Applying a cryptographic metric to post-quantum lattice-based signature algorithms

Markus Rautell
markus.rautell@vtt.fi
VTT Technical Research Centre of Finland
Oulu, Finland

Outi-Marja Latvala
outi-marja.latvala@vtt.fi
VTT Technical Research Centre of Finland
Oulu, Finland

Visa Vallivaara
visa.vallivaara@vtt.fi
VTT Technical Research Centre of Finland
Oulu, Finland

Kimmo Halunen
University of Oulu and National Defence University of Finland
Oulu, Finland
kimmo.halunen@oulu.fi

ABSTRACT

Measuring the security of cryptographic systems is not a simple task. Nevertheless, there is an increasing need for a cryptographic metric which could assist in decision making when choosing between various candidates. The National Institute of Standards and Technology (NIST) has launched a process to standardize quantum-resistance public key encryption, key encapsulation and digital signature algorithms. This is NIST's response to the threat posed by quantum computers against classical public key cryptography. In this paper, we apply a metric taxonomy, produced by earlier studies, to two NIST third round finalist digital signature algorithms Dilithium and Falcon in order to asses the effectiveness and extensiveness of the metric. Although, our results show that clear differences can be found with used metrics, we propose some improvements to them to allow more comprehensive analysis.

CCS CONCEPTS

• **General and reference** → **Metrics**; • **Security and privacy** → *Cryptography*; • **Computer systems organization** → Quantum computing.

KEYWORDS

Metrics, Signatures, Lattice Encryption, Post-Quantum Cryptography

ACM Reference Format:
Markus Rautell, Outi-Marja Latvala, Visa Vallivaara, and Kimmo Halunen. 2022. Applying a cryptographic metric to post-quantum lattice-based signature algorithms. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3538969.3544438

1 INTRODUCTION

Measuring the security of systems in general and cryptographic systems in particular is an interesting and difficult problem. In security there are many metrics that can be applied and based on the system and the user the preference for these may vary greatly. Although progress has been made and measures exist, there still is not a one size fits all metric for security.

In cryptography and with cryptographic systems the situation is somewhat similar, but there exist measures that are used to compare different cryptographic primitives and systems. The most simple metric is the *key length* of the system, which tells how many (usually uniformly random) bits is in the key used to perform cryptographic operations. This allows for comparison between different systems, but it does not really grasp all nuances that affect the security of a cryptographic system.

Comparing the key lengths of symmetric cryptographic systems is straightforward - the more bits you have, the greater the security of the system. However, already with currently used public key cryptosystems the comparison is not that simple. A 1024-bit RSA cipher is much less secure against best known attacks than 128 AES despite the longer key. This is due to the fact that there exist fairly efficient factoring algorithms (e.g. General Number Field Sieve (GNFS) [34]) that are superpolynomial in complexity, but still much more efficient in breaking the cipher than a brute force key search. The relation between symmetric key sizes and asymmetric key sizes can still be computed by using the complexity of the GNFS as a measure to "normalize" the RSA modulus size with the symmetric key length. For elliptic curves there exist similar results that allow for good comparison.

When taking into consideration the new post-quantum cryptography algorithms, the comparison between different key lengths loses much of its meaning. The different candidates in the NIST standardization competition have great differences in their key sizes and just by comparing key lengths one does not get meaningful information about their security. However, NIST has specified five levels of security [1] (by a comparison with symmetric primitives) that different proposals should aim for. The proposals then give different parameter sets for ciphers and claim that these achieve certain levels. Yet even these levels are not very conclusive and there is some debate around different proposals and their parameters that achieve certain levels. Thus, there is a need for more nuanced metrics for cryptosystems.

In [23] the authors present a taxonomy of metrics for evaluating the security of cryptographic systems. This has been then applied to

ARES 2022, August 23–26, 2022, Vienna, Austria

ACM ISBN 978-1-4503-9670-7/22/08…$15.00
https://doi.org/10.1145/3538969.3544438

different versions of TLS protocol in [33] and further refined. These works represent a very different approach to the measurement of the security of cryptographic systems with several different variables measured in different categories.

The purpose of this paper is to further apply the metrics from [33] to lattice-based signature algorithms in the final round of the PQC standardization competition. Our goal is to find out, if these metrics can be used to find relevant differences between the candidates.

The paper is organised as follows. In Section 2 we cover the background of our work. Section 3 consists of comparison on digital signature algorithms Dilithium and Falcon with a metric taxonomy from [33]. After that, Section 4 discusses our findings and future improvements. Section 5 concludes our paper.

2 BACKGROUND

In this section, we present the motivation behind this research, cover relevant lattice problems and introduce NIST's finalist signature algorithms under our evaluation.

2.1 Measuring Security

Measuring the current state of a system is a prerequisite for improving it. Security metrics come in many forms, from high-level abstract metrics of larger systems to specific, individual counts of, e.g., malware found on one host computer. A review on security metrics can be found in Section 6 of [53], where the research on security metrics is divided into four categories: the nature of security metrics, measuring the security of a computer system, managing IT security risks and measuring the effectiveness of a security process.

