Don't throw your nonces out with the bathwater

Daan Sprenkels [*1] and Bas Westerbaan[†2]

[1]Radboud Universiteit Nijmegen
[2]University College London & Cloudflare

September 22, 2020

Abstract

We suggest a small change to the Dilithium signature scheme [DKL+], that allows reusing computation between aborted attempts for a speed-up in signing time.

To create a signature in Dilithium, one picks a vector $\mathbf{y} = (y_1, \ldots, y_\ell)$ over the ring $R = \mathbb{F}_q/(x^{256}+1)$. Not every $\mathbf{y}$ will do: the vector is subjected to four checks to ensure correctness of the algorithm and prevent leakage of the private key. If $\mathbf{y}$ fails any of the tests, the attempt is *aborted* and a fresh $\mathbf{y}$ is sampled to try again. Depending on the instance of Dilithium, one expects having to resample $\mathbf{y}$ between 3.3 and 5.6 times.

Out of the four checks, the following two cause more than 99% of the aborts.

$$\|\mathbf{y} + c\mathbf{s}_1\|_\infty \;\leq\; \gamma_1 - \beta \tag{1}$$
$$\|\mathbf{w}_0 - c\mathbf{s}_2\|_\infty \;\leq\; \gamma_2 - \beta \tag{2}$$

Here β, γ_1 and γ_2 are scheme parameters; $\mathbf{s}_1 \in R^\ell$ and $\mathbf{s}_2 \in R^k$ are part of the private key; $\mathbf{w}_0 \in R^k$ is part of the decomposition $\mathbf{w}_0 + 2\mathbf{w}_1\gamma_2 = \mathbf{A}\mathbf{y}$ with $\mathbf{w}_0$ and $\mathbf{w}_1$ suitably small and $\mathbf{A}$ an $k \times \ell$ matrix over R which is part of the public key *and* $c \in R$ is a challenge generated via hashing from $\mathbf{w}_1$.

As the norm is the sup norm, the first check involves the following ℓ subchecks, one for each component of $\mathbf{y}$: $\|y_i + c(\mathbf{s}_1)_i\|_\infty \leq \gamma_1 - \beta$. If the first subcheck fails (without having performed the other checks or subchecks), then instead of aborting completely and resampling all elements of $\mathbf{y}$, we propose to resample y_1 but keep $y_2, \ldots, y_\ell$. This allows one to reuse the computations of $A_{ij}y_j$ for $j \neq 1$, which were required to compute c via $\mathbf{A}\mathbf{y}$. Note that the challenge c will be different after this partial abort.

If the first check fails at y_2 (after y_1 passed) then we cannot reuse y_1, because it will have to pass the check for at least one other challenge c. This will

[*]daan@dsprenkels.com
[†]bas@westerbaan.name

introduce a bias in y_1, although it is unclear to us whether this bias could lead to a practical attack. Instead we propose to resample only $y_1, \ldots, y_i$ if the first check fails at y_i (and only having checked $y_1, \ldots, y_i$).

As Dilithium is required to be a deterministic scheme for the NIST PQC competition, the $\mathbf{y}$s are derived deterministically using a PRF. To support our optimization the method of sampling of $\mathbf{y}$ needs to be adapted and we propose to do that as follows.

The current version of Dilithium (round 2) uses a single counter κ to keep track of the number of $\mathbf{y}$ elements that have been generated thusfar.[1]

We replace κ with $\boldsymbol{\kappa}$, a vector of length ℓ, wherein κ_i holds the number of times that y_i has been sampled before. That is, the counter κ_i is incremented after a corresponding y_i has been produced by ExpandMask.

Currently ExpandMask generates y_i by absorbing the seed ρ' and then κ as two bytes in little endian byte order into SHAKE-256. Instead we propose ExpandMask first absorbs κ_i as two bytes in little endian byte order after ρ' and then absorbs i as one byte.

To get a rough performance estimate, we wrote a Sage script to simulate Dilithium signature generation to measure the number of components of $\mathbf{y}$ to be sampled with plain Dilithium and with our proposal. Results are listed in Table 1.

Paramset	Baseline	Modified	Decrease
Dilithium II	17.53	14.84	15%
Dilithium III	26.66	22.03	17%
Dilithium IV	21.64	18.12	16%

Table 1: Average number of $\mathbf{y}$-element sampled during the Dilithium signature generation. Averages were computed over 10 000 runs.

Although throughout this paper we have described this optimization in the context of Dilithium, we stress that it is in fact applicable to any scheme that incorporates some rejection-sampling loop; including all *Fiat-Shamir with aborts* [Lyu09] schemes; and including schemes that sample from a distribution that is different from the one used in Dilithium.

Acknowledgements This work was supported in part by the European Commission through the ERC Starting Grant 805031 (EPOQUE).

References

[DKL+] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium – submission to round 2 of the NIST post-quantum project.

[1] Note that there is a small typo in the round 2 specification: line 24 of [DKL+, Fig. 4] reads $\kappa := \kappa + 1$ whereas it's clear that $\kappa := \kappa + \ell$ was meant.

[Lyu09] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009. https://www.iacr.org/archive/asiacrypt2009/59120596/59120596.pdf.