A Contextual Analysis of CRYSTALS-Dilithium, a Quantum-Resistant, Lattice-Based Digital Signature Scheme

Austin Hunt
School of Engineering
Vanderbilt University
Greenville, SC
austin.j.hunt@vanderbilt.edu

***Abstract*—This paper presents an analysis of CRYSTALS-Dilithium, a novel algorithm for generating digital signatures that can withstand future cyberattacks from quantum computers. As announced in July 2022, CRYSTALS-Dilithium was one of the first four winners in the post-quantum cryptography (PQC) standardization project managed by The National Institute of Standards and Technology (NIST). NIST started this competition to call upon cryptographers across the globe for the creation and vetting of quantum-resistant encryption methods and standards in preparation for an inevitable future of quantum-computing-based assaults. As a member of the Cryptographic Suite for Algebraic Lattices (CRYSTALS), Dilithium is a lattice-based algorithm providing not only security based on the hardness of lattice problems, but also competitive public key compression and multiple efficient implementations.**

This paper includes an overview of the Dilithium scheme for quantum-resistant digital signatures and aggregates contextual information around deeper topics supporting its principal design aspects.

***Index Terms*—CRYSTALS, Dilithium, quantum computing, quantum-resistant cryptography, encryption, digital signatures**

I. Introduction

Maintaining trust in the digital world requires a certain guarantee that messages being sent and received are not being tampered with during transmission. Through the use of public-key cryptography, digital signatures provide that guarantee: before transmission, messages are hashed to produce message digests which are then encrypted with the sender's private key to produce the message's digital signature. The digital signature, appended to the message before transmission, allows the receiver to confirm that the message is coming from the expected sender; by hashing the message into a message digest using the same hash function as the sender, and by decrypting the appended digital signature with the sender's public key to obtain the *signed* message digest, the receiver can simply compare the two message digests to ensure they are the same. If they are, the integrity and authenticity of the message is verified.

Since today's digital signatures are built on public-key cryptography (PKC) as described, they are vulnerable to the upcoming wave of quantum computing since the prime factorization on which PKC's security relies is specific to the context of classical computing; factoring large numbers and thus breaking PKC will be a relatively easy and quick task for quantum computers, as shown by Jiang et al. in their work with quantum annealing for prime factorization [1].

Guided by this inevitability, The National Institute of Standards and Technology (NIST) initiated a competitive program in 2016 centered on the development of new public-key cryptographic algorithms resistant to quantum computing attacks [2], which comes down to replacing our currently standardized reliance on prime factorization for security. The candidate algorithms submitted to this competition are either general encryption algorithms for protecting data confidentiality during transmission or digital signature algorithms for protecting data integrity.

In July 2022, NIST announced the selection of four winning candidate algorithms for quantum-resistant public-key cryptography, one of which was the CRYSTALS-Dilithium digital signature algorithm [3]. This algorithm, recommended as the primary choice among two others (FALCON [4] and SPHINCS+ [5]), provides digital signature security based not on prime factorization problems, but the hardness of lattice problems, which have become quite popular in the defensive push for quantum-resistant cryptography as explored by Nathan Manohar in their 2016 Harvard research [6].

II. The CRYSTALS-Dilithium Scheme

This section will outline various important aspects of the CRYSTALS-Dilithium Scheme introduced by Ducas et. al in 2018 [7], and it will additionally provide contextual information for deeper topics on which this scheme rests, such as the hardness of the closest vector problem underlying lattice-based cryptography, the intentional omission of discrete Gaussian sampling by Dilithium's developers, the use of the Learning with Errors problem, and the foundational Fiat-Shamir with aborts technique for lattice-based signature schemes on which Dilithium's design is based.

