applied sciences

MDPI

Article

Performance Analysis of Post-Quantum Cryptography Algorithms for Digital Signature

Filip Opiłka [1], Marcin Niemiec [1,*], Maria Gagliardi [2] and Michail Alexandros Kourtis [3]

[1] AGH University of Krakow, Institute of Telecommunications, Mickiewicza 30, 30-059 Krakow, Poland
[2] Sant'Anna School of Advanced Studies, 56127 Pisa, Italy; maria.gagliardi@santannapisa.it
[3] National Center for Scientific Research "Demokritos", 15310 Athens, Greece; akis.kourtis@iit.demokritos.gr
* Correspondence: marcin.niemiec@agh.edu.pl

Abstract: In the face of advancing quantum computing capabilities posing significant threats to current cryptographic protocols, the need for post-quantum cryptography has become increasingly urgent. This paper presents a comprehensive analysis of the performance of various post-quantum cryptographic algorithms specifically applied to digital signatures. It focuses on the implementation and performance analysis of selected algorithms, including CRYSTALS-Dilithium, Falcon, and SPHINCS+, using the liboqs library. Performance tests reveal insights into key pair generation, file signing, and signature verification processes. Comparative tests with the well-known and popular RSA algorithm highlight the trade-offs between security and time efficiency. The results can help to select secure and efficient ciphers for specific 5G/6G services.

Keywords: post-quantum cryptography; digital signature; cipher; secure services; efficiency

Citation: Opiłka, F.; Niemiec, M.; Gagliardi, M.; Kourtis, M.A. Performance Analysis of Post-Quantum Cryptography Algorithms for Digital Signature. *Appl. Sci.* **2024**, *14*, 4994. https://doi.org/10.3390/app14124994

Academic Editor: Christos Bouras

Received: 18 May 2024
Revised: 1 June 2024
Accepted: 4 June 2024
Published: 7 June 2024

1. Introduction

In an era of access to wireless networks and continuous flow of information, it is requisite to ensure their security. For this purpose, numerous solutions have been developed to ensure the confidentiality, integrity, and authenticity of data. However, sometimes, the security they guarantee is conditional. In the case of currently used asymmetric cryptography, this condition is the difficulty of solving a selected mathematical problem that is achievable by quantum computers. Although, presently, they are not able to threaten the security of our data transmitted on the Internet, it is necessary to stay ahead of potential dangers by preparing solutions that are resistant to attacks involving quantum computers, especially if we consider data protection in future 6G networks.

Data security that is traditionally reliant on cryptographic protocols supported by such ciphers as RSA (Rivest–Shamir–Adleman) or ECC (Elliptic Curve Cryptography) faces imminent threats from quantum computing advances. Quantum algorithms such as Shor's and Grover's algorithms pose a major threat to current encryption methods. Quantum computers potentially will be able to break RSA and ECC within a short time (e.g., minutes). Such a task is unachievable by classical computers in a reasonable timeframe. Quantum computing, while offering disruptive abilities regarding modern technology, introduces new vulnerabilities in data security. This inherent paradox calls for the development of post-quantum cryptography (PQC)—cryptographic algorithms that quantum computers cannot break easily.

It is worth mentioning that the different characteristics of new cryptographic algorithms and solutions are important to be evaluated, also taking into account a legal dimension. Indeed, understanding the advantages and disadvantages of algorithms—as far as security levels and real use possibilities are concerned (e.g., 5G/6G services)—is crucial in the transition to post-quantum cryptography, since it can be helpful in the selection of appropriate solutions for specific security contexts, taking into consideration the possible

different legal requirements or the values and rights involved in the different contexts of real use.

The motivation behind addressing the topic of this paper is that cybersecurity must anticipate and stay ahead of imminent threats, and this is undoubtedly a reality thanks to quantum computers and their computing capabilities. An additional aspect supporting delving into the realm of post-quantum cryptography are European quantum initiatives which support excellence in quantum research. A great example here is the PQ-REACT project [1]—a component of the broader Horizon Europe program. This initiative is dedicated to creating, evolving, and validating a framework designed to facilitate an efficient and smooth transition from traditional cryptography to post-quantum cryptography.

In the current landscape, various post-quantum cryptographic algorithms are being proposed and evaluated for their effectiveness and efficiency. The National Institute of Standards and Technology (NIST) is playing a pivotal role in this transition, working towards the standardization of post-quantum cryptographic algorithms. The exploration and implementation of these algorithms, particularly in the context of digital signatures, is the focus of this paper [2]. It is worth mentioning that hardware acceleration of post-quantum cryptography can improve the performance; however, the speed factor strongly depends on the family of the algorithm [3–6].

Through a comprehensive analysis of current cryptographic practices and an insightful look at post-quantum solutions (resistant to attacks involving a quantum computer), this study aims to shed light on the challenges and opportunities of protecting our digital future. This paper has the objective to demonstrate that post-quantum algorithms can be used for one of the services that maintains data security—digital signatures. To accomplish this, it is necessary to implement a selected post-quantum cryptography algorithm to be used for this mechanism. Such an approach can verify the potential of algorithms that face the computing power of quantum computers, ensuring the proper functioning of widely used solutions such as digital signature mechanisms or other services based on asymmetric cryptography. Furthermore, it is particularly helpful to present a usage example and provide results regarding the time efficiency of the tested solutions.

This article consists of six sections. The motivation, objective, and scope of this work are presented in Section 1. Section 2 introduces modern cryptography, including the idea and application of asymmetric ciphers. Section 3 describes the risk of breaking currently used encryption algorithms by quantum computers and introduces the concept of post-quantum cryptography. The testbed and some details regarding implementation are described in Section 4. Section 5 is dedicated to a performance analysis of post-quantum cryptography. The results of the efficiency analysis are presented, including the performance of key pair generation, file signing, and signature verification for various post-quantum cryptographic algorithms. This article is summarized in Section 6.

2. Modern Cryptography

Cryptography has become an integral part of modern communication. It has significantly evolved over the centuries, resulting in the development of two primary categories: symmetric and asymmetric cryptography. Symmetric cryptography uses only one key, with which both encryption and decryption are executed. It is a simple and remarkably effective method, but it is not without its weaknesses. Among these, the biggest is the way of exchanging or establishing the key between the parties. Fortunately, other methods can be entrusted with this task. For example, asymmetric cryptography or quantum key distribution can be used to transfer the key [7]. Under the condition of key confidentiality and integrity, symmetric cryptography can offer reasonable speed of operations and security.

Asymmetric cryptography solves the biggest problem of symmetric cryptography, namely secure key exchange or key establishment. For encryption and decryption, a key pair, public and private, is used. The private key is obligatorily kept confidential, as only with this key can the owner decrypt communications addressed to him. However, the public key is shared to give senders the ability to encrypt messages. The security of asym-

metric cryptography is fundamentally based on the difficulty of mathematical problems. Among these, we can distinguish the factorisation of large prime numbers, which is the basis of RSA, and systems based on elliptic curves, for example, ECC. Asymmetric cryptography also underlies digital signatures.