Based on an expert opinion survey [49], the key quality aspects of security metrics are correctness, measurability and meaningfulness. Usability is also given a special mention. Correctness of a metric encompasses such concepts as accuracy and precision. It is important to pick the right metrics for a given context and at a correct granularity and abstraction level to aid you in making the right security decisions for your system. Measurability, and also usability, are needed for the practical application of metrics. Measurable information needs to be attainable, and to be precise you need to be able to reproduce or repeat measurements. Meaningfulness is important for the people using the metrics. Clarity and succinctness when formulating a metric make it more useful and understandable. Therefore, the metric is easier to include in the decision making process and more efficient to use.

Cryptographic metrics have previously been quite scattered. The most well known measure of cryptographic security is undoubtedly the key length of an algorithm. There are several governmental and standardization bodies that give advice on how to choose appropriate values for given targets [21]. The key length metric focuses on the algorithms resilience against simple brute force attacks (i.e. trying to guess the key), which leaves out problems with protocols or other implementation level issues (see, e.g., Heartbleed [12]) and niche use cases for otherwise sound algorithms [9].

The purpose of a metric is to help you choose between all the different cryptosystems when buying or designing a new product or service. For example, maybe a company wants to buy secure "tough phones" for their employees and needs a way to compare the options. Often the candidate cryptosystems have the same underlying theoretical proofs and parameters, and commercial products may have undergone certification testing, verifying that it fulfills the certificate-specific security criteria. Therefore, we need to consider the performance of different cryptosystems as well, in order to properly compare different products.

2.2 Post-Quantum Cryptography and Standardization

Current public key systems are based on old mathematical problems: factoring, discrete logarithm in finite fields, and discrete logarithm in elliptic curves. Shor's algorithm [50] on a suitable quantum computer will break all of the current favourites: RSA, ECDSA, (EC)DH and DSA. Symmetric key cryptography standards would also be affected by Grover's quantum search algorithm [22], but less dramatically. Grover's algorithm provides a quadratic speedup over classical search algorithms for finding the symmetric key.

National Institute of Standardization and Technology (NIST) has started standardizing post-quantum cryptography [3]. The standardization started in January 2017 and will be finished around 2022-2023. The goal is to find suitable methods for digital signatures and key exchange, which are the two major use cases for public key cryptography.

Post-quantum signature algorithms are based on different mathematical problems which have much larger keys and signatures than current algorithms. There aren't many good quantum-proof signature candidates to choose from and currently only lattice-based signatures have reasonable performance to sign and verify messages. Even though their public keys and signatures are bigger than for RSA or ECDSA, they are not too impractical either.

2.3 Basics of lattice problems

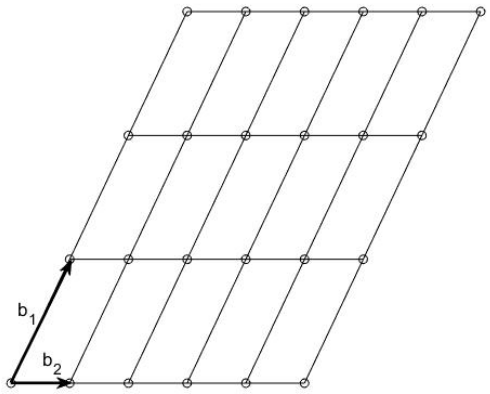

Figure 1: 2-dimensional lattice generated by linearly independent vectors b_1 and b_2.

In this section we briefly present the basics of lattice problems utilized with Dilithium [13] and Falcon [17]. For further reading on lattice-based post-quantum schemes and their mathematical basis see [40].

First we define a lattice to be a set of points in n-dimensional space generated by the set of all integer combinations

$$\mathcal{L}(b_1, ..., b_n) = \{\Sigma_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \text{ for } 1 \leq i \leq n\} \quad (1)$$

of n linearly independent vectors as demonstrated in Figure 1. Given matrix $A \in \mathbb{Z}_q^{m \times n}$ for some integers m, n and a prime number q, we define m-dimensional q-ary lattice,

$$\Lambda_q(A) = \{y \in \mathbb{Z}^m \mid y = As \bmod q \text{ for some } s \in \mathbb{Z}^n\}. \quad (2)$$

In the following we define matrix A to be a set of vectors $a_1, ..., a_m \in \mathbb{Z}_q^n$ sampled from uniform distribution.

Learning With Errors (LWE) problem, introduced by Regev in 2005 [46], consists of finding a secret vector s such that $x = As + e$ where $x, e \in \mathbb{Z}_q^m$ and e is an error vector specified by some probability distribution which adds noise to the equation. The hardness of this problem is based on indistinguishability between pairs (A, x) and (A, u), where u is uniformly random, without knowledge of vectors s and e.