A. The Hardness of Lattice-Based Cryptography

1) Lattices: To understand lattice-based cryptography - that is, cryptography involving and ultimately depending on lattices - it is important to first understand the concept of a lattice, which relies on some linear algebra. Ultimately, a lattice $\mathcal{L}$ can be thought of as an infinite, evenly spaced out grid of points extending into infinity in all directions, where that grid can be any number of dimensions N. If we consider the points as vectors (i.e., quantities with both a distance and a direction from the origin of the grid) and the grid itself as a vector space, then each vector in that infinite N-dimensional vector space, by definition, can be expressed as a linear combination of N basis vectors, meaning the full lattice can be described using those N basis vectors. Put more simply, every point in an N-dimensional lattice represents a weighted sum of N basis vectors. For example, a 3-dimensional lattice can be described with the basis vectors

$$\hat{\boldsymbol{\imath}} = \begin{bmatrix}1\\0\\0\end{bmatrix}, \hat{\boldsymbol{\jmath}} = \begin{bmatrix}0\\1\\0\end{bmatrix}, \hat{\boldsymbol{k}} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

since every 3-dimensional vector v can be expressed as a weighted sum of those vectors, i.e.

$$v \in \mathbb{R}^3 \implies v = a * \hat{\boldsymbol{\imath}} + b * \hat{\boldsymbol{\jmath}} + c * \hat{\boldsymbol{k}}$$

It is key to remember that a lattice can be described by an infinite number of basis vectors.

2) Shortest Vector Problem: The fundamental quantity of interest with lattice-based cryptography, as MIT Professor Vinod Vaikuntanathan explained quite well in his 2015 talk on the mathematics of lattices [8], is the shortest non-zero vector in a lattice. That is, given an arbitrary basis, the goal with this famous Shortest Vector Problem is to find the shortest non-zero vector (of length λ_1) in the lattice described by that basis. An extension to this problem is simply to find an α-approximation of that shortest vector, or more specifically, a vector that is at most $\alpha * \lambda_1$ in length, λ_1 being the shortest length. Extending further upon this idea is the Shortest Independent Vectors Problem, which involves finding the shortest n linearly independent[1] vectors in a lattice such that you obtain the first shortest length λ_1 up to the nth shortest length λ_n.

3) Closest Vector Problem: Contrary to the previously provided 3-dimensional example, lattices used in lattice-based cryptography can be hundreds or even more than a thousand dimensions, as explained by Chris Peikert, a researcher at University of Michigan College of Engineering [9]; this implies the use of hundreds or more than a thousand basis vectors to describe these lattices. In the simplest form, this kind of cryptography relies on the hardness of finding, in a such a high-dimensional lattice, two points (or vectors) that are relatively close to each other. The Closest Vector Problem is a generalization of the Shortest Vector Problem described as follows: given a basis B for a lattice $\mathcal{L}$ and a random vector v not necessarily in $\mathcal{L}$, find the vector (linear combination of the basis) in $\mathcal{L}$ that is closest to v in Euclidean distance.

[1]Vectors are linearly independent if they cannot be written as linear combinations of each other.

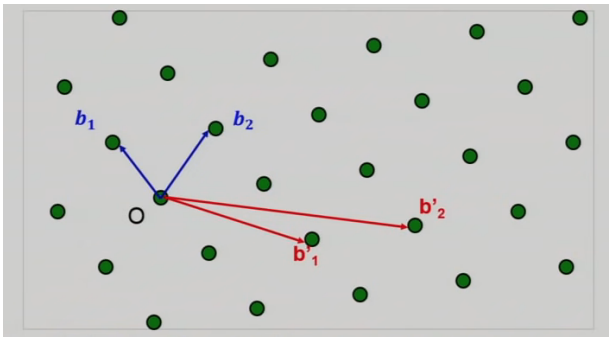

Fig. 1: Good Basis vs. Bad Basis

4) Hardness: In 1998, Miklós Ajtai proved that the Shortest Vector Problem (SVP) is NP-hard [10], and since Goldreich et al. proved the following year that the hardness for the Shortest Vector Problem implies equivalent hardness for the Closest Vector Problem [11], both of these fundamental problems on which lattice-based cryptography is based offer protection from both classical and quantum attacks. The best known algorithms for solving these lattice problems run in exponential time, even using approximation, and even using quantum computing, which is why public-key cryptosystems built on top of such problems, e.g., CRYSTALS-Dilithium, are less vulnerable than classical PKC.

5) Good Bases versus Bad Bases: It is important to note here that with lattices, which can be described with infinitely many bases, there is a notion of good bases and bad bases. In short, the difference comes down to the length, or Euclidean distance, of the basis vectors. A good basis for a lattice is comprised of short vectors, and a bad basis is comprised of long vectors, as visualized very simplistically in Figure 1, in which the blue basis is good and the red basis is bad. This is important because that lattice problems are easier to solve when the given basis is good and are harder to solve when the given basis is bad [8].