Digital signatures are one of the fundamental applications of asymmetric cryptography. Digital signatures ensure the non-repudiation of data and its integrity. In order to create a signature, several elements are necessary: a private key, a signature algorithm, and usually, a hash function. A scheme of digital signature is presented in Figure 1. Firstly, the hash is computed (1) to sign a file—this process is not obligatory but supports efficiency of signature process. Subsequently, the signature algorithm is used. Its inputs are the hash of the file to be signed and the private key of the signer (2). The result of this action is a signature that proves the undeniable authorship of the file. This collection—original data and signature—is sent to the recipient (3).

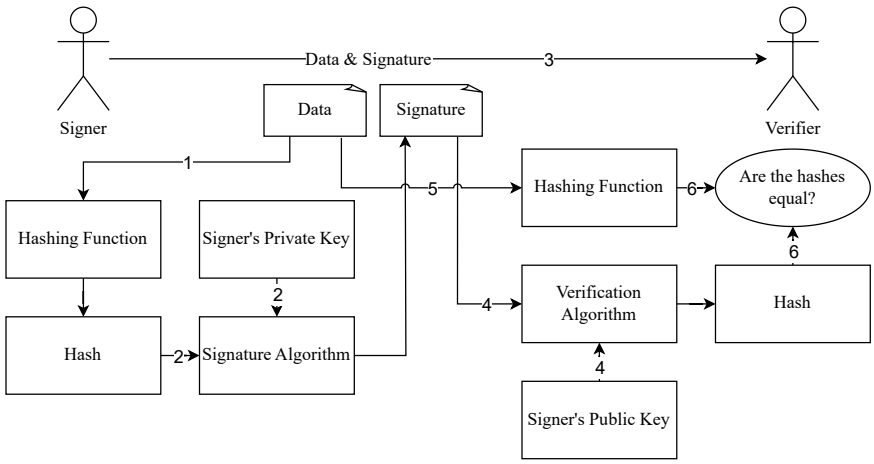

Figure 1. Digital signature scheme.

The person verifying the signature, apart from the file and its signature, also needs to have information on what hashing function was used and what signature algorithm was applied. The verification process begins by applying the signature algorithm together with the signer's public key to the received signature (4). The result of this operation is the hash of the original data. The next step is to separately calculate the hash of the signed file using a known hash function (5). Then, these two hashes must be compared (6). If they are equal, this confirms the non-repudiation and integrity of the signed data. Verification process is also presented in Figure 1.

3. Post-Quantum Cryptography

Quantum computers represent a breakthrough in the computing domain. In contradistinction to classical devices, they use the principles of quantum mechanics to process information. The smallest data unit in conventional computers is called the bit; it can take the value either 0 or 1. However, in quantum computers, this is replaced by qubits, which can be in a superposition of states 0 and 1. This allows them to perform many operations simultaneously, which leads to increased speed. The challenges faced by current prototypes of quantum computers are their stability and scalability. In order to maintain the quantum state, it is necessary to maintain extremely low temperatures as close as possible to the absolute zero. Subsequent factors that negatively affect qubits are all kinds of vibrations or magnetic fields [8]. Even though quantum computers still require a lot of research, they pose a potential threat to current cryptography. Therefore, there is a reason for the formation and development of post-quantum cryptography.

3.1. Quantum Algorithms

One of the most well-known concepts in the field of quantum technology is the Shor's algorithm. This is a quantum algorithm introduced by Peter Shor in 1994 [9]. It was designed for the purpose of integer factorisation. Currently, this problem is too demanding for classical computers to perform the calculation in measurable time. The computational complexity of classical algorithms is exponential, while the Shor's algorithm offers polynomial complexity, which results in significantly faster computations. This solution combines conventional mathematical approaches and the principles of quantum mechanics [10]. The currently widely used RSA algorithm is based precisely on the complexity of the large number factorisation problem.

Another example is the Grover's algorithm. It was designed by Lov Grover in 1996, and its purpose is to search in unordered databases [11]. Whereas this task with the classical algorithm has a computational complexity of $O(N)$, the Grover algorithm achieves a complexity of $O(\sqrt{N})$. The reason for this is the use of quantum superposition, which results in the ability to check multiple potential solutions simultaneously [12]. Its practical application may threaten cryptography algorithms, but the leap in speed is not that significant. It is believed that the lengthening of the keys is sufficient to constitute resistance to attacks involving it. Nevertheless, even if the Grover algorithm does not represent a major threat as the Shor's algorithm, it also has a notable role regarding the risk to current cybersecurity.

3.2. Post-Quantum Ciphers

Post-quantum cryptography is a proactive response to the threats brought on by the emergence of Shore and Grover algorithms. Due to the potential ability to break hitherto widely used algorithms like RSA or ECC, there have to be solutions supporting asymmetrical data encryption but resistant to attacks by quantum computers. This is precisely the purpose behind the development of post-quantum algorithms. They are divided into four main categories.

- Lattice-Based Cryptography is one of the leading candidates of the post-quantum cryptography. They are based on the complexity of problems such as the Shortest Vector Problem (SVP), as well as Learning With Errors (LWE) [13]. The security of these solutions comes from the difficulty of solving some lattice problems in multidimensional spaces. The problem, for example, is finding the constants by which the basis vector should be multiplied, which is not an easy task for quantum computers. Some of the well-known algorithms belonging to this group are NTRU, CRYSTALS-Dilithium, and Falcon.
- Hash-Based Cryptography is a group that bases its security on appropriate hash functions and schemes such as the Merkle signature scheme [13]. Currently, no efficient quantum algorithm is known to be able to break a cryptographic hash function; however, the disadvantage of such a solution may be the limit of signatures generated using a particular key pair. An example of an algorithm that belongs to this group is SPHINCS+.
- Code-Based Cryptography is a group that bases its security on correction codes [14]. Examples of algorithms with such properties are McEliece and its variation Niederreiter. The difficulty of these solutions lies in decoding a general linear code.
- Isogeny-Based Cryptography is the last of the four main groups of the post-quantum algorithms. The solutions within it are based on the properties of elliptic curves and the Claw Finding problem. This involves the operation with elliptic curves that have the property of isogeny [15].

Additionally, it is worth comparing well-known post-quantum algorithms, the effectiveness of which is analyzed by the authors in this paper. The CRYSTALS-Dilithium, Falcon, and SPHINCS+ algorithms are briefly discussed, each representing a different approach to post-quantum cryptography.

- CRYSTALS-Dilithium is a lattice-based algorithm. It was designed for digital signatures, and it was a major finalist in the NIST competition in this exact category, which is its main advantage also in the context of this study. The security of this algorithm lies in the difficulty level of lattice problems over modular lattices [16]. As a lattice-based algorithm, it benefits from the complexity of the Shortest Vector Problem and Learning With Errors problems, which are also considered unbreakable using quantum computers.
- Falcon is an acronym for 'Fast-Fourier Lattice-based Compact Signatures over NTRU'. It was also another candidate in the NIST competition. It uses the lattice structure of NTRU and the Fast Fourier Transform (FFT) to improve performance. Its main advantage is the small size of the generated signatures, which is especially important in the context of high-volume data transmission [17].
- SPHINCS+ represents a different approach to the previous examples, as it is an algorithm based on hash functions. Its advantage is the ability to be based on different hash functions, e.g., SHA2 or SHAKE, allowing it to be adapted according to specific security requirements. On the other hand, its disadvantage is the relatively large size of the generated signatures [18].