Short Integer Solution (SIS) problem, introduced by Ajtai in 1996 [2], consists of finding a short nontrivial vector $z \in \mathbb{Z}_q^m$ for matrix A such that $Az = 0 \in \mathbb{Z}_q^n$. SIS hardness can be illustrated by collision search $Ax = Ay \in \mathbb{Z}_q^n$ where $x, y \in \mathbb{Z}^m$ are vectors close to each other. From the collision condition we get that $Az = Ax - Ay = 0 \in \mathbb{Z}_q^m$ and it follows that $z = x - y \in \mathbb{Z}^m$ which has to be short since x and y are close.

M-LWE and M-SIS are variants of LWE and SIS that utilize module lattices and polynomial rings such as $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ where q is the modulus for integer coefficients and n is the dimension of the ring R [32]. SelfTargetMSIS is a problem which is based on the combined hardness of M-SIS and a cryptographic hash function H [51]. Ducas et al. [13] and Kiltz et al. [30] provide more details on M-LWE, M-SIS and SelfTargetMSIS.

2.4 Falcon

Falcon is a cryptographic digital signature algorithm and stands for Fast Fourier Lattice-based Compact signatures Over NTRU. It is based on the framework described by Gentry, Peikert and Vaikuntanathan (GPV) [20] constructing hash-and-sign lattice-based signature scheme over NTRU lattices [25] and uses Fast Fourier sampling [15] as a trapdoor sampler. Security of this scheme is grounded on the GPV framework under SIS assumption.

Falcon is designed with the intention to minimize the combined size of public key and signature, and it has the smallest bandwidth of all the second round digital signature schemes in the NIST standardization process [3]. Compactness in mind, the Authors have chosen to take advantage of NTRU lattices to obtain compact instantiation of GPV framework [14] and utilize sampling from Gaussian distribution. Computations are done in a modular ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where $q = 12289$ and $n = 512$ or $n = 1024$ for security levels 1 and 5 defined by NIST. Falcon has non-deterministic key and signature generation modules, although signature verification is deterministic.

According to the authors, Falcon has some clear limitations. Firstly, key generation and fast Fourier sampling are not trivial to understand or simple to implement. There also appears to be challenges with the use of floating-point arithmetic and it might become a major limitation when implementing Falcon on constrained devices. Authors previously listed unclear side-channel resistance as limitations [17], due to older version of Gaussian sampler. The current version, Isochronous Gaussian sampler, runs in constant-time and seems to be safe to use. The work of [18] shows that isochrony is an important requirement for the embedded security of this scheme. The current Gaussian sampler is based on [26, 54].

2.5 Dilithium

Dilithium is a lattice-based signature scheme and a part of the CRYSTALS package that was submitted for the NIST standardization process. Its design relies on Fiat-Shamir with Aborts framework [36, 37] and can be seen as improved variant of the Bai-Galbraith scheme [5]. Dilithium's security is based on the hardness of M-LWE and M-SIS lattice problems.

Dilithium uses the same ring $R_q = \mathbb{Z}_q[x]/(x^{256} + 1)$ with fixed $q = 2^{23} + 2^{13} + 1$ for all its three security levels 2, 3 and 5 by NIST criteria. Different security levels are achieved by changing the dimension of the matrix A. Due to this, it is possible to optimize the performance for all security levels by just optimizing operations in R_q. Dilithium is featured with the possibility to choose between deterministic and randomised signature [4]. For the second round of NIST standardization process Dilithium was introduced with version that replaces hash-function SHAKE with AES, which is used to expand matrices and vectors from seeds.

Similar to Falcon, the motivation has been in minimizing the sum of public key and signature sizes. However, instead of Gaussian sampling the authors have chosen to use uniform sampling which has a negative effect considering the compactness. Dilithium also utilizes rejection sampling to make sure that signature is verifiable and it does not leak information about secret key.

3 ANALYZING FALCON AND DILITHIUM WITH A CRYPTOGRAPHIC METRIC

Table 1 presents the analysis of Falcon and Dilithium using the taxonomy from [33]. The metric is divided into three main categories, namely adversarial model (AM), proof framework (PF) and verification and feasibility (VF), which are then divided into subcategories. The metrics in the first two categories focus on the theoretical foundations of the system under investigation, while the last category contains metrics about the deployment and implementation processes.

Table 1 is filled with results from various studies based on round 1 and round 2 submissions of the NIST standardization process and values taken straight from Falcon's and Dilithium's round 3 submission packages. In other words, when we refer to other studies results are based on the earlier states of algorithms and are rather suggestive, and while referring to the submission documentations results describe the current state of these algorithms [13, 17]. Exceptions to these two statements are explicitly mentioned.

3.1 Adversarial model

For analyzing the adversarial metrics we can use the original call text from NIST [1]. Because this starting point is the same for all contenders, the results of these metrics are the same for Falcon and Dilithium.

Table 1: The results of comparing Falcon and Dilithium. Metrics that were not applicable in this study are marked with N/A in the table.

