A Note on the Hint in the Dilithium Digital Signature Scheme

Amit Berman, Ariel Doubchak, and Noam Livne*

Advanced Flash Solutions Lab, Samsung R&D Center, Israel

Abstract

In the Dilithium digital signature scheme, there is an inherent tradeoff between the length of the public key, and the length of the signature. The coefficients of the main part of the public-key, the vector $\mathbf{t}$, are compressed (in a lossy manner), or "quantized", during the key-generation procedure, in order to save on the public-key size. That is, the coefficients are divided by some fixed denominator, and only the quotients are published. However, this results in some "skew" during the verification process, and to fix this, a special signature-dependent "hint" is computed during the signing process. Roughly speaking, stronger compression of $\mathbf{t}$ results in the hint carrying more information, consequently increasing the signature length. Prior to the hint computation, a test is performed to check whether a proper hint can indeed be composed to fix this skew, and if the test fails, the signing process is rerun with a different seed for the (pseudo-)randomness. However, in this short report we observe that this test is not performed optimally: the test calculates a sufficient condition for the hint to work, but not a necessary one. We suggest a new refined test that results in a lower probability for the sign iteration to fail. The new test exhibits some improvement (in terms of expected running time) in certain configurations that are characterized by shorter public-key length on the expense of slightly longer signature length. It is noted that the change does not imply any change in the security of the algorithm.

Keywords: Dilithium, hint, digital signature scheme, module lattices.

1. Introduction

The Dilithium algorithm [1] is a digital signature scheme based on the hardness of lattice problems over module lattices. It is one of the winners of the NIST Post-Quantum Cryptography Standardization contest. In this short note we point out a sub-optimality in one

*Corresponding author: noam.livne@samsung.com

of the procedures of the algorithm, namely, the hint generation. In the rest of this note we will follow closely the notation and definitions of [1]. The algorithm is also published as a NIST publication as FIPS204 [2].

During the verification process of the Dilithium digital signature scheme, the verifier attempts to calculate the value $\mathbf{w_1} = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$ that is calculated during the signing process, where $\mathbf{w} = \mathbf{Ay}$. However, the verifier cannot calculate $\mathbf{w}$ exactly, but only the "noised" value $\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}$.

The signer, thus, when generating the signature, assures that calculating $\mathbf{w_1} = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$ will be possible given $\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}$. This is done in two phases:

- It first checks whether $\mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$ and fails the iteration if the answer is negative.
- It then generates a "hint", which enables retrieving $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$ from $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}, 2\gamma_2)$.

The hint is generated by the MakeHint procedure, and consists of a sequence of binary values that indicate the coordinates where the vector $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$ is different from the vector $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}, 2\gamma_2)$. In other words, the hint indicates the locations where the subtraction of $c\mathbf{t_0}$ changes the high bits.

The UseHint procedure calculates $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$ from $\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}$ and the hint. Since the hint only indicates for each coordinate if there is a difference between the two vectors, but not its magnitude and sign, two requirements assure that the UseHint procedure can perform its task:

- In all coordinates, the difference between the two vectors belongs to $\{-1, 0, 1\}$ modulo $(q-1)/2\gamma_2$.
- In every coordinate, if the hint equals 1, the sign of the difference between the two vectors is deducible from the value of $\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}$ at this coordinate (via some rule to be described in the following).

Given these two requirements, if the hint indicates for some coordinate that there is a difference between the two vectors, that is, that the subtraction of $c\mathbf{t_0}$ changes its high bits, the verifier knows the difference's magnitude is 1, and it can deduce its sign, thus he knows how to correct it. Thus, given $\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}$ and the hint, the verifier can calculate $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$, which is equal to $\mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$ – the desired value.

The rest of this note is organized as following: in Chapter 2 we describe the current test to achieve the aforementioned two requirements, it's sub-optimallity, and our improved test. We then prove the correctness of our new test and argue about its security. In Chapter 3 we demonstrate the improvement in different configurations.

2. Proposed Change

2.1 The Current Way and Its Sub-optimality

The signing process is a (pseudo-)randomized process that contains several tests, and is rerun with different randomness until all tests pass. The test we refer to in this note relates to the vector $c\mathbf{t_0}$. In the current way, the test consists of checking that $\|c\mathbf{t_0}\|_\infty < \gamma_2$. If the answer is negative, the iteration fails. If the test passes, all locations for which $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}, 2\gamma_2)$ is different than $\mathsf{HighBits}_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$ are marked in the hint with '1'.