Each of the mentioned algorithms represents a sufficient security level. These three algorithms were selected by NIST during the standardization process of algorithms to be intended for digital signatures. Considering the availability of all three algorithms presented, they have been implemented to use for digital signature purposes and tested.

4. Implementation

The performance analysis of post-quantum cryptography algorithms is based on implemented CLI application and *liboqs* library [19]. *Liboqs* provides cryptography-related elements, and Click is responsible for creating the command line interface. All the elements combine to form a development environment. These solutions not only provide a functional interface but also facilitate efficient programming.

4.1. Liboqs Library

The *liboqs* library is a set of post-quantum algorithms prepared as a part of the Open Quantum Safe project [20]. The library has been designed to function as a comprehensive testing and evaluation toolkit for researchers and developers, providing them with the opportunity to experiment with a range of cryptographic schemes that are believed to be resistant to potential attacks from quantum computers.

The library is written in C, providing a performance-optimized foundation for cryptographic operations. To ensure broader accessibility and integration into various software stacks, liboqs provides language wrappers, including a Python wrapper. This wrapper enables the use of post-quantum algorithms in Python applications and has been included in the implementation. In order to use *liboqs-python*, it is first necessary to build and install the *liboqs* library and, then, to properly configure the Python virtual environment.

4.2. Application

The implemented application can sign a selected file and verify digital signatures using post-quantum algorithms. Its main functionality is based on the `pq-sign` command, which can be used in the terminal. An example of the sign and verification process is presented in Figure 2. An important function is `generate-keypair`, which generates a private key used to sign documents and a public key, which can be used to verify signatures. The `sign` command is a central feature of the application—it lets users digitally sign documents, ensuring their authenticity and integrity. Verification of the digital signature is carried out with the `pq-sign verify` command. Additionally, the `pq-sign ls` command has been designed to provide a comprehensive list of all post-quantum cryptographic algorithms currently supported by the application.

```
filip@filip-post-quantum:~$ pq-sign sign example.txt Dilithium2_private_key.key
Signing example.txt file with Dilithium2...
Signed
filip@filip-post-quantum:~$ pq-sign verify example.txt signature_example_Dilithium2.sig Dilithium2_public_key.pub
Is signature valid: True
```

Figure 2. Signature and correct verification.

Table 1 provides specifications of the available algorithms [19]. It details the algorithm names along with their corresponding security strengths, represented by the NIST security levels [21]. The security levels of cryptographic algorithms or systems are determined based on a numerical value associated with the amount of work or operations required to compromise them. These levels range from Level 1, offering the lowest level of security, to Level 5, which provides the highest level of security [22]. The numbers at the end of the Dilithium algorithm name correspond to NIST's declared security level. The table also provides the sizes of the public keys, secret keys, and signatures, all expressed in bytes. And it is the size of the generated signature that is crucial regarding distinguishing SPHINCS+ algorithms. It is important to note that comprehending the trade-offs between the security level and resource requirements is imperative for users. This understanding can assist in selecting an appropriate algorithm for a specific security context.

Table 1. Available algorithms specifications [19].

Algorithm	Claimed NIST Level	Public Key Size (Bytes)	Secret Key Size (Bytes)	Signature Size (Bytes)
Dilithium2	2	1312	2528	2420
Dilithium3	3	1952	4000	3293
Dilithium5	5	2592	4864	4595
Falcon-512	1	897	1281	666
Falcon-1024	5	1793	2305	1280
SPHINCS+-SHA2-128f-simple	1	32	64	17,088
SPHINCS+-SHA2-128s-simple	1	32	64	7856
SPHINCS+-SHA2-192f-simple	3	48	96	35,664
SPHINCS+-SHA2-192s-simple	3	48	96	16,224
SPHINCS+-SHA2-256f-simple	5	64	128	49,856
SPHINCS+-SHA2-256s-simple	5	64	128	29,792
SPHINCS+-SHAKE-128f-simple	1	32	64	17,088
SPHINCS+-SHAKE-128s-simple	1	32	64	7856
SPHINCS+-SHAKE-192f-simple	3	48	96	35,664
SPHINCS+-SHAKE-192s-simple	3	48	96	16,224
SPHINCS+-SHAKE-256f-simple	5	64	128	49,856
SPHINCS+-SHAKE-256s-simple	5	64	128	29,792

4.3. Testbed

The project's implementation was based on a WMWare virtual machine with Linux Debian installed. The advantage of this solution is the ease of management of the isolated environment and the ability to recreate the runtime environment. Virtual machines can be saved, cloned, and shared, allowing the exact setup to be reproduced on any compatible host machine. The detailed parameters of the used virtual machine are described in Table 2.

Table 2. Technical specification of the virtual machine.

Parameter	Description
VMWare version	VMWare Workstation 16
Operating system	Ubuntu 22.04.1 LTS
Memory	12 GB
Processors	8 cores
Host processor	Intel Core i7-9750H CPU @ 2.60 GHz
Disk space	30 GB SSD

5. Performance Analysis

This section is dedicated to the analysis of efficiency, focusing specifically on time-based metrics. The idea behind performance testing is to examine post-quantum algorithms in the proposed implementation comprehensively. There are several reasons why time is a key factor regarding cryptographic algorithms. The crucial ones are listed below.

- Scalability—algorithms with faster processing times are better suited for managing large datasets or high-traffic scenarios. If the algorithm is time-efficient, the whole system will be able to handle more operations.
- User Experience—cryptographic algorithms should be fast enough to ensure smooth use of the applications. Latency drastically affects the experience of using a particular service. Additionally, some applications need real-time operation which cannot be supported by inefficient cryptography algorithms.
- Security—if the execution time of a particular algorithm is excessively long, this can create an opportunity for an attacker to collect information about the time this service needs. By measuring the time in different examples of input data, it is possible to obtain information such as the length of used keys. To prevent this vulnerability, it is necessary to ensure the used solutions are time-optimized and designed to ensure that the execution time is constant, which means they are independent of the input data.
- System Performance—cryptographic operations are almost always part of larger systems, and therefore, their time efficiency translates into the aggregate performance of complex services.

Moreover, cryptography algorithms used in practice require balancing security strength and time efficiency. The proper trade-off is crucial, taking into account the usability of such solutions.

In order to test the performance of post-quantum algorithms in a digital signature mechanism, it was necessary to have files that would be signed. It was assumed that entire files are encrypted by the private keys of each tested algorithm. To obtain a representative spectrum of results, three files of size 10 MB, 100 MB, and 1 GB were generated. This was achieved using the native Linux command dd, which is used to copy and convert data. Using this command, pseudo-random data from the /dev/urandom file were copied.