B. Short Integer Solution (SIS) and Learning with Errors (LWE) Problems

The Short Integer Solution (SIS) problem is equivalent to the Shortest Vector Problem on a given lattice, but is expressed a bit differently. SIS is posed as follows:

Given a matrix $A \in \mathbb{Z}_q^{n\times m}$, find a vector $r \in \mathbb{Z}^m$ such that

$$Ar = 0 \text{ (over } \mathbb{Z}_q^n\text{) and } ||r_2|| \leq \beta$$

Essentially, find a short (that is, with norm bounded by β) vector r of dimension m that when transformed by a matrix A produces 0, where A is in $\mathbb{Z}_q^{n\times m}$, meaning A is a matrix whose columns are in the set of all n-dimensional integer vectors modulo q. SIS is an average case problem, meaning any randomly selected instance of this problem will be hard

to solve, which is a necessary requirement for quantum-robust cryptography problems.

Building on top of SIS, the Learning with Errors (LWE) problem, introduced by Oded Regev in 2005 [12], is another quantum-robust lattice problem centered on finding a secret vector in a lattice from information clouded with intentional noise, or error. Similar to SIS, LWE has parameters n and q, with the addition of an error distribution parameter for adding noise that makes finding the secret more difficult. The LWE problem takes multiple forms. The LWE search problem is posed as follows:

Given a fixed n and q, and an error distribution,

find the secret $s \in \mathbb{Z}_q^n$ given many 'noisy inner products':

$$a_1 \leftarrow \mathbb{Z}_q^n, b_1 \approxeq \langle s, a_1 \rangle \bmod q$$

$$a_2 \leftarrow \mathbb{Z}_q^n, b_2 \approxeq \langle s, a_2 \rangle \bmod q$$

$$...$$

$$\Longrightarrow$$

$$a_1 \leftarrow \mathbb{Z}_q^n, b_1 = \langle s, a_1 \rangle + e_1 \in \mathbb{Z}_q$$

$$a_2 \leftarrow \mathbb{Z}_q^n, b_2 = \langle s, a_2 \rangle + e_2 \in \mathbb{Z}_q$$

$$...$$

where e_i for each noisy sample i is generally sampled from a discrete Gaussian distribution. Extending on the LWE search problem, the LWE decision problem poses the challenge of distinguishing (a_i, b_i) pairs generated as described above (with partial corruption via error vector sampling) from truly random and uniform (a_i, b_i) pairs. The LWE problem is another problem leveraged by CRYSTALS-Dilithium [13], and solving it has been proven to be at least as hard as solving approximate lattice problems in the worst case [12].

1) Discrete Gaussian Sampling: As discussed by János Folláth in their 2014 exploration of Gaussian sampling in lattice-based cryptography [14], many cryptographic algorithms involving lattices have been designed to require sampling from discrete Gaussian distributions as described in subsection II-B. According to Regev et al. [15, p. 2], "a discrete Gaussian distribution over some fixed lattice $\mathcal{L}$, denoted as $D_{\mathcal{L},s}$ for some parameter $s > 0$, is a distribution in which each lattice point is sampled with probability proportional to the probability density function of a continuous Gaussian distribution of width s evaluated at that point". These samplings according to literature are used because they provide a way to approximate uniform error vectors without knowing the lattice structure ahead of time. The developers of CRYSTALS-Dilithium, as confirmed by Gregor Seiler [13], intentionally avoided the use of discrete Gaussian sampling in their scheme because it is both difficult to implement correctly and it is even more difficult to achieve constant time execution. This omission contributed to one of their principal design considerations: the scheme needed to be easy to implement securely.

2) Ring-LWE and Module-LWE: One of the principal design goals held by the Dilithium developers was modularity, and a significant factor in achieving that goal was the decision to use Module-LWE rather than plain or Ring-LWE. [13]. To understand this reasoning, we need to understand each of these types of Learning With Errors problems. First, a ring in abstract algebra is just a set R that is closed under the two operations of addition and multiplication (meaning for any $x \in R, y \in R$, $x + y \in R$ and $x * y \in R$). Put even more simply by Dr. Chuck Easttom in his book on Quantum Computing Fundamentals [16, p. 80], "A ring is essentially just an abelian group[2] that has a second operation [of multiplication]". Albrecht et al. explains in their 2017 paper comparing Large Modulus Ring-LWE with Module-LWE [17] that the Ring-LWE problem is basically a form of LWE in which the n-dimensional vectors are replaced by polynomials with degree less than n. With this problem, you start by choosing "a ring R of dimension n, a modulus q and an error distribution χ over a related space of dimension n denoted $K_{\mathbb{R}}$". [17, p. 2]. Then, similar to how a_i is sampled from Z_q^n with plain LWE as shown in subsection II-B, you now instead sample a_i from $\frac{R}{qR}$, ending up with a similar sampling that looks like

$$a_i \leftarrow \frac{R}{qR}, b_i = \frac{1}{q} * a * s + e \bmod R^v$$

$$...$$

where R^v is the dual of ring R. A more precise definition is offered by Albrecht et al. in Section 2.3 of their paper [17, p. 9]. Albrecht et al. offers additional insight on Module-LWE (MLWE), showing that it was actually introduced to address shortcomings in both plain and Ring-LWE by "interpolating between the two" [17, p. 4]. He defines the MWLE informally as basically the same problem as RLWE but with the single elements a (sampled from $\frac{R}{qR}$) and s (the secret) from ring R replaced with module elements (vectors) over the same ring R. From Richard E. Borcherds' online course on Ring Theory [18], one ascertains that that modules in mathematics are defined essentially the same way as vector spaces [3], thus module elements are just vectors. On that note, according to Gregor Seiler's presentation [13], the key advantage of Module-LWE is using a whole vector over the ring R such that security of the digital signature scheme can easily be tuned by adjusting the length of that vector without ever having to change R itself. Since the same ring R is used for all security levels, he says the arithmetic for Dilithium only needs to be optimized one time.

C. Small Public Key and Signature

Another principal design goal shared by the Dilithium developers was the need for a small total size of the digital

[2]An abelian group is a group whose operation, e.g., addition, is commutative

[3]As briefly hinted at in subsection II-A, a vector space is a set of vectors closed under the operations of addition and scalar multiplication; another way of viewing it is as the set of all linear combinations of the basis vectors describing the vector space.

signature combined with the public key. Multiple design choices contributed to achieving this goal, as outlined in the following subsections. Ultimately, CRYSTALS-Dilithium offered the second-smallest combined size (after FALCON [4]) among the NIST PQC candidates.

1) Fiat-Shamir with Aborts: One of the team members behind Dilithium - Gregor Seiler, an IBM researcher - in his presentation on the scheme at the 2018 Conference on Cryptographic Hardware and Embedded Systems (CHES) [13], pointed out that the scheme design is based on a technique called Fiat-Shamir with Aborts. This technique, invented by Vadim Lyubashevsky in 2009 [19], opened the door to lattice-based signature schemes that could produce digital signatures on the order of about 50,000 bits compared to prior schemes that were producing signatures on the order of millions of bits in length; considering digital signatures need to be transferred over the network in addition to whatever messages are being signed to verify their integrity, this implied a significant advantage over previous techniques. Also, the security of Lyubashevsky's technique was based on the hardness of the shortest vector problem as described in subsection II-A.

2) Compression: CRYSTALS-Dilithium also leveraged developments from a series of papers in 2012 [20] and 2014 [21] to implement digital signature compression which helped them achieve signature sizes over 50% smaller by essentially only sending about half of the signature data over the network. On top of this, Dilithium introduced compression of the public key as well, achieving about a 60% smaller public key than previous schemes with only a slight increase in the size of the digital signature (by 100 bytes). Keeping in mind the Fiat-Shamir with Aborts advantage of reducing the signature size from millions of bits to 50,000 bits, a 100 byte increase is very minor.

III. Implementation

The CRYSTALS-Dilithium project is housed on GitHub [22] in two forms: one is the reference implementation written in plain C and the other implementation is optimized to run with the AVX2 instruction set.[4]. This section provides a simplified look at the Dilithium algorithm and includes performance data shared by Seiler in 2018 [13] for the reference version of the implementation.

A. The Algorithm

The Dilithium scheme is split into 3 main components: key generation, signing, and verification.