Cat.	Subcategory	Metric	Falcon	Dilithium
AM	Degrees of freedom	Observe, choose, choose adaptively	choose adaptively	choose adaptively
		Corruption power - number of principals	N/A	N/A
		Corruption power - degree of corruption	N/A	N/A
		Security game compliance	Game, generic	Game, generic
	Adversarial available information	Pre-crypto	Public key	Public key
		Post-crypto	2^{64} signatures of chosen messages	2^{64} signatures of chosen messages
		Secret key material	no material	no material
		Protocol runs	N/A	N/A
		Setup parameters	N/A	N/A
		Simulation environment	N/A	N/A
	Adversarial goal	Goal	Semantic deduction, forgery	Semantic deduction, forgery
	Adversarial resources	Computation power, instantiated	quantum computer	quantum computer
		Computation power, non-instantiated	BQP	BQP
		Memory, instantiated	quantum computer	quantum computer
		Memory, non-instantiated	unlimited	unlimited
PF	Security assumptions	Mathematical complexity	computational / search	computational / search
	Abstraction assumptions	Type	ROM, QROM	ROM, QROM
		Number of assumptions	N/A	N/A
		Maturity of assumptions	N/A	N/A
		Tightness	tight reduction	non-tight reduction
		Rigor	N/A	N/A
VF	Assurance levels	Assurance standard or profile	N/A	N/A
		Evaluation assurance level	N/A	N/A
	Test coverage	Percentage of covered areas	N/A	N/A
	Vulnerabilities	Number	1	1
		Number of classified	1	1
	Side-channels	Existence	known side-channel attacks	known side-channel attacks
	Metadata	Leaked amount and type	N/A	N/A
	Evaluator acceptance & reputation	Reviews	12	19
	Evaluator experience	Academic publications	1873	3763
		Experience in years	348	696
	Verification time	Time since released for evaluation	Released 30.11.2017	Released 30.11.2017
		Size and efforts of evaluation community	Cryptography II, Authentication	Cryptography II, Authentication
	Openness of target	Software/design	open source	open source
	Readiness level	Technology Readiness Level	TRL 4 - TRL 6	TRL 4 - TRL 6
		Integration Readiness Level	IRL 4 - IRL 6	IRL 4 - IRL 6
		System Readiness Level	SRL 2 - SRL 3	SRL 2 - SRL 3
		PETS maturity model	Proof-of-concept$^{(+/++)}$	Proof-of-concept$^{(+/++)}$
	Key length	Bits for criteria compliance	1 998 - 3 840 (B)	2 032 - 3 920 (B)
	Time costs	Execution overhead	See Table 2	See Table 2
		Communication overhead	N/A	N/A
	Memory and transmission costs	Run-time memory	4 300, 2 708 (kB)	5 524, 3 048 (kB)
		Storage capacity	3 561 - 6 913 (B)	5 764 - 11 107 (B)
		Communication bandwidth	1 536 - 3 073 (B)	3 732 - 7 187 (B)
	Implementation complexity	Size of software	356 532 - 1 112 295 (B)	165 478 - 281 698 (B)
		Dedicated hardware requirements	no special hardware needed	no special hardware needed
	Energy efficiency	Algorithm complexity dependent joules	N/A	N/A
		Hardware platform dependent joules	N/A	N/A

In Section 4.A.4 of the call text [1] a requirement that the signature schemes must be EUF-CMA secure is presented. Therefore, the adversary is able to adaptively choose messages to sign. We can also say that the adversary is following this generic security game.

The corruption power metrics, on the other hand, are more important when investigating interactive protocols or even bigger systems, where the adversary might be able to access or modify some private information but not others. Similarly, the protocol runs, setup parameters and simulation environment metrics are not applicable here.

In the EUF-CMA schema, the adversary is given the public key generated by the challenger, and the private key is kept secret. The call text [1] also states that the contenders should assume that the attacker has access to 2^{64} signatures of chosen messages. Moreover, NIST advises the contenders to consider attacks that use classical (rather than quantum) queries to a signing oracle.

Table 2: CPU cycles used for each operation

Operation	Algorithm	NIST security level			
		1	2	3	5
Key Gen	Dilithium	-	300 751	544 232	819 475
	Falcon	19 872 000	-	-	63 135 000
Sign	Dilithium	-	1 081 174	1 713 783	2 383 399
		-	1 355 434	2 348 703	2 856 803
	Falcon	386 678	-	-	789 564
Verify	Dilithium	-	327 362	522 267	871 609
	Falcon	82 340	-	-	168 498

Finally, the computational resources of the adversary are the crux of the whole standardization process. The adversary is assumed to have access to a quantum computer. In a non-instantiated computation power estimate we have the bounded-error quantum polynomial-time (BQP) as a limit. Quantum information processing is under a lot of active research [52][16], so it is unclear what the limits are; therefore we choose to not limit it for the non-instantiated metric.

3.2 Proof framework

We analyze theoretical security and hardness of problems that Falcon's and Dilithium's security assumptions are build on. In this section our results and speculations are based on NIST third round submission documentations of examined algorithms.