Every functionality was tested for each algorithm. Each measurement was repeated 101 times. The first measurement time for a given algorithm was noticeably longer and was not included in the graphs and further analysis. This is due to the data being read from the disk and loaded into cache memory. By omitting the first measurement in each series, it was possible to exclude the disk read speed factor from the measurement component and obtain the proper time needed to execute the tested algorithm. Therefore, the presented graphs show the average execution time of the algorithm calculated from 100 iterations of a given operation. Error bars showing the Standard Error of the Mean (SEM) are also included for each algorithm. Before starting the measurements, it was necessary to prepare the various components. For the file signing and signature verification processes, three files of 10 MB, 100 MB, and 1 GB were prepared. A corresponding key pair was also generated for each algorithm. During the key pair generation evaluation, the files containing the key pairs were deleted after each iteration. Regarding the file signing tests, each iteration was run on the same file, and as with the key pair generation, the resulting file was deleted after each iteration. Following the same procedure, the signature verification tests were performed on the same file for each algorithm, in order to compare the results of the measurements.

5.1. Measurement Results

The performance of key pair generation for various post-quantum cryptographic algorithms was tested firstly. Figure 3 presents the results of this scenario. It can be observed that Dilithium algorithms had the best time performance. Algorithms from the SPHINCS+ group with the suffix 'f' also achieved a gratifying result, distinguishing themselves from the second part of this group. However, it should be remembered that they are optimized for speed, not size. Therefore, they generate a longer signature than those

with the suffix 's', as shown in Table 1. Both of these groups obtained times an order of magnitude larger than the Falcon and SPHINCS+ algorithms with the suffix 's'. However, it is thought-provoking that the SPHINCS + SHA-192s-simple and SPHINCS+SHAKE-192s-simple algorithms exhibited longer execution times relative to their counterparts with longer keys, namely SPHINCS + SHA-256s-simple and SPHINCS + SHAKE-256s-simple, respectively. This performance trend was counter-intuitive and at odds with the behavior observed from the other algorithms. For example, the SPHINCS+ algorithms with the suffix 'f' and the Falcon algorithms demonstrated a dependency that the larger the key size, the longer the execution time. This unexpected behavior may be a direct consequence of the way certain algorithms are implemented in the library.

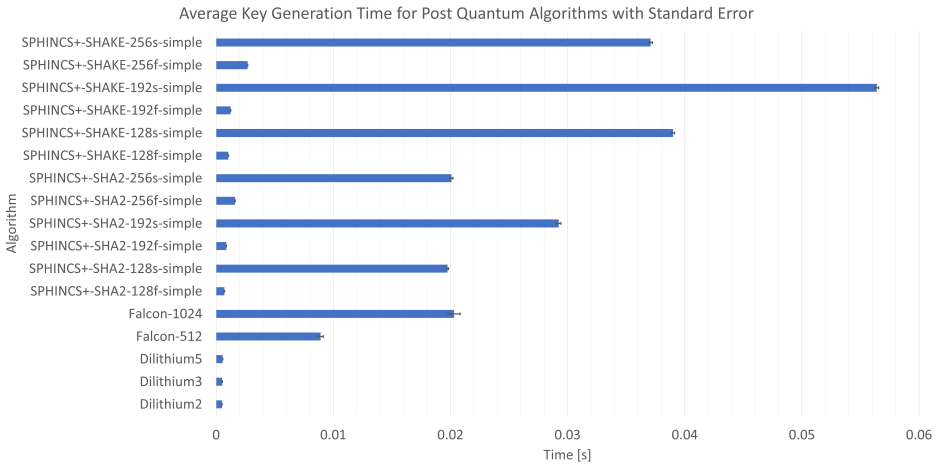

Figure 3. Average key generation time.

Figures 4–6 refer to the file signing function. They show the average execution time obtained by each algorithm when signing files of size 10 MB, 100 MB, and 1 GB, respectively. It is a reasonable trend that the larger the size of the file to be signed, the longer this operation takes. In all three cases, the Dilithium and Falcon algorithm groups achieved the best time performance. Similar to Figure 3, showing the average key pair generation times, a disparity between the SPHINCS+ algorithms with 's' and 'f' suffixes was noticeable in the advantage of the speed-optimized ones. This result is in accordance with expectations. However, the difference decreased as the size of the signed file increased. As in Figure 3, results that are counter-intuitive were also present, namely in the algorithms of the SPHINCS+ group—examples of each file size were visible where the algorithm operating with a shorter key achieved a worse time than its counterpart operating with a longer key. As the performance of the algorithms in the two groups, Dilithium and Falcon, were very similar, further analysis was carried out. The purpose of this additional research was to gain a detailed insight into their performance. The results are presented in Figure 7. Based on the analysis of the results, it can be concluded that the Dilithium algorithms achieve better performance even for relatively small signed data sizes. In accordance with intuition and measurements on larger files, the Falcon-1024 algorithm performed almost twice as long as its Falcon-512 equivalent due to the twice-as-long key.

Figures 8–10 show the average verification time for signed files of 10 MB, 100 MB, and 1 GB, respectively. Figure 8 referring to the case with the smallest file size presents that the Dilithium algorithms performed the worst, which is completely different from the previous results. In this case, the SPHINCS + SHA2-192 and SPHINCS + SHA2-256 algorithms achieved the best result. The SPHINCS + algorithm based on the SHAKE hash function achieved comparable results regardless of the key size. In the operation involving a 100 MB file as shown in Figure 9, the SPHINCS + SHAKE-256s-simple algorithm recorded the longest duration. The trio of SPHINCS + SHA2 algorithms also showed slight variations. However, apart from that, the results were relatively comparable. The results shown in

Figure 10 (1 GB file) were the most consistent across all evaluated scenarios. The outlier was a set of four algorithms from the SPHINCS+ group, specifically two variants each of SPHINCS + SHA2-192 and SPHINCS + SHA2-256, which recorded significantly improved results over the other tested examples. Nevertheless, the variance did not exceed 30% compared to other algorithms. The precise values are presented in Table 3 (key generation and signing) and Table 4 (verification of signatures).

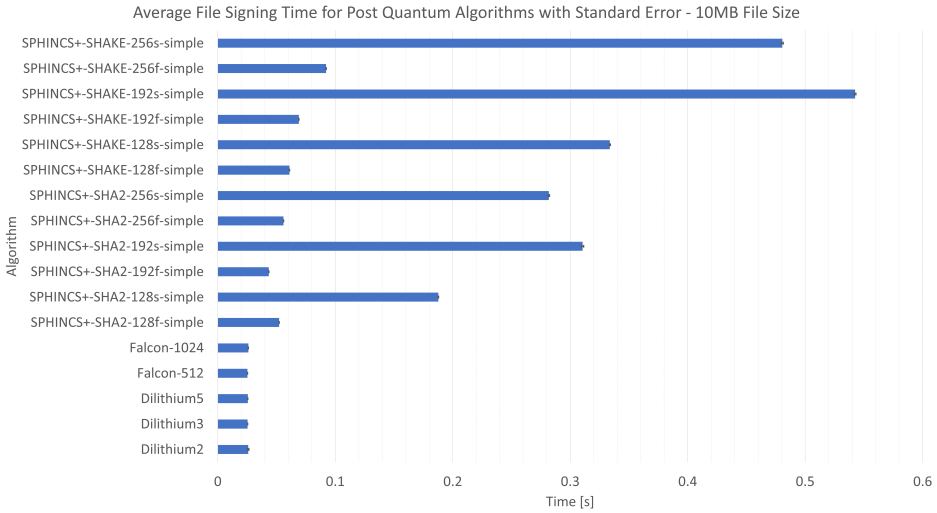

Figure 4. Average file signing time—10 MB file size.