1) Key Generation:

$$A \leftarrow R^{5\times 4}$$

$$s_1 \leftarrow S_5^4, s_2 \leftarrow S_5^5$$

$$t = As_1 + s_2$$

$$pk = (A, t), sk = (A, t, s_1, s_2)$$

For key generation, a matrix A is obtained via random sampling. Then, two short vectors s_1 and s_2 are also sampled. Then, A, s_1, and s_2 are used to calculate t, an LWE vector. A and t then comprise the public key, while A, t, s_1 and s_2 comprise the secret, or private, key.

2) Signing:

$$y \leftarrow S_\gamma^4$$

$$w = Ay$$

$$c = \text{H}(\text{High}(w),\ M) \in B_{60}$$

$$z = y + cs1$$

$$||z||_\infty > \gamma - \beta \vee ||\text{Low}(w - cs_2)||_\infty > \gamma - \beta \implies \text{restart}$$

$$sig = (z, c)$$

The signing portion of the scheme uses a Fiat-Shamir transform. First, a short vector y is chosen, then Ay is stored in w. Then, the high bits of w are put into a hash function with a message M (signature compression comes from using only the high bits of w) and the digest (hash output) becomes the challenge polynomial c. Then, the sum $y + cs_1$ is stored in z. At this point, there is the important step of rejection sampling where basically the scheme is checking if either z or $w - cs_2$ reveals any secret information. If it does, the signing process is restarted. Otherwise, the digital signature is created using z and c.

3) Verification:

$$c' = \text{H}(\text{High}(\underbrace{Az - ct}_{w - cs_2}), M)$$

$$||z||_\infty \leq \gamma - \beta \wedge c' = c \implies \text{accept}$$

This verification process run by the message receiver first assigns another challenge polynomial c' using what equates to the high bits of w. It then runs yet another rejection sampling step to check if that new challenge polynomial c' equals the previous one c that was included in the received digital signature. This check corresponds to the classical digital signature verification step of comparing a message digest produced by hashing the received message against the message digest produced from decrypting the digital signature with the sender's public key. In both cases, a match indicates integrity and authenticity of the received message.

B. Overview

The main two operations important to the performance of the Dilithium scheme are polynomial multiplication in a fixed ring R (because it doesn't need to change thanks to MLWE as previously discussed) and the extensive parallel use of the SHAKE XOF (extendable output function)[5]. Seiler emphasizes that their implementation runs in constant time [13].

[4]Advanced Vector Extensions 2 (AVX2) is an extension to the Intel x86 instruction set that is able to handle single instruction multiple data (SIMD) instructions over 256-bit vectors. [23]

[5]An extendable output function (XOF) is just a cryptographic hash function that takes a bit string as input and can output a random string of bits of any desired length n.

C. Performance Data

Table I displays performance data from tests run with the reference implementation on the Intel Skylake i7-6600U processor. At a glance, one can see that multiplication and the use of the SHAKE extendable output function make up most of the time for the signing portion of the scheme by a significant margin. Table II displays similar performance data from tests run with the AVX2-implementation on the same processor. With this implementation, one quickly notices that each of the column totals dropped significantly, with the total median cycles for the signing portion of the scheme dropping from approximately 2.2 million to a competitive 635,000 on average.

IV. Conclusion

The CRYSTALS-Dilithium digital signature scheme submitted to and selected as one of a few winners of the NIST PQC competition exhibits many interesting characteristics, some of which are fundamental and shared by many lattice-based cryptography projects, some of which are unique and offer a competitive edge against other digital signature schemes in the quantum-resistant digital signatures arena. The scheme, like others, utilizes decades of lattice-based mathematics and number theory to provide robust protection against the upcoming surge of quantum-based attacks, and simultaneously offers implementation benefits like high performance, impressive efficiency, and strong public key and signature compression that will make deploying it practically beneficial.

References

[1] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, "Quantum annealing for prime factorization," *Scientific reports*, vol. 8, no. 1, pp. 1–9, 2018.

[2] L. Chen, D. Moody, and Y.-K. Liu, "Post-quantum cryptography," https://csrc.nist.gov/projects/post-quantum-cryptography, 2022, accessed: 2022-09-24.