Used metric taxonomy founds similarities between Falcon and Dilithium concerning mathematical complexity and abstraction assumption types. Same output for mathematical complexity results from the fact that they are both lattice-based signature schemes and due to that their security lays on hardness of finding short non-zero vectors from lattices. On the other hand, we can see from sections 2.5 and 2.4 that security of these algorithms are build on different lattice problems. The used metrics are simply not efficient enough to recognize difference at this level. In case of abstraction assumption types, both algorithms utilize Random Oracle Model (ROM) [8] and Quantum Random Oracle Model (QROM) [10] in their security proofs.

Divergence is found from the tightness of security proofs. Falcon has security proofs in both types with tight reductions [3]. In the case of Dilithium, there is an approach to get a non-tight reduction using forking lemma [7, 42] from M-SIS to SelfTargetMSIS in the ROM. The QROM case is a bit unclear, but there seems to be an approach to prove the security with non-tight reduction [13]. From the security perspective, used metric taxonomy indicates that Falcon is the safer one since there seems to be less uncertainty involved.

3.3 Verification and feasibility

In this section taxonomy of metrics measures strength, correctness, maturity and feasibility of Falcon and Dilithium. First we briefly go through all similarities that appeared as a result of the measurement. After that we focus on analyzing the differences found by used metrics between algorithms and draw some observations from them.

Discovered similarities are related to vulnerabilities, side-channels, verification time, readiness level, openness of target and hardware requirements. Even though the categories for vulnerabilities and side-channels gave same outcome, they relate to different aspects of examined algorithms and they will be covered at the end of this section.

Identical results in verification time is due to the identical submission dates of algorithms and similarity in algorithm type. Submissions of both algorithms are submitted to NIST on the deadline date as proposed in [1]. Since both examined algorithms are lattice-based digital signature algorithms, they naturally are categorized in the same evaluation community. We have used the definition in [29] for the size of evaluation community and come to the conclusion that signature schemes can be classified into either one of Cryptography II or Authentication communities.

Due to arguments given above, it is quite natural that readiness level gives same results between examined algorithms in every subcategory. The criteria can be found for TRL [38], IRL [47], SRL [48] and PETS [24]. Results are given in level intervals, because these tools are not specific to the field of these algorithms. Due to this it is not possible to point out one certain level, which would be absolutely correct.

Used metric gives same output considering openness of software and design of both algorithms. Submission packages sent to NIST are made public which enables comprehensive evaluation and research on algorithms by the whole scientific community. This aspect has helped authors of Falcon and Dilithium to improve and fix some shortcomings of their algorithms. Neither Falcon nor Dilithium requires special hardware. They are designed to be able to run on ordinary classical computers.

Differences between examined algorithms occurred in categories evaluator acceptance and reputation, evaluator experience, key length, execution overhead, memory and transmission costs, storage capacity and implementation complexity.

Key length is the length of the private key which varies between different security levels. For Falcon the private key is defined to be about three times the size of the signature. This information is available on Falcon's website [1]. In case of Dilithium, there is a formula in the third round submission documentation that defines the length of private keys for each security level. Both, Falcon and Dilithium, have announced that their private keys can be compressed to a 32 byte seed ρ, but this requires signer to run the key generation

[1] https://falcon-sign.info/

before signing. For post-quantum schemes the length of private key does not equal the amount of security, as mentioned earlier. Thus, straightforward comparison is not possible between key lengths only. This value mainly affects the storage capacity of the signature schemes.

Table 2 shows Falcon's and Dilithium's performance in CPU cycles. Values for Dilithium are taken from the round 3 submission package as they are presented there. Falcon's performance is expressed in form of milliseconds or operations per second in the round 3 submission documentation. We have converted these values into CPU cycles by the reported clock frequency 2.3 GHz. All values in Table 2 are based on reference implementations. Given values of Dilithium are medians of 1000 executions each. For signing operation the first row of Dilithium is the median and second row the average number of cycles. This is due to the great difference between median and average values. The original values of Falcon's performance have been measured from sustained workloads over prolonged periods of time. When considering the security level 5 we can see that Falcon needs 77 times the cycles of Dilithium for key generation, for signing Dilithium takes 3 to 4 times and for verifying 5 times the cycles of Falcon. In the concept of digital signatures, there isn't always need to generate new keys when generating signature. So, for some use cases, we could ignore the disturbingly great difference in key generation CPU cycles. This would make Falcon the more efficient one, at least in case of reference implementations.

Memory and transmission costs consist of run-time memory, storage capacity and communication bandwidth. Results for run-time memory are given in [6] and they are based on the reference implementations. First value refers to how much memory, in its entirety, is allocated to the process and the second one shows how much is actually located in RAM. These values are useful for evaluating the viability of devices with more restricted memory such as embedded systems. The reader should be critical towards this source since there is no similar research that would support or differ from it. Results still back up the assumption that Falcon is the compact one of examined algorithms.

Communication bandwidth covers the sum of the signature and the public key sizes, because these are the objects that need to be sent over the communication channel. Storage capacity and communication bandwidth are based on values from the submission documentations of examined algorithms. Storage capacity is defined to be the sum of private key and communication bandwidth, and it describes how much storage memory the algorithm needs for objects which it generates. Here we have used the whole length of the private key, not the size of the seed ρ, because otherwise this value would be unnecessary close to the value presented for communication bandwidth. The range of numerical values in these two subcategories is due to variation on key sizes between different security levels. These results advocate the perception that Falcon defeats Dilithium in compactness.

For implementation complexity the size of the software is decided to be the size of C programming language implementation from the third round submission packages. Falcon's submission package contains reference implementation and optimized implementation, so we have taken both of these into account. For Falcon the numerical range of this subcategory is due to sizes differences between different security levels and implementations types. In case of Falcon-512 sizes of reference implementation is 356 532 bytes and size of optimized implementation is 1 112 136 bytes. By the same principle implementations sizes of Falcon-1024 are 356 585 and 1 112 295 bytes. Dilithium's submission package contains packages of reference implementations and additional implementations, which both include four different versions of each security level. The additional implementations contains the AVX2 optimized version of Dilithium. We have decided to use the "basic" versions in this study for both reference and additional implementations. The AES-versions are not included since Dilithium is originally designed to use SHAKE instead. Due to this decision Dilithium's values are more comparable to Falcon, which also uses SHAKE. There are only two values presented for Dilithium since the implementation sizes are the same for every security level. First value of presented range is the size of reference implementation and second is the size of optimized implementation. Here we see that Falcon is considerably larger, especially in the case of optimized implementations.

Evaluator acceptance and reputation is quantified by number of search results from Scopus. The main idea was to find articles and conference papers, which include the name of desired signature scheme in their title, abstract or keyword list. We had to make the search quite restricted, due to Falcon's commonly used name, to avoid invalid search results. For that reason, our sampling is small, and we see this metric only as a guideline for review distribution between these two signature schemes. The meaning of this study is not to be a review article of previously published studies and we want to emphasize that this is just one metric among many others.

Evaluator experience is defined by the sum of academic publications of all evaluators and experience in years, which is the sum of years that evaluators have participated in academic research. In this study, evaluator is defined to be a person that has been involved in at least one search result, specified above, and hasn't been named as an author for signature scheme under evaluation. These two paragraphs indicate that Falcon is less studied than Dilithium. Even though in section 3.2 we claimed that there was less uncertainty considering Falcon, metrics of this section indicate that there could be some hidden vulnerabilities that we are just not aware of. This is not necessarily the case, but it seems more likely than with Dilithium.

Vulnerabilities of examined algorithms are related to side-channel resistance. NIST stressed the importance of side-channel resistance in [3] so major part of studies cited in this paragraph are very recently published and at this point their validity is hard to assess. Important to this study is that there certainly seems to be side-channel vulnerabilities, which is recognized by used taxonomy of metrics. We list studies considering attacks against Falcon [39] and Dilithium [11, 19, 27, 31, 43–45]. What the used metrics does not recognize is that a difference can be found from the algorithm implementation level. Side-channel vulnerabilities are related to Fast Fourirer Transform (FFT) [35] and Number Theoretic Transform (NTT), which is a version of FFT that works over the finite field $\mathbb{Z}_q$. Falcon utilizes FFT and Dilithium NTT during secret key related operations and they leak information about the secret key when exposed to a side-channel attack. However, there is evidence [28, 41] indicating that FFT has possibly a lower side-channel leakage compared to NTT. Let it be mentioned that Falcon also utilizes NTT

but only during public key related operations, so there is no secret key information leakage concerning the use of NTT. Thus the used metrics give a flawed idea that there are no differences considering the vulnerabilities of examined algorithms. More importantly, it does not recognize the degree of vulnerability.

4 DISCUSSION

There are a few aspects that caused challenges during our evaluation process. We consider first the ones that generally made the comparison more complicated and second we focus on the ones that affected the usability of the metric taxonomy.

Both signature schemes have made changes to their security parameters along the way of the NIST standardization process. Studies done before round 3 submission have lost some of their relevance since the parameter changes affect the level of security and performance of the examined algorithms. Also the fact that algorithms aim at different levels of security has a negative effect on comparability. Let it be also mentioned, that NIST's demands and the lattice-based nature bring these algorithms close to each other in many metrics.

Considering the taxonomy of metrics from [33] it is important to note that it aims to be generic to any cryptographic system. In this study examined algorithms are still under development and standardization, so the used tool is not perfectly suitable. We also noticed that rigidity of metrics at lower subcategories makes it unnecessarily difficult to find suitable data. These facts are causing the blank spots in our metric Table 1. It is also notable that there is a lot of data in the submission documentations and plenty of studies considering these algorithms that just did not quit fit to used metric model and were therefore left out.

Pursuant to the above, we think the metrics could be improved by decreasing the rigidity of lower subcategories without bringing down their level of relevance. Also it would be favourable to deepen the level of analysis for some subcategories to achieve more comprehensive comparability. In order to maintain general usability, one solution could be to develop at least partially separated metrics to key encapsulation and digital signature algorithms.

Finally, based on Section 3.2 the used metrics indicate that Falcon is more safe from theoretical security perspective. In case of Section 3.3, the taxonomy of metrics suggests that Falcon is more compact with the exception of implementation complexity. On the other hand, Falcon is more likely to have hidden vulnerabilities due to the evaluator acceptance, reputation and experience. The efficiency perspective depends mainly on the purpose of use. If keys need to be generated often, them Dilithium is more efficient to that application and if not then Falcon is preferable.

5 CONCLUSIONS

In this paper, we have compared two post-quantum signature algorithms, Falcon and Dilithium, using a taxonomy of cryptographic metrics from [33]. We conclude that clear differences can be found. However, we think that the metric should be improved so that it would allow for a more comprehensive analysis of this type of algorithms.

As future work, we propose that more of the different PQC candidates should be evaluated with respect to these metrics. It would be preferable to compare several candidate algorithms concurrently to get better knowledge about differences that the used metrics can and cannot detect. For the same reason, it would be interesting to compare candidates based on different mathematical problems. As mentioned earlier, there is need to improve the used metrics and solve the difficult trade off between simplicity and coverage.

ACKNOWLEDGMENTS

This work has been conducted in the Post Quantum Cryptograpy Finland project, which has received funding from Business Finland.

REFERENCES

[1] 2016. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals

[2] Miklós Ajtai. 1996. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 99–108.

[3] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. 2020. Status report on the second round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST* (2020).

[4] S Bai, L Ducas, E Kiltz, T Lepoint, V Lyubashevsky, P Schwabe, G Seiler, and D Stehlé. 2021. Crystals-dilithium algorithm specifications and supporting documentation (version 3.1). *NIST Post-Quantum Cryptography Standardization Round* 3 (2021).

[5] Shi Bai and Steven D Galbraith. 2014. An improved compression technique for signatures based on learning with errors. In *Cryptographers' Track at the RSA Conference*. Springer, 28–47.

[6] Kevin Baptista. 2020. Performance Evaluation of Round 2 Submissions for the NIST Post-Quantum Cryptography Project. *Performance Evaluation* 2020 (2020), 05–16.

[7] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*. 390–399.

[8] Mihir Bellare and Phillip Rogaway. 1993. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 62–73.

[9] Alex Biryukov and Dmitry Khovratovich. 2009. Related-key cryptanalysis of the full AES-192 and AES-256. In *International conference on the theory and application of cryptology and information security*. Springer, 1–18.

[10] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random oracles in a quantum world. In *International conference on the theory and application of cryptology and information security*. Springer, 41–69.

[11] Leon Groot Bruinderink and Peter Pessl. 2018. Differential fault attacks on deterministic lattice signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 21–43.

[12] Marco Carvalho, Jared DeMott, Richard Ford, and David A. Wheeler. 2014. Heartbleed 101. *IEEE Security Privacy* 12, 4 (2014), 63–67. https://doi.org/10.1109/MSP.2014.66

[13] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2021. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (Version 3.1). (2021). https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf Accessed: 2021-08-04.

[14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. 2014. Efficient identity-based encryption over NTRU lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 22–41.

[15] Léo Ducas and Thomas Prest. 2016. Fast fourier orthogonalization. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*. 191–198.

[16] Fulvio Flamini, Nicolò Spagnolo, and Fabio Sciarrino. 2018. Photonic quantum information processing: a review. *Reports on Progress in Physics* 82, 1 (nov 2018), 016001. https://doi.org/10.1088/1361-6633/aad5b2

[17] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. 2020. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. (2020). https://falcon-sign.info/falcon.pdf Accessed: 2021-08-04.

[18] Pierre-Alain Fouque, Paul Kirchner, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. 2020. Key Recovery from Gram–Schmidt Norm Leakage in Hash-and-Sign Signatures over NTRU Lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 34–63.

[19] Apostolos P Fournaris, Charis Dimopoulos, and Odysseas Koufopavlou. 2020. Profiling dilithium digital signature traces for correlation differential side channel attacks. In *International Conference on Embedded Computer Systems*. Springer, 281–294.

[20] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 197–206.

[21] Damien Giry. [n.d.]. BlueKrypt - Cryptographic Key Length Recommendation. https://www.keylength.com/ accessed: 2021-06-07.

[22] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.

[23] Kimmo Halunen, Jani Suomalainen, Outi-Marja Latvala, Markku Kylänpää, Visa Vallivaara, and Mikko Kiviharju. 2019. A Taxonomy of Metrics for Cryptographic Systems. In *Thirteenth International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2019*.

[24] M Hansen, JH Hoepman, M Jensen, and S Schiffner. 2015. Readiness Analysis for the Adoption and Evolution of Privacy Enhancing Technologies: Methodology, Pilot Assessment, and Continuity Plan. *Technical report: ENISA* (2015). https://www.enisa.europa.eu/publications/pets Accessed: 2022-06-16.

[25] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*. Springer, 267–288.

[26] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. 2020. Isochronous Gaussian sampling: from inception to implementation. In *International Conference on Post-Quantum Cryptography*. Springer, 53–71.