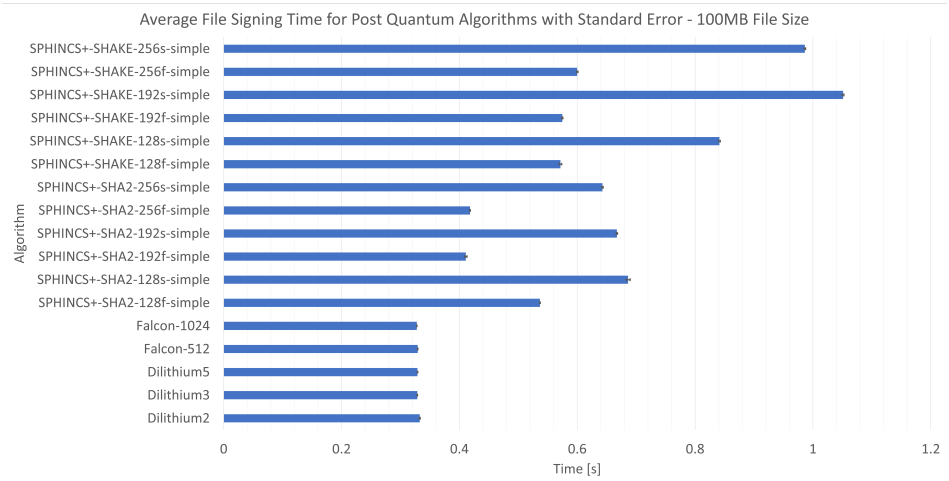

Figure 5. Average file signing time—100 MB file size.

The conclusion that can be drawn regarding the SPHINCS+ group algorithms is that for signing large-sized files, algorithms with an 's' suffix would be a better choice. The reason for this is that when handling large files, the disparity between algorithms optimized for speed and those optimized for size decreases. However, algorithms with the suffix 's' retain the advantage of a significantly smaller size of the generated signature. Although algorithms with the suffix 'f' present significantly reduced key pair generation times, in the usual use case, a once-generated key pair is used to sign multiple files. Additionally, another distinctive feature of the SPHINCS+ group of algorithms is worth mentioning: regardless of the file size, the disparity in verification time between speed-optimized and size-optimized variants is not significant.

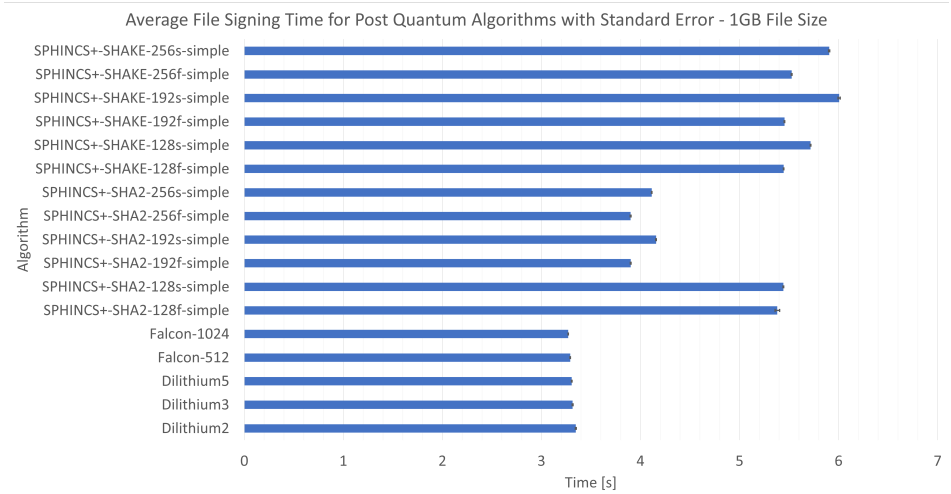

Figure 6. Average file signing time—1 GB file size.

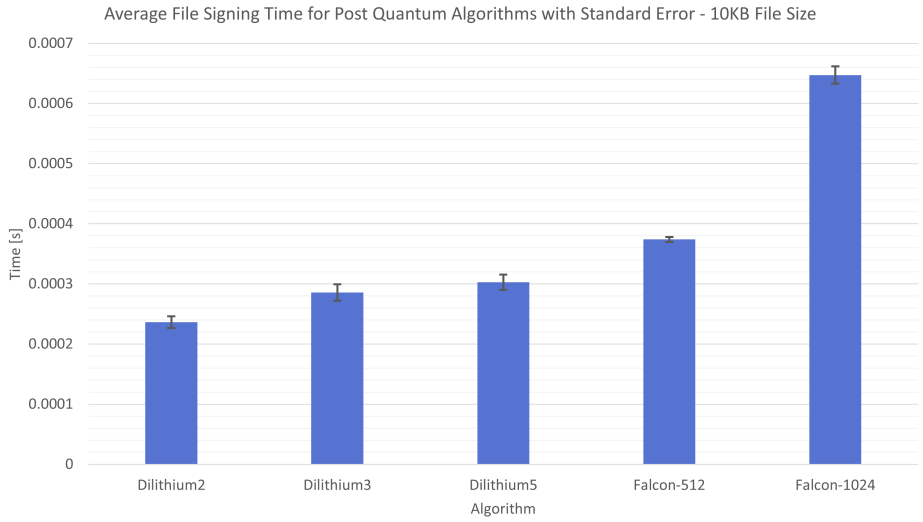

Figure 7. Average file signing time—10 KB file size.

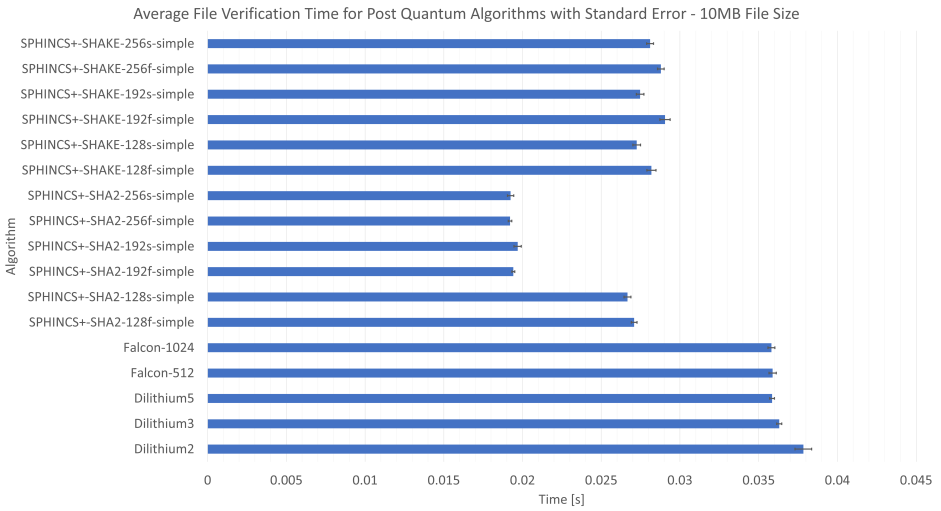

Figure 8. Average file verification time—10 MB file size.

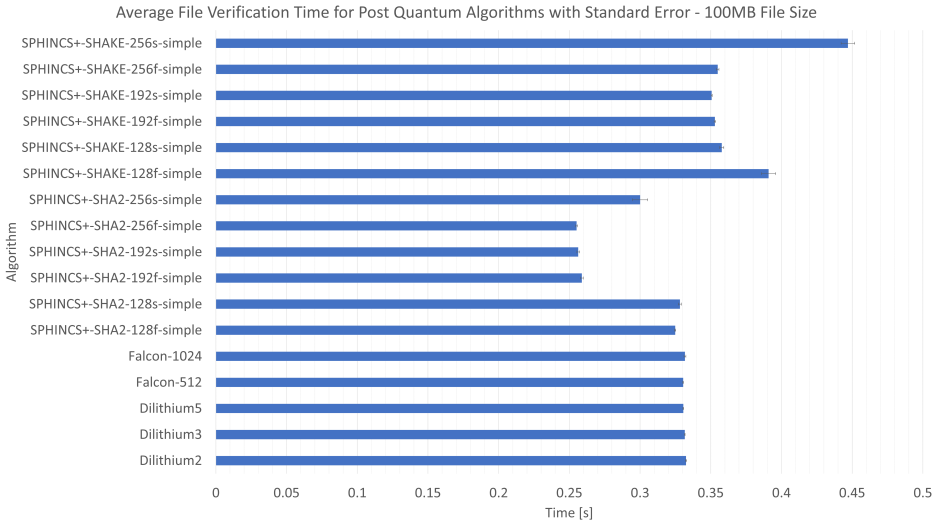

Figure 9. Average file verification time—100 MB file size.

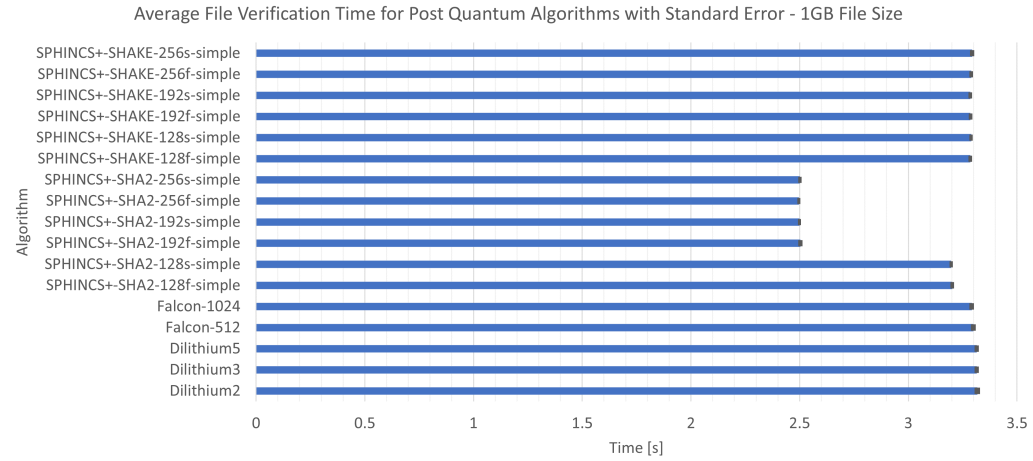

Figure 10. Average file verification time—1 GB file size.

Table 3. Performance measurements of the post-quantum algorithms for keypair generation and file signing.

Algorithm \ Function	Keypair Generation [s $\times 10^{-5}$]	Signing 10 MB File [s $\times 10^{-5}$]	Signing 100 MB File [s $\times 10^{-5}$]	Signing 1 GB File [s $\times 10^{-5}$]
Dilithium2	50 ± 2	2592 ± 6	33,273 ± 7	334,713 ± 49
Dilithium3	51 ± 2	2538 ± 1	32,852 ± 5	331,453 ± 44
Dilithium5	55 ± 1	2553 ± 1	32,892 ± 4	330,594 ± 28
Falcon-512	891 ± 23	2529 ± 1	32,931 ± 4	329,089 ± 25
Falcon-1024	2031 ± 51	2602 ± 1	32,776 ± 2	327,077 ± 14
SPHINCS + SHA2-128f-simple	70 ± 1	5223 ± 1	53,661 ± 6	538,160 ± 230
SPHINCS + SHA2-128s-simple	1975 ± 8	18,773 ± 2	68,634 ± 35	544,226 ± 40
SPHINCS + SHA2-192f-simple	87 ± 1	4338 ± 1	41,089 ± 21	390,097 ± 37
SPHINCS + SHA2-192s-simple	2924 ± 19	31,049 ± 10	66,722 ± 11	415,751 ± 28
SPHINCS + SHA2-256f-simple	162 ± 1	5594 ± 1	41,797 ± 7	390,028 ± 45
SPHINCS + SHA2-256s-simple	2009 ± 12	28,183 ± 5	64,288 ± 12	411,376 ± 21
SPHINCS + SHAKE-128f-simple	104 ± 1	6104 ± 1	57,144 ± 22	544,599 ± 24
SPHINCS + SHAKE-128s-simple	3900 ± 12	33,373 ± 4	84,141 ± 13	571,716 ± 30
SPHINCS + SHAKE-192f-simple	124 ± 1	6907 ± 1	57,464 ± 10	545,416 ± 31
SPHINCS + SHAKE-192s-simple	5639 ± 16	54,257 ± 6	105,115 ± 16	600,620 ± 110
SPHINCS + SHAKE-256f-simple	269 ± 1	9224 ± 2	59,994 ± 13	552,686 ± 39
SPHINCS + SHAKE-256s-simple	3709 ± 15	48,057 ± 8	98,623 ± 13	590,551 ± 61

Table 4. Performance measurements of the post-quantum algorithms for verification of signatures.

Algorithm \ Function	Verifying 10 MB File [s $\times 10^{-5}$]	Verifying 100 MB File [s $\times 10^{-5}$]	Verifying 100 MB File [s $\times 10^{-5}$]
Dilithium2	3784 ± 5	33,253 ± 4	331,713 ± 62
Dilithium3	3630 ± 2	33,162 ± 2	331,443 ± 38
Dilithium5	3585 ± 2	33,060 ± 2	331,443 ± 38
Falcon-512	3588 ± 2	33,046 ± 2	329,893 ± 45
Falcon-1024	3581 ± 2	33,177 ± 7	32,9181 ± 44
SPHINCS+SHA2-128f-simple	2709 ± 2	32,485 ± 4	320,258 ± 21
SPHINCS+SHA2-128s-simple	2667 ± 2	32,815 ± 12	319,706 ± 16
SPHINCS+SHA2-192f-simple	1942 ± 1	25,881 ± 11	250,359 ± 29
SPHINCS+SHA2-192s-simple	1969 ± 2	25,624 ± 10	249,951 ± 11
SPHINCS+SHA2-256f-simple	1921 ± 1	25,513 ± 7	249,642 ± 12
SPHINCS+SHA2-256s-simple	1924 ± 2	30,004 ± 54	250,094 ± 15
SPHINCS+SHAKE-128f-simple	2819 ± 3	39,087 ± 50	32,8476 ± 17
SPHINCS+SHAKE-128s-simple	2725 ± 2	35,776 ± 15	328,881 ± 17
SPHINCS+SHAKE-192f-simple	2905 ± 3	35,290 ± 5	328,593 ± 19
SPHINCS+SHAKE-192s-simple	2747 ± 2	35,067 ± 4	328,475 ± 17
SPHINCS+SHAKE-256f-simple	2879 ± 2	35,492 ± 9	328,934 ± 17
SPHINCS+SHAKE-256s-simple	2810 ± 2	44,699 ± 48	329,375 ± 37

5.2. Comparative Tests

This section presents a comparison of measured post-quantum algorithms with the well-known RSA algorithm, the cornerstone of current cryptographic systems but endangered by the vision of quantum computers. This comparison aims to evaluate the performance and usability of post-quantum cryptographic solutions in the context of real-world applications. It investigates how post-quantum algorithms compare to the popular asymmetric cipher in modern applications. One of the main points of this analysis will be an attempt to answer a pivotal question: is the time overhead of post-quantum algorithms justified to achieve quantum resilience? This question is of paramount importance as the transition to post-quantum cryptography might become necessary. To obtain results relevant for comparison, an RSA implementation originating from the Python library 'cryptography' was used. The size of the used key was 2048 bytes, and the exponent was set at 65537. The measurements were performed in the same environment as the post-quantum algorithm measurements.

Figures 11–13 present a comparison of the average file signing times of the post-quantum and RSA algorithms. Due to the complexity of post-quantum algorithms, a time overhead is expected compared to RSA. Examples with 100 MB and 1 GB file sizes meet this hypothesis, while an example with a 10 MB file size shows that a post-quantum algorithm can be even a bit faster than RSA. Looking at the full context of these three scenarios, it can be observed that differences are not overwhelming. The question then remains, is the significant security improvement worth the time overhead? Dilithium algorithms, including the best one—labelled Dilithium5, appear to be suitable candidates for replacing RSA. It has the NIST security level 5 declared by OpenQuantumSafe, while the tested version of RSA is estimated to be level 1. In the conducted measurements on the largest file (1 GB), Dilithium5 achieved an average result of 3.31 s, while RSA scored 2.56 s. This represents an overhead of approximately 27.7% in exchange for resistance to quantum cryptanalysis.

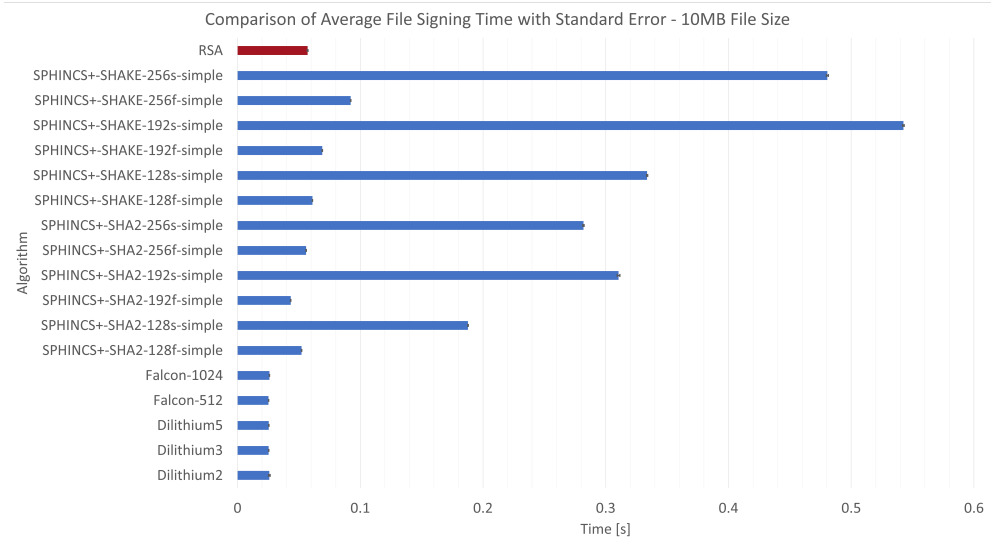

Figure 11. Comparison of average file signing time—10 MB file size.

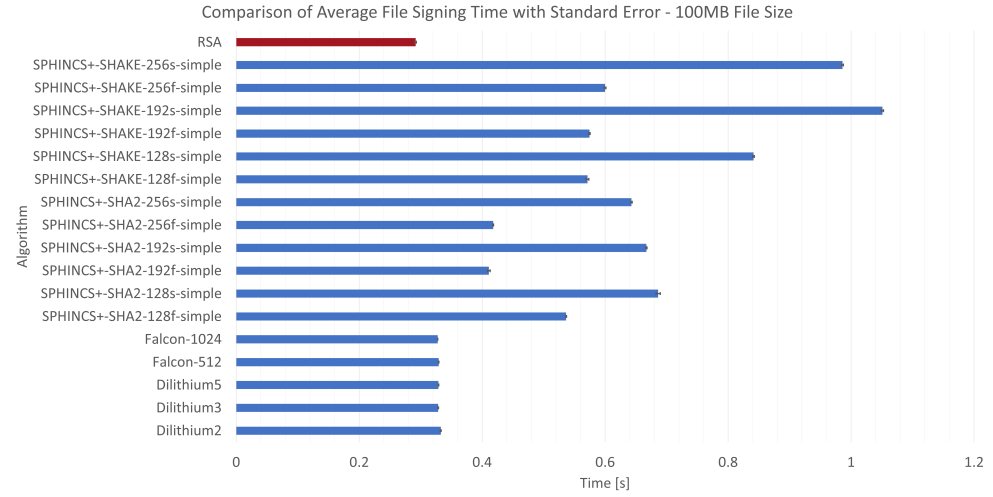

Figure 12. Comparison of average file signing time—100 MB file size.

Figure 13. Comparison of average file signing time—1 GB file size.

6. Conclusions

This paper emphasizes the fundamental role of post-quantum cryptography in enhancing digital signatures against the emerging threats associated with the potential of quantum computers. Through the implementation and testing of selected algorithms, the ability of post-quantum algorithms in application to digital signatures has been demonstrated. They are undoubtedly an alternative to the cryptographic solutions currently in use and a promising candidate for their replacement.