It is noted that with current parameters (in all three modes) the hint is typically a sparse vector, and in order to save on the signature length, it is compressed as following: for each of the k hint 256-vectors, specify its weight in one byte, and list all its 1's indexes, each in one byte. In order to keep the signature length fixed, another test is performed to count the number of 1's in the hint vector, and if it exceeds some threshold, meaning the aforementioned compression will be too long, the iteration fails. Besides this, two additional tests are performed during the signing process to assure that no information on $\mathbf{s_1}$ and $\mathbf{s_2}$ is leaked.

Clearly the condition $\|c\mathbf{t_0}\|_\infty < \gamma_2$ is sufficient to ensure the first requirement, that for all i it holds that $\mathsf{HighBits}_q((\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0})_i, 2\gamma_2) = \mathsf{HighBits}_q((\mathbf{w} - c\mathbf{s_2})_i, 2\gamma_2) \pm 1 \mod (q-1)/2\gamma_2$.[1] However, this is not a necessary condition and in fact $\|c\mathbf{t_0}\|_\infty \leq 2\gamma_2$ is also sufficient (but still not necessary, as we will soon see). As for the second requirement, in terms of limiting the infinity-norm of $c\mathbf{t_0}$ it is a sufficient and (nearly) necessary condition,[2] that is, γ_2 is the largest value the norm of $c\mathbf{t_0}$ can be limited to in order to meet the requirement. However, our observation is that instead of simply limiting the infinity-norm of $c\mathbf{t_0}$, we can test each of its coefficients independently, with respect to the corresponding coefficient in $\mathbf{w} - c\mathbf{s_2}$. If we allow such test, then testing that the norm of every coefficient of $c\mathbf{t_0}$ is smaller or equal to γ_2 is not necessary also with respect to the second requirement. This is because while a coefficient of $c\mathbf{t_0}$ can distance a coefficient of $\mathbf{w} - c\mathbf{s_2}$ too far for the hint to work, exactly how far can it distance it depends on the initial value of the coefficient of $\mathbf{w} - c\mathbf{s_2}$ (and in particular, on its low bits).

The norm of a coefficient of $c\mathbf{t_0}$ can be larger than γ_2 without failing the hint, and only in the extreme case a coefficient of $\mathbf{w} - c\mathbf{s_2}$ bounds the norm of the corresponding coefficient of $c\mathbf{t_0}$ by γ_2, and even then it is only in one direction (i.e., depending on the coefficient of $\mathbf{w} - c\mathbf{s_2}$ the coefficient of $c\mathbf{t_0}$ need be either larger or equal to $-\gamma_2$ or less than or equal to γ_2).

Performing such coordinate-wise test allows for two types of coefficients of $c\mathbf{t_0}$ that

[1] For ease of notation, throughout this note we will take i as a double index into one coefficient of one polynomial in the module vector, so $i = (i_1, i_2)$ where $1 \leq i_1 \leq k$ is the polynomial index and $1 \leq i_2 \leq 256$ is the coefficient index.

[2] Nearly – because while in [1], as well as in [2], the value is limited to $\gamma_2 - 1$, it is in fact sufficient to limit it to γ_2.

would otherwise fail the test: coefficients that would not even yield '1' in the corresponding location in the hint, and coefficients that would yield '1' in the corresponding location in the hint, but are still correctable by the UseHint procedure.

2.2 Improved Test

In this light, we propose the following new test, instead of testing that $\|c\mathbf{t_0}\|_\infty \leq 2\gamma_2$:

Given some i, denote:

- $r_{0,i} = \mathsf{LowBits}_q((\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0})_i, 2\gamma_2)$
- $r_{1,i} = \mathsf{HighBits}_q((\mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0})_i, 2\gamma_2)$

Denote also $m = (q-1)/\alpha$.

New Test. *For each i, check that one of the following two conditions hold:*

- $r_{0,i} > 0$ *and* $-\gamma_2 < r_{0,i} - (c\mathbf{t_0})_i \leq 3\gamma_2$
  (or $r_{1,i} = m-1$, $r_{0,i} > 0$ *and* $r_{0,i} - (c\mathbf{t_0})_i = 3\gamma_2 + 1$*)*
- $r_{0,i} \leq 0$ *and* $-3\gamma_2 < r_{0,i} - (c\mathbf{t_0})_i \leq \gamma_2$
  (or $r_{1,i} = 0$, $r_{0,i} \leq 0$ *and* $r_{0,i} - (c\mathbf{t_0})_i = -\gamma_2 - 1$*)*

Note that with the new test, the allowed range for the coefficients of $c\mathbf{t_0}$ is doubled.