[27] Emre Karabulut, Erdem Alkim, and Aydin Aysu. [n.d.]. Single-Trace Side-Channel Attacks on ω-Small Polynomial Sampling. ([n. d.]).

[28] Emre Karabulut and Aydin Aysu. [n.d.]. Falcon Down: Breaking Falcon Post-Quantum Signature Scheme through Side-Channel Attacks. ([n. d.]).

[29] Sotirios Katsikeas, Pontus Johnson, Mathias Ekstedt, and Robert Lagerström. 2021. Research Communities in cyber security: A Comprehensive Literature Review. *arXiv preprint arXiv:2104.13196* (2021).

[30] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. 2018. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 552–586.

[31] Il-Ju Kim, Taeho Lee, Jaeseung Han, Bo-Yeon Sim, and Dong-Guk Han. 2020. Novel Single-Trace ML Profiling Attacks on NIST 3 Round candidate Dilithium. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1383.

[32] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography* 75, 3 (2015), 565–599.

[33] Outi-Marja Latvala, Jani Suomalainen, Kimmo Halunen, Markku Kylänpää, Reijo Savola, and Mikko Kiviharju. 2020. Applicability of a Cryptographic Metric Taxonomy in Cryptosystem Procurement Process and in Evaluation of Open Standards. *International Journal on Advances in Security* 13, 3&4 (2020), 121–135.

[34] Arjen K Lenstra, Hendrik W Lenstra, Mark S Manasse, and John M Pollard. 1993. The number field sieve. In *The development of the number field sieve*. Springer, 11–42.

[35] Patrick Longa and Michael Naehrig. 2016. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *International Conference on Cryptology and Network Security*. Springer, 124–139.

[36] Vadim Lyubashevsky. 2009. 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, 598–616.

[37] Vadim Lyubashevsky. 2012. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 738–755.

[38] John C Mankins et al. 1995. Technology readiness levels. *White Paper, April* 6, 1995 (1995), 1995.

[39] Sarah McCarthy, James Howe, Neil Smyth, Séamus Brannigan, and Máire O'Neill. 2019. BEARZ Attack FALCON: Implementation Attacks with Countermeasures on the FALCON signature scheme. *IACR Cryptol. ePrint Arch.* 2019 (2019), 478.

[40] Hamid Nejatollahi, Nikil Dutt, Sandip Ray, Francesco Regazzoni, Indranil Banerjee, and Rosario Cammarota. 2019. Post-quantum lattice-based cryptography implementations: A survey. *ACM Computing Surveys (CSUR)* 51, 6 (2019), 1–41.

[41] Peter Pessl and Robert Primas. 2019. More practical single-trace attacks on the number theoretic transform. In *International Conference on Cryptology and Information Security in Latin America*. Springer, 130–149.

[42] David Pointcheval and Jacques Stern. 2000. Security arguments for digital signatures and blind signatures. *Journal of cryptology* 13, 3 (2000), 361–396.

[43] Robert Primas, Peter Pessl, and Stefan Mangard. 2017. Single-trace side-channel attacks on masked lattice-based encryption. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 513–533.

[44] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. 2019. Exploiting determinism in lattice-based signatures: practical fault attacks on pqm4 implementations of NIST candidates. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. 427–440.

[45] Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2019. Number "not used" once-practical fault attack on pqm4 implementations of NIST candidates. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 232–250.

[46] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* 56, 6 (2009), 1–40.

[47] Brian Sauser, Ryan Gove, Eric Forbes, and Jose Emmanuel Ramirez-Marquez. 2010. Integration maturity metrics: Development of an integration readiness level. *Information Knowledge Systems Management* 9, 1 (2010), 17–46.

[48] Brian Sauser, Dinesh Verma, Jose Ramirez-Marquez, and Ryan Gove. 2006. From TRL to SRL: The concept of systems readiness levels. In *Conference on Systems Engineering Research, Los Angeles, CA*. Citeseer, 1–10.

[49] Reijo M. Savola. 2013. Quality of security metrics and measurements. *Computers & Security* 37 (2013), 78–90. https://doi.org/10.1016/j.cose.2013.05.002

[50] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (jan 1999), 303–332. https://doi.org/10.1137/S0036144598347011

[51] Rajeev Sobti and Ganesan Geetha. 2012. Cryptographic hash functions: a review. *International Journal of Computer Science Issues (IJCSI)* 9, 2 (2012), 461.

[52] G Wendin. 2017. Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics* 80, 10 (sep 2017), 106001. https://doi.org/10.1088/1361-6633/aa7e1a

[53] George O.M. Yee. 2013. Chapter 32 - Security Metrics: An Introduction and Literature Review. In *Computer and Information Security Handbook (Second Edition)* (second edition ed.), John R. Vacca (Ed.). Morgan Kaufmann, Boston, 553–566. https://doi.org/10.1016/B978-0-12-394397-2.00032-5

[54] Raymond K Zhao, Ron Steinfeld, and Amin Sakzad. 2019. FACCT: fast, compact, and constant-time discrete Gaussian sampler over integers. *IEEE Trans. Comput.* 69, 1 (2019), 126–137.