To conclude on the practical applications of post-quantum algorithms for digital signatures, it is necessary to state that the implementations have been positively verified in several usage scenarios. The results are consistent with the benchmarking of the liboqs library [19]. In the analysis of time performance, the Dilithium algorithm (especially Dilithium5) proved to be the best candidate for digital signatures. Although it performed slightly behind some competitors during signature verification, it clearly dominated during key pair generation and signing processes. An additional fact in its favour is the size of the generated signatures. All three available variants of the Dilithium algorithm generate a signature several times smaller than the SPHINCS+ group algorithms. But if the size of the generated signature is the primary criterion, the Falcon algorithm appears to be the favourite, achieving the smallest signatures. Representing a hash function approach, the algorithms in the SPHINCS+ group show great diversity due to their ability to cooperate with different hash functions. When deciding between implementations from this particular group, algorithms optimized for signature size would be a better choice. This is because, as the file size increases, the disparity between the two variants of the SPHINCS+ algorithm decreases. In the final analysis, it can be concluded that there are algorithms that can replace RSA with marginal overhead in time, while diametrically increasing security, whereas the choice of a particular algorithm may be determined by the needs of a particular case.

Research on the evaluation and standardization of post-quantum algorithms is constantly in progress, with NIST leading the way. Therefore, the optimization of post-quantum solutions can be expected, as well as the emergence of new solutions. On 17 July 2023, NIST announced additional candidates for digital signatures in the PQC standardization process. The list can be found on the institution's website, and more information can be expected in 2024, when the fifth PQC standardization conference will be held [23]. As NIST is continually engaged in standardization projects, future work may involve testing new algorithms suggested by the institution.

Author Contributions: Conceptualization, F.O. and M.N.; methodology, F.O. and M.N.; software, F.O.; validation, F.O.; formal analysis, F.O. and M.N.; investigation, F.O., M.N. and M.A.K.; writing—original draft preparation, F.O. and M.N.; writing—review and editing, F.O., M.N., M.G. and M.A.K.; visualization, F.O.; supervision, M.N., M.G. and M.A.K.; project administration, M.N.; funding acquisition, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research project was partly supported by the program "Excellence initiative—research university" for the AGH University of Krakow. This work was also supported by the EU Horizon Europe Framework Program under Grant Agreement no. 101119547 (PQ-REACT).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. PQ-REACT Website. Available online: https://pqreact.eu/ (accessed on 15 May 2024).
2. Hegde, S.B.; Jamuar, A.; Kulkarni, R. Post Quantum Implications on Private and Public Key Cryptography. In Proceedings of the 2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), Tumakuru, India, 7–8 July 2023; pp. 1–6. https://doi.org/10.1109/ICSSES58299.2023.10199503.

3. Lopez-Valdivieso, J.; Cumplido, R. Design and implementation of hardware-software architecture based on hashes for SPHINCS$^+$. *Acm Trans. Reconfigurable Technol. Syst.* **2024** . https://doi.org/10.1145/3653459.
4. Di Matteo, S.; Gerfo, M.L.; Saponara, S. VLSI Design and FPGA Implementation of an NTT Hardware Accelerator for Homomorphic SEAL-Embedded Library. *IEEE Access* **2023**, *11*, 72498–72508. https://doi.org/10.1109/ACCESS.2023.3295245.
5. Zhou, Z.; He, D.; Liu, Z.; Luo, M.; Choo, K.K.R. A Software/Hardware Co-Design of Crystals-Dilithium Signature Scheme. *ACM Trans. Reconfigurable Technol. Syst.* **2021**, *14*, 1–21. https://doi.org/10.1145/3447812.
6. Beckwith, L.; Nguyen, D.T.; Gaj, K. Hardware Accelerators for Digital Signature Algorithms Dilithium and FALCON. *IEEE Des. Test* **2023**, *99*, 1. https://doi.org/10.1109/MDAT.2023.3305156.
7. Mehic, M.; Niemiec, M.; Rass, S.; Ma, J.; Peev, M.; Aguado, A.; Martin, V.; Schauer, S.; Poppe, A.; Pacher, C.; et al. Quantum Key Distribution: A Networking Perspective. *ACM Comput. Surv.* **2020**, *53*, 1–41. https://doi.org/10.1145/3402192.
8. Bertels, K.; Sarkar, A.; Hubregtsen, T.; Serrao, M.; Mouedenne, A.A.; Yadav, A.; Krol, A.; Ashraf, I.; Almudever, C.G. Quantum Computer Architecture Toward Full-Stack Quantum Accelerators. *IEEE Trans. Quantum Eng.* **2020**, *1*, 1–17. https://doi.org/10.1109/TQE.2020.2981074.
9. Shor, P. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134. https://doi.org/10.1109/SFCS.1994.365700.
10. Hlukhov, V. Quantum-Inspired Computing: Shor's Algorithm and Euler's Totient Function. In Proceedings of the 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Dortmund, Germany, 7–9 September 2023; Volume 1, pp. 865–869. https://doi.org/10.1109/IDAACS58523.2023.10348718.
11. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 22–24 May 1996; STOC '96; pp. 212–219. https://doi.org/10.1145/237814.237866.
12. Shrivastava, P.; Soni, K.K.; Rasool, A. Evolution of Quantum Computing Based on Grover's Search Algorithm. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019; pp. 1–6. https://doi.org/10.1109/ICCCNT45670.2019.8944676.
13. Wong, D. *Real-World Cryptography*; Manning Publications Co.: Shelter Island, NY, USA, 2021.
14. Kuznetsov, A.; Kiian, A.; Pushkar'ov, A.; Mialkovskyi, D.; Smirnov, O.; Kuznetsova, T. Code-Based Schemes for Post-Quantum Digital Signatures. In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; Volume 2, pp. 707–712. https://doi.org/10.1109/IDAACS.2019.8924271.
15. Peng, C.; Chen, J.; Zeadally, S.; He, D. Isogeny-Based Cryptography: A Promising Post-Quantum Technique. *IT Prof.* **2019**, *21*, 27–32. https://doi.org/10.1109/MITP.2019.2943136.
16. PQ-CRYSTALS. Available online: https://pq-crystals.org/ (accessed on 15 May 2024).
17. Pornin, T. New Efficient Constant-Time Implementations of Falcon. Available online: https://falcon-sign.info (accessed on 15 May 2024).
18. Mohan, P.V.A. Hash-based Digital Signatures—A tutorial review. In Proceedings of the 2023 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA), Bangalore, India, 8–9 September 2023; pp. 1–8. https://doi.org/10.1109/PKIA58446.2023.10262777.
19. Open Source Liboqs Library. Available online: https://openquantumsafe.org/liboqs (accessed on 15 May 2024).
20. Stebila, D.; Mosca, M. Post-quantum key exchange for the Internet and the Open Quantum Safe project. In Proceedings of the Selected Areas in Cryptography (SAC) 2016, St. John's, NL, Canada, 10–12 August 2017; Lecture Notes in Computer Science; Volume 10532, pp. 1–24.
21. NIST Post-Quantum Cryptography. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography (accessed on 15 May 2024).
22. National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) 140-2, Security Requirements for Cryptographic Modules (with Change Notice 2), 2001. Available online: https://csrc.nist.gov/publications/detail/fips/140/2/final (accessed on 15 May 2024).
23. National Institute of Standards and Technology. Additional PQC Digital Signature Candidates Announced. 2023. Available online: https://csrc.nist.gov/news/2023/additional-pqc-digital-signature-candidates (accessed on 15 May 2024).