2.3 Proof of Correctness

The new proposed test does not require any consequent changes in the algorithm. In particular, the MakeHint and UseHint procedures work just as before. To prove the new test validity, we will follow [1]. Lemma 1 in [1] consists of three parts, stating three facts about the generation and usage of the hint. Lemmas 5,6, which prove parts 2,3 of Lemma 1, stay true after the change, and their proofs need not be changed. As for Lemma 4, which proves part 1 of Lemma 1, its conclusion can be proved from a new hypothesis that reflects the new test. We change Lemma 4 accordingly, and reprove its conclusion.

Lemma 4. *Let $r, z \in \mathbb{Z}_q$, and denote $r_0 = \mathsf{LowBits}_q(r, \alpha)$ and $r_1 = \mathsf{HighBits}_q(r, \alpha)$.*

Suppose one of the following two conditions hold (the parts in parentheses relate to the special case of the last segment of $\{0, ..., q-1\}$ which is larger by one, as explained in Section 2.4 in [1]):

- $r_0 > 0$ *and* $-\alpha/2 < r_0 + z \leq 3\alpha/2$
  (or $r_1 = m-1, r_0 > 0$ *and* $r_0 + z = 3\alpha/2 + 1$*.)*
- $r_0 \leq 0$ *and* $-3\alpha/2 < r_0 + z \leq \alpha/2$
  (or $r_1 = 0, r_0 \leq 0$ *and* $r_0 + z = -3\alpha/2$*.)*

Then:

$$\mathsf{UseHint}_q(\mathsf{MakeHint}_q(z, r, \alpha), r, \alpha) = \mathsf{HighBits}_q(r + z, \alpha)$$

Proof. Denote $m = (q-1)/\alpha$ as before. The two conditions can be rephrased as the following three mutually exclusive conditions (again, the parts in parentheses relate to the special case of the last segment of $\{0, ..., q-1\}$ which is larger by one):

1. $-\alpha/2 < r_0 + z \leq \alpha/2$
   (or $r_1 = 0$ and $r_0 + z = -\alpha/2$)
   (hint=0)

2. $r_0 > 0$ and $\alpha/2 < r_0 + z \leq 3\alpha/2$
   (or $r_1 = m - 1, r_0 > 0$ and $r_0 + z = 3\alpha/2 + 1$)
   (hint=1 and and high bit is increased)

3. $r_0 \leq 0$ and $-3\alpha/2 < r_0 + z \leq -\alpha/2$
   (or $r_1 = 1, r_0 \leq 0$ and $r_0 + z = -3\alpha/2 + 1$)
   (hint=1 and and high bit is decreased)

Note that for any r, r_0 and r_1 defined as above, $r = r_1\alpha + r_0$, with $0 \leq r_1 < m$ and $-\alpha/2 < r_0 \leq \alpha/2$ or with $r_1 = 0$ and $r_0 = -\alpha/2$, and that this decomposition is unique.

If condition 1 is met, then $r + z = r_1\alpha + r_0 + z$ where $0 \leq r_1 < m$, and from the condition $-\alpha/2 < r_0 + z \leq \alpha/2$ or $r_1 = 0$ and $r_0 + z = -\alpha/2$. It follows that $\mathsf{HighBits}_q(r + z, \alpha) = r_1 = \mathsf{HighBits}_q(r, \alpha)$. Thus, $\mathsf{MakeHint}_q(r, z, \alpha)$ will output $h = 0$, and $\mathsf{UseHint}_q(h, r, \alpha)$ will output r_1 as required.

If condition 2 is met, then we can write $r+z = r_1\alpha+r_0+z = (r_1+1)\alpha+r_0+z-\alpha$ with $r_0 > 0$ and $-\alpha/2 < r_0 + z - \alpha \leq \alpha/2$ (or $r_1 + 1 = m, r_0 > 0$ and $r_0 + z - \alpha = \alpha/2 + 1$). If follows that $\mathsf{HighBits}_q(r + z, \alpha) = r_1 + 1 (\mod m) \neq r = \mathsf{HighBits}_q(r, \alpha)$. Thus, $\mathsf{MakeHint}_q(r, z, \alpha)$ will output $h = 1$, and since $r_0 > 0$ $\mathsf{UseHint}_q(h, r, \alpha)$ will output $r_1 + 1 \mod m$ as required.