[3] C. Boutin, "Nist announces first four quantum-resistant cryptographic algorithms," 2022.

[4] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhange, "Falcon: Fast-fourier lattice-based compact signatures over ntru," https://falcon-sign.info/, 2017, accessed: 2022-09-24.

[5] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The sphincs+ signature framework," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2129–2146, accessed: 2022-09-24.

[6] N. Manohar, "Hardness of lattice problems for use in cryptography," Ph.D. dissertation, 2016.

[7] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, 2018.

[8] Vinod Vaikuntanathan, "The mathematics of lattices i," https://www.youtube.com/watch?v=LlPXfy6bKIY, 2015, accessed: 2022-09-24.

[9] Chris Peikert, "Lattice cryptography: A new unbreakable code," https://www.youtube.com/watch?v=2IyotuA8eJc, 2019, accessed: 2022-09-24.

[10] M. Ajtai, "The shortest vector problem in l2 is np-hard for randomized reductions (extended abstract)," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 10–19. [Online]. Available: https://doi.org/10.1145/276698.276705

[11] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert, "Approximating shortest lattice vectors is not harder than approximating closest lattice vectors," *Information Processing Letters*, vol. 71, no. 2, pp. 55–61, 1999.

[12] O. Regev, "The learning with errors problem," *Invited survey in CCC*, vol. 7, no. 30, p. 11, 2010, accessed: 2022-09-24.

[13] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice based digital signature scheme," 2018.

[14] J. Folláth, "Gaussian sampling in lattice based cryptography," *Tatra Mountains Mathematical Publications*, vol. 60, no. 1, pp. 1–23, 2014, accessed: 2022-09-24.

[15] D. Aggarwal and O. Regev, "A note on discrete gaussian combinations of lattice vectors," *arXiv preprint arXiv:1308.2405*, 2013, accessed: 2022-09-24.

[16] C. Easttom, *Quantum computing fundamentals*. Addison-Wesley Professional, 2021.

[17] M. R. Albrecht and A. Deo, "Large modulus ring-lwe $\geq$ module-lwe," *Cryptology ePrint Archive*, 2017.

[18] Richard E Borcherds, "Rings and modules 1 introduction," https://www.youtube.com/watch?v=4WmIDodIgaclist=PL8yHsr3EFj52XDLrmvrFDgw 2021, accessed: 2022-09-24.

[19] V. Lyubashevsky, "Fiat-shamir with aborts: Applications to lattice and factoring-based signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2009, pp. 598–616, accessed: 2022-09-24.

[20] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 530–547.

[21] S. Bai and S. D. Galbraith, "An improved compression technique for signatures based on learning with errors," in *Cryptographers' Track at the RSA Conference*. Springer, 2014, pp. 28–47.

[22] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Dilithium," https://github.com/pq-crystals/dilithium, 2018.

[23] Intel, "Intrinsics for intel advanced vector extensions 2 (intel avx2)," https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/intrinsics/intrinsics-for-avx2.html, 2022, accessed: 2022-09-24.

	Key Generation	Signing	Signing (average)	Verification
Multiplication	89,591	987,666	1,280,053	143,924
SHAKE	178,487	314,570	377,068	161,079
Modular Reduction	11,944	120,793	163,017	10,626
Rounding	6,586	108,412	137,324	11,821
Rejection Sampling	60,740	76,893	94,607	28,082
Addition	8,008	58,696	79,498	10,723
Packing	7,114	17,183	18,856	8,883
Total	381,178	1,778,148	2,260,429	396,043

TABLE I: Median cycles of 5000 executions on Intel Skylake i7-6600U processor for reference implementation

Median cycles of 5000 executions on Intel Skylake i7-6600U processor				
	Key Generation	Signing	Signing (average)	Verification
Multiplication	15,974	155,721	201,347	25,471
SHAKE	96,779	170,232	205,847	90,921
Modular Reduction	1,034	7,902	10,541	708
Rounding	728	7,541	9,904	2,479
Rejection Sampling	62,272	67,193	81,278	27,737
Addition	8,028	46,755	62,453	8,659
Packing	6,997	16,200	17,526	8,712
Total	199,306	510,298	635,019	174,951

TABLE II: Median cycles of 5000 executions on Intel Skylake i7-6600U processor for AVX2-optimized implementation