On the Properties of HighBits and LowBits Functions and their Applications

Alice Pellet-Mary[1] and Michel Seck[2]

alice.pellet-mary@math.u-bordeaux.fr, mseck@ept.edu.sn

[1] Univ Bordeaux, CNRS, Inria, Bordeaux INP, IMB, UMR 5251, Talence, France
[2] Ecole Polytechnique de Thies, CRISIN'2D, LTISI, Thies, Senegal

Abstract. ML-DSA is a lattice-based signature scheme that has recently been standardized by NIST as FIPS 204. Among the many subroutines used by ML-DSA are the high bits (Hb) and low bits (Lb) functions, which, as their name suggest, return only the higher bits or the lower bits of some integer. Recently, Seck and Roux-Langlois (IACR CiC 2025) conjectured that the high bits of a sum of two integers t and r can be expressed as the sum of the high bits of each integer, plus an error term that can take at most seven different values. In this paper, we study the properties of the functions Hb and Lb and we prove that the conjecture of Seck and Roux-Langlois holds. In addition, we provide a complete characterization of the error term. As an application, we explain how these properties can be used to design lattice-based signature schemes with advanced features.

Keywords: HighBits · LowBits · ML-DSA · Advanced Signature Schemes

1 Introduction

Most of the widely used public-key cryptographic schemes in the last decades such as ECDSA [Mil86] and RSA [RSA78] are based on the hardness of problems like integer factorization and discrete logarithm. However, these two problems can be efficiently solved using Shor's algorithm [Sho94] on a sufficiently large quantum computer. To address this issue, new cryptographic schemes have been developed, with the hope that they would remain secure against quantum attacks: this is known as post-quantum cryptography. Since December 2016, NIST (National Institute of Standards and Technology) [Nat16] has been leading an effort to standardize post-quantum cryptographic algorithms. In 2024, they announced the selection of several algorithms for standardization, including ML-DSA [Nat24a], a lattice-based signature scheme build on top of CRYSTALS-Dilithium [BDK+21; DKL+18; CGT+23], as a winner of the signature category and ML-KEM [Nat24b], a lattice-based key encapsulation mechanism build on top of CRYSTALS-Kyber [BDK+18], as a winner of the KEM category.

In the NIST's standard ML-DSA [Nat24a], a function high bits (which we will denote Hb in this article, and corresponds to the first part of the function Power2Round in [BDK+21, Fig. 3]) is used to compress the public key. More precisely, a public key pk is in the form $(\rho, \mathsf{Hb}(t))$ instead of (ρ, t) where $t = \mathbf{A}s_1 + s_2$ is an instance of Module learning with errors (MLWE) problem and ρ is a seed used to generate the matrix $\mathbf{A}$. This compression technique was introduced by Bai and Galbraith [BG14] and is used in many other lattice-based signature schemes such as HAETAE [CCD+24] a winner of the KpqC [BCG+24] competition, and Racoon [PKP+24], that was submitted to the NIST's standardization process for additional post-quantum primitives. The high bits and low bits functions are also used in the design of signature schemes with advanced features such as signature schemes with rerandomizable keys [SR25] and threshold signatures [DPKM+24; BCP+25].

In 2023, Boudgoust and Takahashi [BT23] pointed out some vulnerabilities in the Dilithium based multi-signature scheme by Fukumitsu and Hasegawa [FH20], describing how to mount a (partial) secret-key recovery attack. They proposed a fix to this attack using the high bits function Hb but the non-linearity of Hb (i.e. $\mathsf{Hb}(t+r) \neq \mathsf{Hb}(t) + \mathsf{Hb}(r)$ in general) introduced other challenges. In 2024, Snetkov *et al.* [SVL24] designed an efficient lattice-based two party threshold signature scheme on top of Dilithium, thanks to the properties of the high bits functions.

Recently, Seck and Roux-Langlois [SR25] constructed Dilithium-RK, an advanced signature scheme build on top of Dilithium with non-compressed public keys. For the compression of the public keys, they conjectured that

$$\mathsf{Hb}(t+r) = \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r))\alpha) + \varepsilon, \tag{1}$$

where ε is an error term that can take at most seven different values, and where α is a parameter of the Hb function (the low bits of an integer t correspond to $t \bmod^{\pm}(\alpha)$). In this paper, we study the properties of the functions Hb and Lb and give a positive answer to the conjecture of Seck and Roux-Langlois [SR25]. Note that the function Hb depends on the choice of two parameters q and α, which are chosen respectively as a prime integer and a power-of-two in Dilithium. Our proof holds for arbitrary positive integers q and α, as long as $q > \alpha$.

Contributions. Our main contributions can be summarized as follows:

1. We prove in Lemma 3.1 that the high bits (resp. the low bits) of a sum of two integers can be expressed as the sum of the high bits (resp. the low bits) of each integer, plus an error term that can take at most six different values. More precisely, we show that for any integers t and r, we have $\mathsf{Hb}(t+r) = \mathsf{Hb}(t) + \mathsf{Hb}(r) + t^h$ and $\mathsf{Lb}(t+r) = \mathsf{Lb}(t) + \mathsf{Lb}(r) + t^l$, where t^h and t^l can take at most the six different values, which we specify depending on the choice of the parameters q and α. We extend this result to the case

of a sum of n integers $t_1, t_2, \ldots, t_n$ in Theorems 3.3 and 3.4.

2. We prove in Theorem 3.6 that the conjecture of Seck and Roux-Langlois [SR25] holds. More specifically, $\mathsf{Hb}(t+r) = \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r))\alpha) + \varepsilon$ where ε is an error term that can take at most seven different values. Furthermore, we provide a complete characterization of the error term ε depending on the choice of the parameters q and α.

3. We show how these properties can be used to design a lattice-based signature scheme with advanced features such as rerandomization of both public and secret keys on top of ML-DSA.

Paper Organization. The rest of the paper is organized as follows. In Section 2, we introduce necessary notations and definitions used in the paper. In Section 3, we present our main contributions. First, we prove some properties of the Hb and Lb functions and then we prove that the conjecture of Seck and Roux-Langlois [SR25] holds. The last Section 4 is dedicated to one application of these properties in the design of a lattice-based signature scheme with rerandimizable keys.

Acknowledgments. Alice Pellet-Mary is supported by the France 2030 program, managed by the French National Research Agency (ANR-22-PETQ-0008 PQ-TLS). Michel Seck is supported by the French National Research Organization CNRS through the CNRS-Afrique project DIALS (2025-26).

2 Preliminaries

For two finite sets of integers S_1 and S_2, we define $S = S_1 + S_2$ as the set

$$S = \{s_1 + s_2 \mid s_1 \in S_1, s_2 \in S_2\}.$$

Let $n > 0$ be an integer. For an integer r, we define $r' = r \bmod^{\pm}(n)$ to be the unique integer r' in the interval $(-\frac{n}{2}, \frac{n}{2}]$ such that $r' \equiv r \pmod{n}$. We also define $r' = r \bmod^{+}(n)$ to be the unique integer r' in the range $0 \leq r' < n$ such that $r' \equiv r \pmod{n}$.

Definition 2.1 (Power2Round, Hb, and Lb). *Let q, α be two positive integers such that $q > \alpha$. For any integer $r \in \mathbb{Z}$, we define the function lower bits $\mathsf{Lb}_{q,\alpha}(r) = r_0 := r_q \bmod^{\pm}(\alpha)$, where $r_q = r \bmod^{+}(q)$. We also define the function higher bits $\mathsf{Hb}_{q,\alpha}(r) = r_1 := (r_q - r_0)/\alpha$, where r_q and r_0 are as before. The function $\mathsf{Power2Round}_{q,\alpha} : \mathbb{Z} \to \mathbb{Z} \times \mathbb{Z}$ is defined as*

$$\mathsf{Power2Round}_{q,\alpha}(r) = (\mathsf{Hb}_{q,\alpha}(r), \mathsf{Lb}_{q,\alpha}(r))$$

If the parameters q and α are clear from the context, we simply denote these functions by Hb, Lb *and* Power2Round *respectively.*

Remark 2.2. Notice that if we have two integers $r, r' \in \mathbb{Z}$ such that $r = r' \bmod q$, then $\mathsf{Hb}_{q,\alpha}(r) = \mathsf{Hb}_{q,\alpha}(r')$ and $\mathsf{Lb}_{q,\alpha}(r) = \mathsf{Lb}_{q,\alpha}(r')$. So we can extend the functions $\mathsf{Hb}_{q,\alpha}(\cdot)$ and $\mathsf{Lb}_{q,\alpha}(\cdot)$ to elements of $\mathbb{Z}_q$.

It is important to observe that the functions Hb and Lb defined above are *not* the same as the functions $\mathsf{HighBits}$ and $\mathsf{LowBits}$ from Dilithium [BDK+21, Fig. 3]. Indeed, Dilithium uses two very similar high-bits/low-bits decomposition functions, which differ only on a border case: the $\mathsf{Power2Round}$ and the $\mathsf{Decompose}$ functions. The functions $\mathsf{HighBits}$ and $\mathsf{LowBits}$ from [BDK+21, Fig. 3] correspond to the two components of the function $\mathsf{Decompose}$. However, in this article, we are interested in the function $\mathsf{Power2Round}$ (which is the one used to compress the public key), and what we call Hb and Lb are the two components of this slightly different function.

3 Properties of Hb and Lb Functions

3.1 Hb and Lb of a sum

In this section, we explore some important properties of the $\mathsf{Hb}(\cdot)$ and $\mathsf{Lb}(\cdot)$ functions from Definition 2.1. These properties will be useful in the design of advanced signature schemes based on the Dilithium framework. We are mainly interested in the relationship between $\mathsf{Hb}_{q,\alpha}(\sum t_i)$ (resp. $\mathsf{Lb}_{q,\alpha}(\sum t_i)$) and $\sum \mathsf{Hb}_{q,\alpha}(t_i)$ (resp. $\sum \mathsf{Lb}_{q,\alpha}(t_i)$) for a set of integers $\{t_i\}$.

Sum of 2 elements. We start by proving a result in the case where we sum over two t_i's only. This will serve as the base case for the more general results stated in Theorems 3.3 and 3.4.

Lemma 3.1. *Let q and α be two positive integers such that $q > \alpha$. Define $q_0 = q \bmod^{\pm} (\alpha)$ and $q_1 = \frac{q-q_0}{\alpha}$ (i.e. $q = q_1\alpha + q_0$). For any integers $t, \hat{t} \in \mathbb{Z}$, the following properties hold:*

$$\mathsf{Hb}(t+\hat{t}) = \mathsf{Hb}(t) + \mathsf{Hb}(\hat{t}) + \varepsilon^h \text{ and } \mathsf{Lb}(t+\hat{t}) = \mathsf{Lb}(t) + \mathsf{Lb}(\hat{t}) + \varepsilon^l,$$

where

$$\varepsilon^h \in \{-1, 0, 1, -q_1 - 1, -q_1, -q_1 + 1\} \text{ and } \varepsilon^l \in \{-\alpha, 0, \alpha, -q_0 - \alpha, -q_0, -q_0 + \alpha\}.$$

Proof. Since $\mathsf{Hb}_{q,\alpha}(\cdot)$ and $\mathsf{Lb}_{q,\alpha}(\cdot)$ are invariant when we add integer multiples of q, we can assume without loss of generality that $t, \hat{t} \in [0, q)$. Define the sum $S = t + \hat{t} \bmod^{+} (q)$. We can write:

$$t = t_1\alpha + t_0, \quad \hat{t} = \hat{t}_1\alpha + \hat{t}_0,$$

where $t_1 = \mathsf{Hb}(t)$, $t_0 = \mathsf{Lb}(t)$, $\hat{t}_1 = \mathsf{Hb}(\hat{t})$ and $\hat{t}_0 = \mathsf{Lb}(\hat{t})$. Then, we have:

$$t + \hat{t} = (t_1 + \hat{t}_1)\alpha + (t_0 + \hat{t}_0).$$

Moreover, we have $t + \hat{t} \in [0, 2q)$, so there is an $\varepsilon \in \{0, 1\}$ such that

$$S = t + \hat{t} - \varepsilon q = (t_1 + \hat{t}_1 - \varepsilon q_1)\alpha + (t_0 + \hat{t}_0 - \varepsilon q_0).$$

Now, we need to consider the possible values of $A := t_0 + \hat{t}_0 - \varepsilon q_0 \in \left(-\frac{3\alpha}{2}, \frac{3\alpha}{2}\right)$:

- If $-\frac{\alpha}{2} < A \leq \frac{\alpha}{2}$, then

$$\mathsf{Hb}(S) = t_1 + \hat{t}_1 - \varepsilon q_1, \quad \mathsf{Lb}(S) = t_0 + \hat{t}_0 - \varepsilon q_0.$$

  Then $\varepsilon^h = -\varepsilon q_1$ and $\varepsilon^l = -\varepsilon q_0$. Since $\varepsilon \in \{0, 1\}$, we have $\varepsilon^h \in \{0, -q_1\}$ and $\varepsilon^l \in \{0, -q_0\}$.
- If $\frac{\alpha}{2} < A \leq \frac{3\alpha}{2}$, then we will shift down A by α and get

$$S = (t_1 + \hat{t}_1 - \varepsilon q_1 + 1)\alpha + (t_0 + \hat{t}_0 - \varepsilon q_0 - \alpha).$$

  Thus,

$$\mathsf{Hb}(S) = t_1 + \hat{t}_1 - \varepsilon q_1 + 1, \quad \mathsf{Lb}(S) = t_0 + \hat{t}_0 - \varepsilon q_0 - \alpha.$$

  Then $\varepsilon^h = 1 - \varepsilon q_1$ and $\varepsilon^l = -\alpha - \varepsilon q_0$. Since $\varepsilon \in \{0, 1\}$, we have $\varepsilon^h \in \{1, 1 - q_1\}$ and $\varepsilon^l \in \{-\alpha, -\alpha - q_0\}$.
- If $-\frac{3\alpha}{2} < A \leq -\frac{\alpha}{2}$, then we will shift up A by α and get

$$S = (t_1 + \hat{t}_1 - \varepsilon q_1 - 1)\alpha + (t_0 + \hat{t}_0 - \varepsilon q_0 + \alpha).$$

  Thus,

$$\mathsf{Hb}(S) = t_1 + \hat{t}_1 - \varepsilon q_1 - 1, \quad \mathsf{Lb}(S) = t_0 + \hat{t}_0 - \varepsilon q_0 + \alpha.$$

  Then $\varepsilon^h = -1 - \varepsilon q_1$ and $\varepsilon^l = \alpha - \varepsilon q_0$. Since $\varepsilon \in \{0, 1\}$, we have $\varepsilon^h \in \{-1, -1 - q_1\}$ and $\varepsilon^l \in \{\alpha, \alpha - q_0\}$.

Combining all the cases, we conclude that

$$\varepsilon^h \in \{-1, 0, 1, -q_1 - 1, -q_1, -q_1 + 1\} \text{ and } \varepsilon^l \in \{-\alpha, 0, \alpha, -q_0 - \alpha, -q_0, -q_0 + \alpha\}.$$

□

Corollary 3.2 (Dilithium parameters). *Let d be a positive integer and q be a prime such that $q \equiv 1 \mod \alpha$ with $\alpha = 2^d$. For any integers $t, \hat{t} \in \mathbb{Z}_q$, the following properties hold:*

$$\mathsf{Hb}(t + \hat{t}) = \mathsf{Hb}(t) + \mathsf{Hb}(\hat{t}) + \varepsilon^h \text{ and } \mathsf{Lb}(t + \hat{t}) = \mathsf{Lb}(t) + \mathsf{Lb}(\hat{t}) + \varepsilon^l,$$

where

$$\varepsilon^h \in \left\{-1, 0, 1, \frac{1-q}{2^d}, \frac{1-q}{2^d} - 1, \frac{1-q}{2^d} + 1\right\}$$

and

$$\varepsilon^l \in \left\{-2^d, 0, 2^d, -2^d - 1, -1, 2^d - 1\right\}.$$

In particular, for Dilithium parameters where $d = 13$ and $q = 8380417$, we have

$$\varepsilon^h \in \{-1, 0, 1, -1024, -1023, -1022\} \text{ and } \varepsilon^l \in \{-8193, -8192, -1, 0, 8191, 8192\}.$$

Numerical experiments. We ran some numerical experiments to verify the correctness of Lemma 3.1 and Corollary 3.2. For Lemma 3.1, We generated 10 random pairs (α, q), with $\alpha \leq 100$ and $200 \leq q \leq 500$. We also generated 8 additional pairs where α is a power-of-two and $q = 1 \bmod \alpha$. We then verified experimentally that for these pairs, the values of ε^h and ε^l achieved by running over all pairs $(t, \hat{t})$ in $[0, q) \times [0, q)$ were all contained in the sets described in Lemma 3.1. For Corollary 3.2, since it was too costly to check for all pairs $(t, \hat{t})$ in $[0, q) \times [0, q)$, we picked 10^5 random pairs $(t, \hat{t})$, and verified that for those pairs, ε^h and ε^l were indeed in the sets described in Corollary 3.2. All the tests were successful. The code used to perform these tests is available at `https://github.com/funibus/africacrypt-2026-supplementary-material.git`.

Sum of n elements. Now that we have proven Lemma 3.1 for the sum of two elements, we can state and prove a more general result, where we sum over an arbitrary number n of elements.

Theorem 3.3. *Let q and α be two positive integers such that $q > \alpha$. Define $q_0 = q \bmod^{\pm}(\alpha)$ and $q_1 = \frac{q-q_0}{\alpha}$ (i.e. $q = q_1\alpha + q_0$). For any integers $t_1, t_2, \ldots, t_n \in \mathbb{Z}$, the following property holds:*

$$\mathsf{Hb}(\sum_{i=1}^{n} t_i) = \sum_{i=1}^{n} \mathsf{Hb}(t_i) + E_n^h$$

where

$$E_n^h \in \{-(n-1), \ldots, 0, \ldots, (n-1)\} + \{0, -q_1, -2q_1, \ldots, -(n-1)q_1\}\,.$$

Proof. We prove the theorem by induction on n.

- **Base case:** For $n = 2$, the result holds by the previous Lemma 3.1.
- **Inductive step:** Assume the result holds for some $k \geq 2$. We need to show it holds for $k + 1$.
  Let $t_1, t_2, \ldots, t_{k+1} \in \mathbb{Z}_q$. By the inductive hypothesis, we have:

$$\mathsf{Hb}(\sum_{i=1}^{k} t_i) = \sum_{i=1}^{k} \mathsf{Hb}(t_i) + E_k^h,$$

  where

$$E_k^h \in \{-(k-1), \ldots, 0, \ldots, (k-1)\} + \{0, -q_1, -2q_1, \ldots, -(k-1)q_1\}\,.$$

  Now, consider:

$$\mathsf{Hb}(\sum_{i=1}^{k+1} t_i) = \mathsf{Hb}(\left(\sum_{i=1}^{k} t_i\right) + t_{k+1}).$$

  Applying the previous lemma to $\sum_{i=1}^{k} t_i$ and t_{k+1}, we get:

$$\mathsf{Hb}(\sum_{i=1}^{k+1} t_i) = \mathsf{Hb}(\sum_{i=1}^{k} t_i) + \mathsf{Hb}(t_{k+1}) + \varepsilon^h,$$

where

$$\varepsilon^h \in \{-1, 0, 1, -q_1, -q_1 - 1, -q_1 + 1\}.$$

Substituting the inductive hypothesis into this equation gives:

$$\mathsf{Hb}(\sum_{i=1}^{k+1} t_i) = \left(\sum_{i=1}^{k} \mathsf{Hb}(t_i) + E_k^h\right) + \mathsf{Hb}(t_{k+1}) + \varepsilon^h.$$

Thus,

$$\mathsf{Hb}(\sum_{i=1}^{k+1} t_i) = \sum_{i=1}^{k+1} \mathsf{Hb}(t_i) + (E_k^h + \varepsilon^h) = \sum_{i=1}^{k+1} \mathsf{Hb}(t_i) + E_{k+1}^h.$$

Now, we need to analyze the term $E_{k+1}^h = E_k^h + \varepsilon^h$. Since

$$E_k^h \in \{-(k-1), \ldots, (k-1)\} + \{0, -q_1, -2q_1, \ldots, -(k-1)q_1\}$$

and

$$\varepsilon^h \in \{-1, 0, 1\} + \{0, -q_1\},$$

therefore, we have as desired

$$E_{k+1}^h \in \{-k, \ldots, k\} + \{0, -q_1, -2q_1, \ldots, -kq_1\}$$

□

Theorem 3.4. *Let q and α be two positive integers such that $q > \alpha$. Define $q_0 = q \bmod^{\pm}(\alpha)$ and $q_1 = \frac{q-q_0}{\alpha}$ (i.e. $q = q_1\alpha + q_0$). For any integers $t_1, t_2, \ldots, t_n \in \mathbb{Z}$, the following property holds:*

$$\mathsf{Lb}(\sum_{i=1}^{n} t_i) = \sum_{i=1}^{n} \mathsf{Lb}(t_i) + E^l$$

where

$$E^l \in \{-(n-1)\alpha, \ldots, 0, \ldots, (n-1)\alpha\} + \{0, -q_0, -2q_0, \ldots, -(n-1)q_0\}.$$

Proof. The proof is similar to that of Theorem 3.3, using the second part of Lemma 3.1 instead. □

3.2 On the Conjecture of Seck and Roux-Langlois

In this section, we study the conjecture of Seck and Roux-Langlois in [SR25] regarding the behavior of the Hb function when applied to sums of two elements of $\mathbb{Z}_q$. The conjecture states that for any $t, r \in \mathbb{Z}_q$, the high bits of their sum can be closely approximated by using the sum of their individual high bits, with a bounded error term ε which can take at most seven different values. More formally, the conjecture is as follows.

Conjecture 3.5 (Seck and Roux-Langlois [SR25]). Let q be a prime number and $\alpha = 2^d$ for some positive integer d. For any $t, r \in \mathbb{Z}_q$, the following holds:

$$\mathsf{Hb}(t+r) = \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r))\alpha) + \varepsilon$$

where ε is an error term that can take at most seven different values. More precisely,

$$\varepsilon \in \{-1, 0, 1, e_4, e_5, e_6, e_7\} \mod q,$$

for some integers e_4, e_5, e_6 and e_7.

Notice that this conjecture is a more sophisticated variant of Lemma 3.1. While in Lemma 3.1, the error term ε can take at most six different values depending on the parameters q and α, in the conjecture of Seck and Roux-Langlois, it is suggested that the error term ε can take up to seven different values, and, for some values of q and α, it can take exactly seven different values. For instance, when $q = 3236147$ and $\alpha = 131072$ (i.e. $d = 17$), the error term ε can take the following seven different values:

$$\varepsilon \in \{-1, 0, 1, -24, 23, 24, 25\} \mod q.$$

In the following, we analyze this conjecture and prove that it holds for arbitrary positive integers q and α such that $q > \alpha$. Depending on the value of $q \bmod^{\pm}(\alpha)$, we show that the error term ε can take at most 5, 6 or 7 different values (when t and r vary). More precisely, we prove the following theorem.

Theorem 3.6. *Let q and α be two positive integers such that $q > \alpha$. Define $q_0 = q \bmod^{\pm}(\alpha)$ and $q_1 = \frac{q-q_0}{\alpha}$ (i.e., $q = q_1\alpha + q_0$ with $q_0 \in (-\alpha/2, \alpha/2]$). For any integers $t, r \in [0, q)$, the following property hold:*

$$\mathsf{Hb}(t+r) = \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r)) \cdot \alpha) + \varepsilon,$$

where ε belongs to one of the following sets

1. $\{-1, 0, 1, -q_1 + 1, q_1 - 2, q_1 - 1, q_1\}$, *if* $q_0 \in (-\alpha/2, -\alpha/4)$;
2. $\{-1, 0, 1, -q_1 + 1, q_1 - 1, q_1\}$, *if* $q_0 \in [-\alpha/4, 0]$;
3. $\{-1, 0, 1, -q_1, -q_1 + 1, q_1 - 1\}$, *if* $q_0 \in (0, \alpha/2)$;
4. $\{-1, 0, 1, -q_1, q_1\}$, *if* $q_0 = \alpha/2$.

In the case where $q \equiv 1 \pmod{\alpha}$, one obtains the following corollary.

Corollary 3.7. *Let q and α be two positive integers such that $q \equiv 1 \pmod{\alpha}$. Let $t, r \in [0, q)$, and $\varepsilon = \mathsf{Hb}(t+r) - \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r))\alpha)$. Then*

$$\varepsilon \in \{-1, 0, 1, -\frac{q-1}{\alpha}, -\frac{q-1}{\alpha} + 1, \frac{q-1}{\alpha} - 1\}.$$

Proof. This is a direct consequence of Theorem 3.6 and the fact that if $q \equiv 1 \pmod{\alpha}$, then $q_0 = 1$ and $q_1 = (q-1)/\alpha$. □

The rest of this section is devoted to the proof of Theorem 3.6. This proof being a bit technical, we will present it as a succession of smaller lemmas, in order to ease readability. Let us first fix some notations.

Definition 3.8. *Let $q > \alpha$ be integers, let $q_0, q_1 \in \mathbb{Z}$ be such that $q = q_0 + q_1\alpha$ with $q_0 \in (-\alpha/2, \alpha/2]$ (as in the theorem statement). Let t, r be any integers in $[0, q)$, and write them similarly as $t = t_0 + t_1\alpha$ and $r = r_0 + r_1\alpha$ for some t_0, t_1, r_0, r_1 integers with $t_0, r_0 \in (-\alpha/2, \alpha/2]$.*

Define $S = t + r \bmod^+(q)$, and let $\varepsilon_S \in \mathbb{Z}$ be such that $S = t + r - \varepsilon_S q$. Define also $X = (t_1 + r_1) \cdot \alpha \bmod^+(q)$, and let similarly $\varepsilon_X \in \mathbb{Z}$ be such that $X = (t_1 + r_1) \cdot \alpha - \varepsilon_X q$. Finally, let $S = S_0 + S_1\alpha$ and $X = X_0 + X_1\alpha$ with S_0, S_1, X_0, X_1 integers and $S_0, X_0 \in (-\alpha/2, \alpha/2]$.

We first give a lemma that motivates these definitions and notations.

Lemma 3.9. *Using the notations from Definition 3.8, we have $\mathsf{Hb}(t + r) = S_1$ and $\mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r)) \cdot \alpha) = X_1$.*

Proof. This follows from the definition of $\mathsf{Hb}(\cdot)$. Note that since t and r are in $[0, q)$, then $t = t \bmod^+(q)$ and $r = r \bmod^+(q)$, so that $t_1 = \mathsf{Hb}(t)$ and $r_1 = \mathsf{Hb}(r)$. Similarly, we have $X_1 = \mathsf{Hb}(X)$ and $S_1 = \mathsf{Hb}(S)$ (since X and S are in $[0, q)$). We conclude using the definition of $X = (t_1 + r_1) \cdot \alpha \bmod^+(q)$ and $S = t + r \bmod^+(q)$, and the fact that $\mathsf{Hb}(x) = \mathsf{Hb}(y)$ if $x = y \bmod q$. □

Using Lemma 3.9, we see that in order to prove Theorem 3.6, one needs to study the difference $S_1 - X_1$. Before giving an explicit description of this difference (depending on the value of q_0), we start by a few auxiliary lemmas. The first one below is a simple observation that the quantities t_1 and r_1 live in some bounded interval, a result which will be helpful multiple times during the proof.

Lemma 3.10. *We have $\max(0, \frac{t}{\alpha} - \frac{1}{2}) \leq t_1 < \frac{t}{\alpha} + \frac{1}{2}$ and similarly $\max(0, \frac{r}{\alpha} - \frac{1}{2}) \leq r_1 < \frac{r}{\alpha} + \frac{1}{2}$.*

Proof. Let us do the proof for t_1, the proof for r_1 is obtained similarly. First, note that $t_1 = \frac{t - t_0}{\alpha}$. Since $t_0 > -\alpha/2$ by assumption, then $-t_0/\alpha < 1/2$, and we obtain the desired upper bound $t_1 < t/\alpha + 1/2$. Using similarly the upper bound $t_0 \leq \alpha/2$, we obtain $t_1 \geq t/\alpha - 1/2$. For the bound $t_1 \geq 0$, observe that since $t_0 = t \bmod^{\pm}(\alpha)$ and $t \geq 0$, then we always have $t \geq t_0$. Hence we obtain $t_1 = \frac{t - t_0}{\alpha} \geq \frac{t_0 - t_0}{\alpha} = 0$. □

We now provide two lemmas that make explicit the values that can be taken by the quantities ε_S and ε_X.

Lemma 3.11. *We have*

$$\varepsilon_S = \begin{cases} 0 \text{ if } t + r < q, \\ 1 \text{ if } t + r \geq q. \end{cases}$$

Proof. Since t and r are both in $[0, q)$, then $t + r \in [0, 2q)$, and so we have only two possibilities: either $t + r \in [0, q)$, in which case $S = t + r \bmod^+(q) = t + r$ and $\varepsilon_S = 0$; or $t + r \in [q, 2q)$, in which case $S = t + r \bmod^+(q) = t + r - q$ and $\varepsilon_S = 1$. □

Lemma 3.12. *We have*

$$\varepsilon_X = \begin{cases} 0 \text{ if } t_1 + r_1 < q/\alpha, \\ 1 \text{ if } t_1 + r_1 \in [q/\alpha, 2q/\alpha), \\ 2 \text{ if } t_1 + r_1 \geq 2q/\alpha. \end{cases}$$

Proof. Using Lemma 3.10, we know that $t_1 + r_1 \geq 0$ and that $t_1 + r_1 < (r + t)/\alpha + 1$. Using the fact that r and t are both $< q$ and that $q/\alpha \geq 1$, we see that $t_1 + r_1 < 2q/\alpha + 1 \leq 3q/\alpha$. This implies that $(t_1 + r_1) \cdot \alpha \in [0, 3q)$.

From this and the definition of ε_X, we conclude that ε_X can take only 3 values. Namely $\varepsilon_X = 0$ if $(t_1 + r_1) \cdot \alpha \in [0, q)$, i.e., if $t_1 + r_1 < q/\alpha$; $\varepsilon_X = 1$ if $(t_1 + r_1) \cdot \alpha \in [q, 2q)$, i.e., if $t_1 + r_1 \in [q/\alpha, 2q/\alpha)$; and $\varepsilon_X = 2$ if $(t_1 + r_1) \cdot \alpha \in [2q, 3q)$, i.e., if $t_1 + r_1 \geq 2q/\alpha$. □

We note that the case $\varepsilon_X = 2$ is quite extreme, and cannot happen for some values of q_0, as stated in the following lemma.

Lemma 3.13. *If $\varepsilon_X = 2$, then $\varepsilon_S = 1$ and $q_0 \in (-\alpha/2, 0]$.*

Proof. Assume that $\varepsilon_X = 2$. Then from Lemma 3.12, we know that $t_1 + r_1 \geq 2q/\alpha$. Let us first prove that $\varepsilon_S = 1$. By Lemma 3.10, we know that $t_1 + r_1 < (t + r)/\alpha + 1$. Said differently, we have $t + r > (t_1 + r_1) \cdot \alpha - \alpha$. By the lower bound on $t_1 + r_1$, this implies that $t + r > 2q - \alpha \geq q$ (since $q \geq \alpha$). From Lemma 3.11, this then implies that $\varepsilon_S = 1$.

Let us now prove that $q_0 \in (-\alpha/2, 0]$. Since $t_1 + r_1 \geq 2q/\alpha$, then at least one of t_1 or r_1 must be $\geq q/\alpha$. Let us assume without loss of generality that $t_1 \geq q/\alpha$. From Lemma 3.10, we know that $t_1 < t/\alpha + 1/2 \leq q/\alpha + 1/2$. Moreover, t_1 is an integer. Combining everything, this means that the set $[q/\alpha, q/\alpha + 1/2) \cap \mathbb{Z}$ is non-empty (since it contains t_1). Recall that $q = q_0 + q_1 \cdot \alpha$, hence $q/\alpha = q_1 + q_0/\alpha$, with q_1 an integer. Since the set $[q_1 + q_0/\alpha, q_1 + q_0/\alpha + 1/2) \cap \mathbb{Z}$ is non empty, then it must be that $q_0/\alpha \in (-1/2, 0]$. This implies that $q_0 \in (-\alpha/2, 0]$. □

Now that we know what values can be taken by ε_X and ε_S, we will provide a collection of lemmas, one for each possible choice of the pair $(\varepsilon_X, \varepsilon_S)$, that describe the possible values taken by $X_1 - S_1$ for this choice of $(\varepsilon_X, \varepsilon_S)$.

Lemma 3.14. *If $\varepsilon_S = \varepsilon_X = 0$, then $X_1 - S_1 \in \{-1, 0, 1\}$.*

Proof. Since $\varepsilon_X = 0$, we know that $X = (t_1 + r_1) \cdot \alpha$. By the definition of X_1 and X_0, it must be that $X_1 = t_1 + r_1$ and $X_0 = 0$.

Let us now consider the case of S_1. Since $\varepsilon_S = 0$, then $S = t + r = (t_1 + r_1) \cdot \alpha + (t_0 + r_0)$. Moreover, $t_0 + r_0 \in (-\alpha, \alpha]$, by choice of t_0 and r_0. We now distinguish three cases.

1. If $t_0 + r_0 \in (-\alpha, -\alpha/2]$, then $S = (t_1 + r_1 - 1) \cdot \alpha + (t_0 + r_0 + \alpha)$ and $(t_0 + r_0 + \alpha) \in (0, \alpha/2] \subseteq (-\alpha/2, \alpha/2]$. From which we conclude that $S_0 = (t_0 + r_0 + \alpha)$ and $S_1 = t_1 + r_1 - 1$.
2. If $t_0 + r_0 \in (-\alpha/2, \alpha/2]$, then $S = (t_1 + r_1) \cdot \alpha + (t_0 + r_0)$ and $(t_0 + r_0) \in (-\alpha/2, \alpha/2]$. From which we conclude that $S_0 = (t_0 + r_0)$ and $S_1 = t_1 + r_1$.
3. If $t_0 + r_0 \in (\alpha/2, \alpha]$, then $S = (t_1 + r_1 + 1) \cdot \alpha + (t_0 + r_0 - \alpha)$ and $(t_0 + r_0 - \alpha) \in (-\alpha/2, 0] \subseteq (-\alpha/2, \alpha/2]$. From which we conclude that $S_0 = (t_0 + r_0 - \alpha)$ and $S_1 = t_1 + r_1 + 1$.

Combining the possible values for X_1 and S_1, we obtain as desired that $X_1 - S_1 \in \{-1, 0, 1\}$. □

Lemma 3.15. *If $\varepsilon_S = \varepsilon_X = 1$, then $X_1 - S_1 \in \{-1, 0, 1\}$.*

Proof. To simplify this proof, let us consider a slightly different decomposition of $q = q'_0 + q'_1 \cdot \alpha$ with $q'_0 \in [-\alpha/2, \alpha/2)$ (instead of $q_0 \in (-\alpha/2, \alpha/2]$ as above). Since $\varepsilon_X = 1$, we know that $X = (t_1 + r_1) \cdot \alpha - q = (t_1 + r_1 - q'_1) \cdot \alpha - q'_0$. By choice of q'_0, we know that $-q'_0 \in (-\alpha/2, \alpha/2]$, and so we obtain that $X_0 = -q'_0$ and that $X_1 = t_1 + r_1 - q'_1$.

Let us now consider the case of S_1. Since $\varepsilon_S = 1$, then $S = t + r - q = (t_1 + r_1 - q'_1) \cdot \alpha + (t_0 + r_0 - q'_0)$. Moreover, $t_0 + r_0 - q'_0 \in (-3\alpha/2, 3\alpha/2]$, by choice of t_0, r_0 and q'_0. We now distinguish three cases.

1. If $t_0 + r_0 - q'_0 \in (-3\alpha/2, -\alpha/2]$, then $S = (t_1 + r_1 - q'_1 - 1) \cdot \alpha + (t_0 + r_0 - q'_0 + \alpha)$ and $(t_0 + r_0 - q'_0 + \alpha) \in (-\alpha/2, \alpha/2]$. From which we conclude that $S_0 = (t_0 + r_0 - q'_0 + \alpha)$ and $S_1 = t_1 + r_1 - q'_1 - 1$.
2. If $t_0 + r_0 - q'_0 \in (-\alpha/2, \alpha/2]$, then $S = (t_1 + r_1 - q'_1) \cdot \alpha + (t_0 + r_0 - q'_0)$ and $(t_0 + r_0 - q'_0) \in (-\alpha/2, \alpha/2]$. From which we conclude that $S_0 = (t_0 + r_0 - q'_0)$ and $S_1 = t_1 + r_1 - q'_1$.
3. If $t_0 + r_0 - q'_0 \in (\alpha/2, 3\alpha/2]$, then $S = (t_1 + r_1 - q'_1 + 1) \cdot \alpha + (t_0 + r_0 - q'_0 - \alpha)$ and $(t_0 + r_0 - q'_0 - \alpha) \in (-\alpha/2, \alpha/2]$. From which we conclude that $S_0 = (t_0 + r_0 - q'_0 - \alpha)$ and $S_1 = t_1 + r_1 - q'_1 + 1$.

Combining the possible values for X_1 and S_1, we obtain as desired that $X_1 - S_1 \in \{-1, 0, 1\}$. □

Lemma 3.16. *If $\varepsilon_S = 0$ and $\varepsilon_X = 1$, then*

$$X_1 - S_1 \in \begin{cases} \{-q_1, -q_1 + 1\} & \text{if } q_0 \in (-\alpha/2, 0] \\ \{-q_1 + 1\} & \text{if } q_0 \in (0, \alpha/2) \\ \{-q_1\} & \text{if } q_0 = \alpha/2 \end{cases}$$

Proof. In this case, we know that $t + r < q$ (since $\varepsilon_S = 0$) and $t_1 + r_1 \geq q/\alpha$ (since $\varepsilon_X = 1$). Moreover, by Lemma 3.10, we also know that $t_1 + r_1 < (t + r)/\alpha + 1 < q/\alpha + 1$. This means that $t_1 + r_1$ is an integer in the interval $[q/\alpha, q/\alpha + 1)$. There is a unique possibility for $t_1 + r_1$, which is $t_1 + r_1 = \lceil q/\alpha \rceil$. We now make a remark that will be useful later: if $q_0 \in (0, \alpha/2]$, then $t_1 + r_1 \geq q/\alpha + 1/2$. Indeed, we know that $q/\alpha = q_1 + q_0/\alpha$ and, if $q_0 \in (0, \alpha/2]$, then we have that

$q_0/\alpha \in (0, 1/2]$, so the rounding up operation increases the quantity by at least $1/2$.

Regarding X, since $\varepsilon_X = 1$, we have that $X = (t_1+r_1)\cdot\alpha - q = (t_1+r_1-q_1)\cdot\alpha - q_0$. If $q_0 \in (-\alpha/2, \alpha/2)$, then $X_0 = -q_0$ and $X_1 = t_1 + r_1 - q_1$. If $q_0 = \alpha/2$, then $X_0 = \alpha/2 = \alpha - q_0$ and $X_1 = t_1 + r_1 - q_1 - 1$.

Regarding S, since $\varepsilon_S = 0$, we have that $S = t + r = (t_1 + r_1) \cdot \alpha + (t_0 + r_0)$. Notice that $t_0 + r_0 < 0$. Indeed, we have $t_0 + r_0 = t + r - (t_1 + r_1) \cdot \alpha < q - q = 0$, where we used the fact that $t + r < q$ and $t_1 + r_1 \geq q/\alpha$. This implies that we have either $S_1 = t_1 + r_1$ (if $t_0 + r_0 \in (-\alpha/2, 0)$) or $S_1 = t_1 + r_1 - 1$ (if $t_0 + r_0 \in (-\alpha, -\alpha/2]$).

Let us now consider the three cases from the statement, depending on the value of q_0.

1. If $q_0 \in (-\alpha/2, 0]$, then from the previous discussion we have $X_1 = t_1 + r_1 - q_1$ and $S_1 = t_1 + r_1$ or $S_1 = t_1 + r_1 - 1$. This implies that $X_1 - S_1$ belongs to $\{-q_1, -q_1 + 1\}$, as desired.
2. If $q_0 \in (0, \alpha/2)$, then, from the previous discussion we have $X_1 = t_1 + r_1 - q_1$. Let us be a little bit finer for the value of S_1. We have seen at the beginning of the proof that when $q_0 \in (0, \alpha/2]$, then $t_1 + r_1 \geq q/\alpha + 1/2$. This implies that

$$t_0 + r_0 = t + r - (t_1 + r_1) \cdot \alpha < q - (q/\alpha + 1/2) \cdot \alpha = -\alpha/2.$$

   From this, we conclude that $S_1 = t_1 + r_1 - 1$ is the only option for S_1 (see discussion above). Hence, we see that $X_1 - S_1 = -q_1 + 1$ as desired.
3. If $q_0 = \alpha/2$, then, from the previous discussion we have $X_1 = t_1 + r_1 - q_1 - 1$. The computation for S_1 is the same as in the case $q_0 \in (0, \alpha/2)$, so we have $S_1 = t_1 + r_1 - 1$. We conclude that $X_1 - S_1$ is always equal to $-q_1$. This concludes the proof.

□

Lemma 3.17. *If $\varepsilon_S = 1$ and $\varepsilon_X = 0$, then*

$$X_1 - S_1 \in \begin{cases} \{q_1 - 1\} & \text{if } q_0 \in (-\alpha/2, 0] \\ \{q_1, q_1 - 1\} & \text{if } q_0 \in (0, \alpha/2) \\ \{q_1\} & \text{if } q_0 = \alpha/2 \end{cases}$$

Proof. In this case, we know that $t+r \geq q$ (since $\varepsilon_S = 1$) and $t_1+r_1 < q/\alpha$ (since $\varepsilon_X = 0$). Moreover, by Lemma 3.10, we also know that $t_1 + r_1 \geq (t+r)/\alpha - 1 \geq q/\alpha - 1$. This means that $t_1 + r_1$ is an integer in the interval $[q/\alpha - 1, q/\alpha)$. There is a unique possibility for $t_1 + r_1$, which is $t_1 + r_1 = \lceil q/\alpha - 1 \rceil$. We now make a remark that will be useful later: if $q_0 \in (-\alpha/2, 0]$, then $t_1 + r_1 < q/\alpha - 1/2$. Indeed, we know that $q/\alpha - 1 = q_1 - 1 + q_0/\alpha$ and, if $q_0 \in (-\alpha/2, 0]$, then we have $q_0/\alpha \in (-1/2, 0]$, so the rounding up operation increases the quantity by $< 1/2$.

Regarding X, since $\varepsilon_X = 0$, we have that $X = (t_1+r_1)\cdot\alpha$, and so $X_1 = t_1 + r_1$.

Regarding S, since $\varepsilon_S = 1$, we have that $S = t + r - q = (t_1 + r_1 - q_1) \cdot \alpha + (t_0 + r_0 - q_0)$. Notice that $t_0 + r_0 - q_0 > -\alpha/2$. Indeed, we have $(t_0 + r_0) - q_0 = t + r - (t_1 + r_1) \cdot \alpha - q_0 > q - q - \alpha/2 = -\alpha/2$, where we used the fact that $t + r \geq q$, that $t_1 + r_1 < q/\alpha$, and that $q_0 \leq \alpha/2$. This implies that we have either $S_1 = t_1 + r_1 - q_1$ (if $t_0 + r_0 - q_0 \in (-\alpha/2, \alpha/2]$) or $S_1 = t_1 + r_1 - q_1 + 1$ (if $t_0 + r_0 - q_0 \in (\alpha/2, 3\alpha/2]$).