If condition 3 is met, then we can write $r+z = r_1\alpha+r_0+z = (r_1-1)\alpha+r_0+z+\alpha$ with $r_0 \leq 0$ and $-\alpha/2 < r_0 + z + \alpha \leq \alpha/2$ (or $r_1 - 1 = 0, r_0 \leq 0$ and $r_0 + z + \alpha = -\alpha/2 + 1$). It follows that $\mathsf{HighBits}_q(r + z, \alpha) = r_1 - 1 (\mod m) \neq r = \mathsf{HighBits}_q(r, \alpha)$. Thus, $\mathsf{MakeHint}_q(r, z, \alpha)$ will output $h = 1$, and since $r_0 \leq 0$ $\mathsf{UseHint}_q(h, r, \alpha)$ will output $r_1 - 1 \mod m$ as required.

□

2.4 Security

Our new test has no impact on the security of the algorithm compared to the old test. Regarding potential leakage of information: like the old test, the new test may leak information about $c\mathbf{t_0}$, but this value is not secret. It may also leak some information about $\mathbf{w} - c\mathbf{s_2}$, but the previous test that $\|r_0\|_\infty < \gamma_2 - \beta$ ensures that knowing even the

exact value of $\mathbf{w} - c\mathbf{s_2}$ does not leak any information about $c\mathbf{s_2}$, and thus on $\mathbf{s_2}$. Besides that, since the verify process is unchanged, the challenge for a potential forger stays the same.

3. Improvement

In the dilithium algorithm there is an inherent tradeoff between the length of the public key, and the length and runtime of the signature. Compressing the public key stronger, that is, increasing d and thus reducing more bits from the vector $\mathbf{t}$, means the magnitude of $\mathbf{t_0}$, and thus of $c\mathbf{t_0}$, increases. In particular, every extra bit that is reduced from the coefficients of $\mathbf{t}$ scales the distribution of the coefficients of $c\mathbf{t_0}$ by a factor of 2. This has two consequences: the $c\mathbf{t_0}$ test (both old and new) fails with higher probability, and the Hamming weight of the hint increases. Our new test affects only the first consequence: the new test fails with lower probability, but the hint weight is unchanged (that is, iterations that pass both old and new tests generate exactly the same hint).

We have incorporated our new test into the Dilithium algorithm and measured its performance in different configurations. In Table 1 we show three configurations where our new test exhibits advantage. These configurations are the 3 configurations from [1] with d increased (and all other parameters of the mode unchanged). Thus, in these configurations the public key length is reduced, but, as described above, the hint weight increases. This renders the naïve compression currently used for the hint compression detrimental, thus imposing a small increase on the signature length. However, once no compression is used there is also no point in limiting the hint weight, thus less iterations fail and we enjoy some reduction in expected running time (this is true for both old and new test). Thus, our configurations exhibit reduction in both public key length and running time, on the expense of increase in signature length. The difference between the old and new test is in the runtime reduction: with the new test, the reduction is larger. It is also noted that increasing d slightly harms the security of the protocol, and we do not quantify this consequence. Removing the limit on the hint weight also slightly harms the security, but this limit is not accounted for also with the original test, and is only mentioned in passing in [1].

One way to view these results is as following: while the option of increasing d in order to reduce public key length on the expense of increasing signature length was not favoured with the old test, the new test exhibits a better tradeoff due to shorter signature running time compared to the old test for this option. Thus, the new test may tip the scale in favor of this choice.

Note that if one further allows non-trivial hint-compression in the signing process, the advantage over the old test in running time can be traded for an advantage in the hint size (as more running time means the threshold for compression failure can be less strict), resulting in smaller increase (compared to the configuration with $d = 13$) in signature

Configuration	Expected number of iterations, old test	Expected number of iterations, new test	Runtime reduction	PK length saving (vs. $d = 13$), old & new	Signature length increase (vs. $d = 13$), old & new
Mode 2, $d = 14$	5.22	4.45	14.78%	9.76%	1.82%
Mode 3, $d = 15$	5.55	5.23	5.80%	19.67%	3.98%
Mode 5, $d = 15$	4.65	4.13	11.23%	19.75%	3.76%

Table 1: Improvemnt of new test over old test

length over the old test.

Acknowledgments

The authors would like to thank Vadim Lyubashevsky for a helpful discussion that contributed to this note.

References

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (Version 3.1)," National Institute of Standards and Technology, Technical Report, 2021, post-Quantum Cryptography Standardization Process. [Online]. Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

[2] National Institute of Standards and Technology, "FIPS 204: CRYSTALS-DILITHIUM Digital Signature Algorithm," National Institute of Standards and Technology, Federal Information Processing Standards Publication 204, February 2024, dOI: https://doi.org/10.6028/NIST.FIPS.204. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf