comparison on performance in Haswell cycles

	Public Key	**Secret Key**	**Ciphertext**
Kyber-1	800	1632	768
NTRU-1	699	935	699
Kyber-3	1184	2400	1088
NTRU-3	930	1234	930
Kyber-5	1568	3168	1568
NTRU-5	1230	1590	1230

Table 2: Kyber and NTRU comparison on size. Shared secret was 32-bytes long for both KYBER and NTRUPrime

The 1,3,5 next to the names of the algorithms refer to the security level offered as specified by NIST.

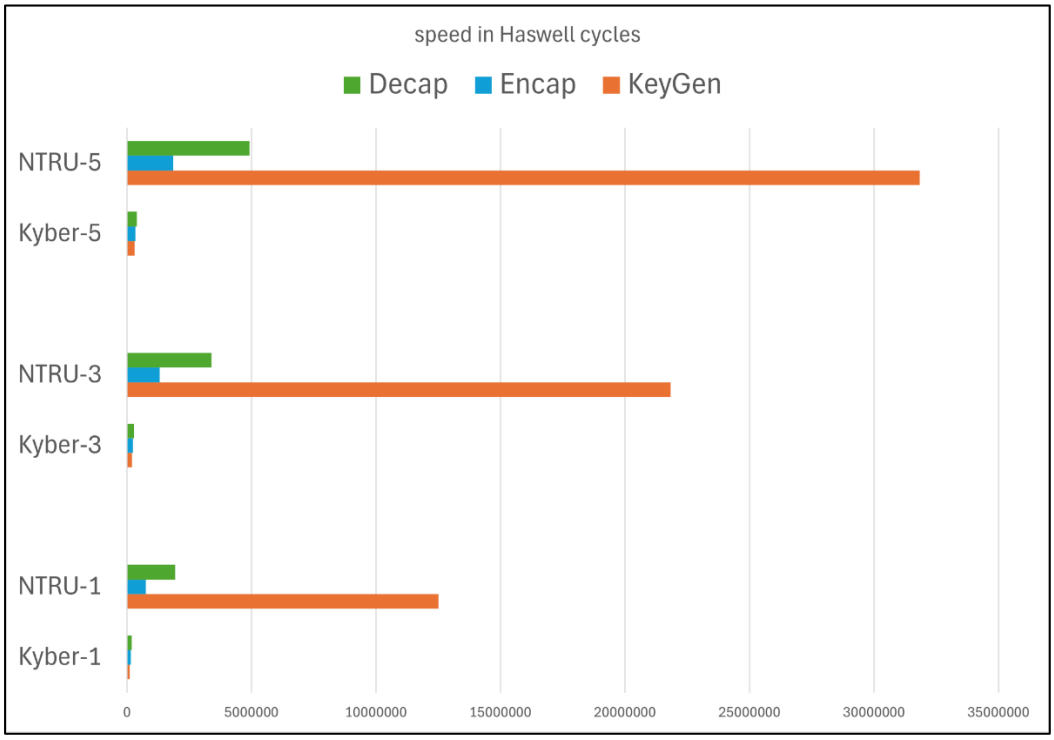


Figure 3: Running time

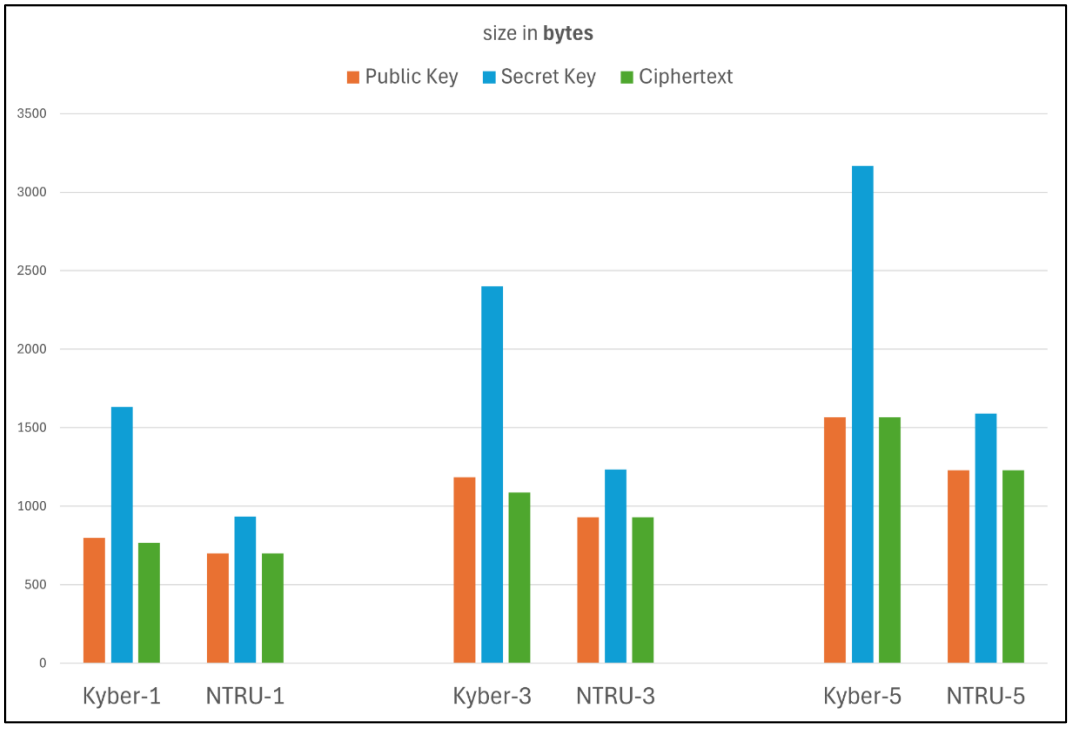


Figure 4: Size of public, secret key and ciphertext

6. Code-based cryptography

The first code-based cryptosystem was developed by and named after Robert McEliece in 1978 [3]. Code-based cryptography relies on the idea that decoding a random linear code is computationally hard, NP-complete to be exact [4].

The figure below explains the McEliece and the basic construction.

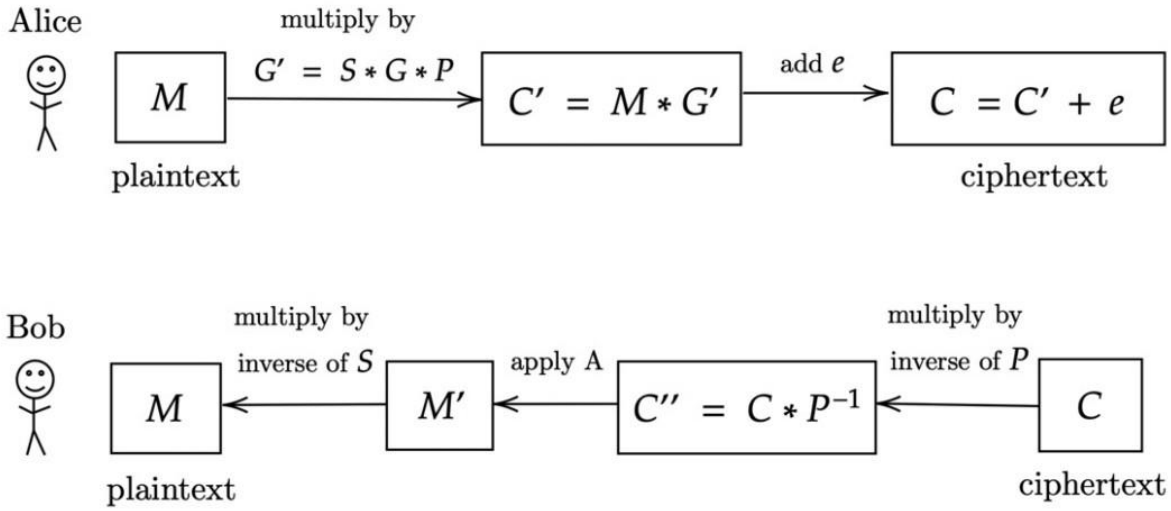


Figure 5: Alice wants to send a message to Bob using the McEliece cryptosystem

Alice wants to securely send a message to Bob. She turns her message into a linear code that has an efficient decoding algorithm A. The linear code Alice decides to use has some generator matrix G which will extend the message M with some parity bits. After multiplying M by G, she will add some random noise in the form of an error vector e. Bob will be able to decrypt the message by using an efficient decoding algorithm. To ensure that Eve will not be able to apply this efficient decoding algorithm, the McEliece Cryptosystem disguises the code by multiplying G by an invertible matrix S and a permutation matrix P. When we multiply all three together, we obtain the public key G' . Eve does not know the factorization of G' , so she will not be able to obtain G. The ciphertext, M ∗ G'+e, will look like a random linear code to Eve. Bob on the other hand has the private key as well, which tells him how to undo the effects of the matrices S and P. Now, he can efficiently decode the code word to obtain the original message.

The ciphertext is made up of the message multiplied by some generator matrix with an error vector added to it.

The security of these schemes is based on the hardness of decoding structured codes, making them resistant to quantum attacks. The goal of the general decoding problem is to find the nearest valid code word to the vector we have been given. Once we have obtained the code word, we decode it to get the original message.

A different way of looking at the decoding problem is to view it as a syndrome decoding problem. We take our McEliece ciphertext, consisting of a code word and an error vector, and multiply this by parity check matrix. $H * (c + e) = 0 + H * e$. This results in the term $H * e$, commonly referred to as the syndrome. The goal of the syndrome decoding problem is to extract the error vector from this $H * e$ term when we know H [5]. The general decoding problem and the syndrome decoding problem are equivalent and have been proven to be NP-complete [4].

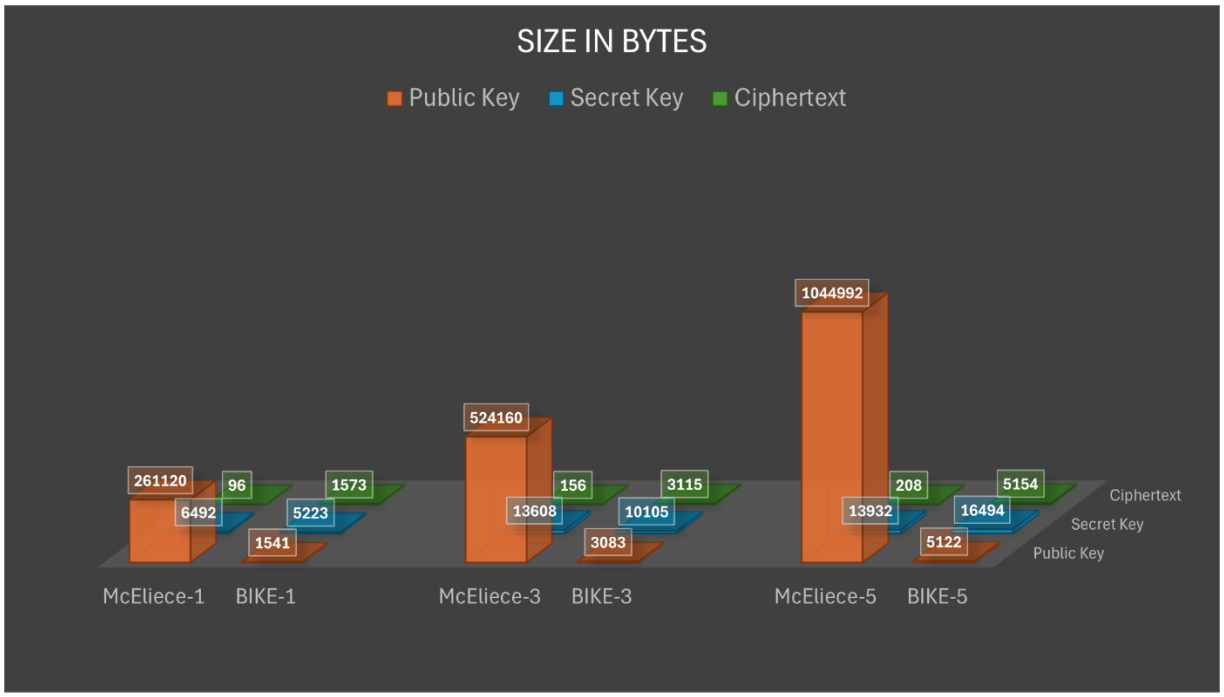
While the original **McEliece** crypto-system is built upon the general decoding problem, the Niederreiter crypto-system makes use of the equivalent syndrome decoding problem. **BIKE** is based on the Niederreiter crypto-system and uses QC-MDPC (Quasi-cyclic moderate-density parity-check) codes. The key size is significantly smaller than the ones generated in McEliece system which uses Binary Gappa codes. BIKE doesn't offer CCA security.

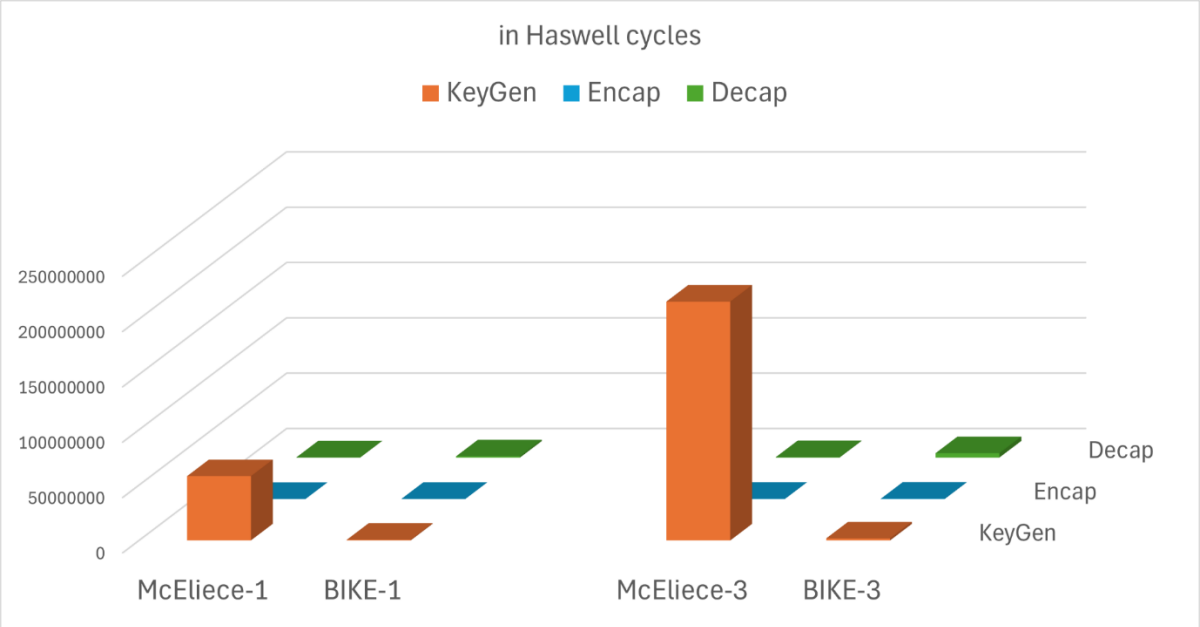
	Public Key	**Secret Key**	**Ciphertext**
McEliece-1	261120	6492	96
BIKE-1	1541	5223	1573
McEliece-3	524160	13608	156
BIKE-3	3083	10105	3115
McEliece-5	1044992	13932	208
BIKE-5	5122	16494	5154

Table 3: McEliece and BIKE comparison on size. Shared secret was 32-bytes long for both McEliece and BIKE

	KeyGen	Encap	Decap
McEliece-1	58034411	44350	134745
BIKE-1	589000	97000	1135000
McEliece-3	215785433	117782	271694
BIKE-3	1823000	223000	3887000

Table 4: McEliece and BIKE comparison on performance in Haswell cycles





7. Conclusion

Lattice-based cryptography algorithms seem to be the most promising and quantum-safe. The maximum number of algorithms announced by NIST in 3rd round belonged to the lattice-based cryptography family). FrodoKEM and NTRU Prime have been announced as alternate candidates for Key Encapsulation Mechanisms. Classic McEliece has been tried and tested for its security hence, it is part of the alternative algorithms chosen by NIST along with BIKE. It is useful in cases where public key doesn't need to be stored (McEliece was used in PQ Wireguard VPN) [6]

8. References

[1] G. Zhang and J. Qin. "Lattice-based threshold cryptography and its applications in distributed cloud computing". (June 2015).

[2] https://openquantumsafe.org/

[3] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory", California Institute of Technology, 1978.

[4] E. R. Berlekamp, R. J. McEliece and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems", 1978.

[5] G. Kachigar and J. Tillich, "Quantum Information Set Decoding Algorithms", April 2017.

[6] A. Hülsing, K. -C. Ning, P. Schwabe, F. J. Weber and P. R. Zimmermann, "Post-quantum WireGuard," *2021 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2021, pp. 304-321, doi: 10.1109/SP40001.2021.00030.