Let us now consider the three cases from the statement, depending on the value of q_0.

1. If $q_0 \in (0, \alpha/2)$, then from the previous discussion we have $X_1 = t_1 + r_1$ and $S_1 = t_1 + r_1 - q_1$ or $S_1 = t_1 + r_1 - q_1 + 1$. This implies that $X_1 - S_1$ belongs to $\{q_1, q_1 - 1\}$, as desired.
2. If $q_0 \in (-\alpha/2, 0]$, we still have $X_1 = t_1 + r_1$. Let us be a little bit finer for the value of S_1. We have seen at the beginning of the proof that when $q_0 \in (-\alpha/2, 0]$, then $t_1 + r_1 < q/\alpha - 1/2$. This implies that

$$t_0 + r_0 - q_0 \geq t_0 + r_0 = t + r - (t_1 + r_1) \cdot \alpha > q - (q/\alpha - 1/2) \cdot \alpha = \alpha/2.$$

   From this, we conclude that $S_1 = t_1 + r_1 - q_1 + 1$ is the only option for S_1 (see discussion above). Hence, we see that $X_1 - S_1 = q_1 - 1$ as desired.
3. If $q_0 = \alpha/2$, we still have $X_1 = t_1 + r_1$. Moreover, in this case, we have $(t_0 + r_0) - q_0 \leq \alpha/2 + \alpha/2 - \alpha/2 = \alpha/2$. From this, we conclude that $S_1 = t_1 + r_1 - q_1$, and we obtain $X_1 - S_1 = q_1$. This concludes the proof.

□

Lemma 3.18. *If $\varepsilon_X = 2$, then*

$$X_1 - S_1 \in \begin{cases} \{-q_1 + 1, -q_1 + 2\} & \text{if } q_0 \in (-\alpha/2, -\alpha/4) \\ \{-q_1, -q_1 + 1\} & \text{if } q_0 \in [-\alpha/4, 0] \end{cases}$$

Proof. First of all, let us recall that from Lemma 3.13, when $\varepsilon_X = 2$, then it must be that $\varepsilon_S = 1$ and that $q_0 \in (-\alpha/2, 0]$. So we only have to consider these cases for the choice of q_0.

Let us first prove that $t_1 + r_1 = 2q_1$. To see this, observe that, by assumption, we have $t_1 + r_1 \geq 2q/\alpha$ (because $\varepsilon_X = 2$), and $t_1 + r_1 < \frac{t+r}{\alpha} + 1 \leq 2q/\alpha + 1$ by Lemma 3.10. So $t_1 + r_1$ is the unique integer in the interval $[2q/\alpha, 2q/\alpha + 1)$, i.e., it is equal to $\lceil 2q/\alpha \rceil$. But $\frac{2q}{\alpha} = 2q_1 + \frac{2q_0}{\alpha}$. Since $q_0 \in (-\alpha/2, 0]$ by assumption, then $\frac{2q_0}{\alpha} \in (-1, 0]$, from which we conclude that $t_1 + r_1 = \lceil 2q_1 + \frac{2q_0}{\alpha} \rceil = 2q_1$, as desired.

Let us use this observation to compute X_1. We have, that $X = (t_1 + r_1) \cdot \alpha - 2q$ since $\varepsilon_X = 2$. Using the formula for $t_1 + r_1$, we obtain $X = 2q_1\alpha - 2q = -2q_0$. If $q_0 \in [-\alpha/4, 0]$, then $-2q_0 \in [0, \alpha/2]$, and we conclude that $X_1 = 0$. On the other hand, if $q_0 \in (-\alpha/2, -\alpha/4)$, then we have $-2q_0 \in (\alpha/2, \alpha)$, and so we have $X_1 = 1$.

Regarding S_1, recall that $\varepsilon_S = 1$, so that $S = t + r - q = (t_1 + r_1 - q_1) \cdot \alpha + (t_0 + r_0 - q_0) = q_1\alpha + (t_0 + r_0 - q_0)$, where we used that $t_1 + r_1 = 2q_1$. Moreover,

$$t_0 + r_0 = t + r - (t_1 + r_1)\alpha = t + r - 2q_1 \cdot \alpha < 2q - 2q_1\alpha = 2q_0.$$

From this, we see that $t_0 + r_0 - q_0 < q_0 \leq 0$. Hence, we have either $S_1 = q_1$ if $t_0 + r_0 - q_0 \in (-\alpha/2, 0]$, or $S_1 = q_1 - 1$ if $t_0 + r_0 - q_0 \in (-3\alpha/2, -\alpha/2]$.

Overall, we see that $X_1 - S_1$ belongs either to the set $\{-q_1, -q_1 + 1\}$ if $q_0 \in [-\alpha/4, 0]$ (i.e., $X_1 = 0$), or to the set $\{-q_1+1, -q_1+2\}$ if $q_0 \in (-\alpha/2, -\alpha/4)$ (i.e., $X_1 = 1$). □

Combining all these lemmas, we can finally prove Theorem 3.6.

Proof of Theorem 3.6. Let α, q, q_0 and q_1 be as in the theorem statement. Let us fix two integers $t, r \in [0, q)$. From Lemma 3.9, we know that

$$\varepsilon = \mathsf{Hb}(t + r) - \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r))\alpha) = S_1 - X_1.$$

The result then follows by inspection of Lemmas 3.14, 3.15, 3.16, 3.17, and 3.18. Note that these lemmas indeed cover all the possible cases, thanks to Lemma 3.11, 3.12, and 3.13. □

Numerical experiments. We ran some numerical experiments to verify the correctness of Theorem 3.6. For each of the four possible interval choices for q_0, we generated 10 random pairs (α, q) with q_0 in the desired interval. We then verified experimentally that for these pairs, the values of ε achieved by running over all pairs (t, r) in $[0, q) \times [0, q)$ were all contained in the sets described in Theorem 3.6. All the 40 tests were successful. The code used to perform these tests is available at `https://github.com/funibus/africacrypt-2026-supplementary-material.git`.

4 Applications

In this section, we show how one can use our results to construct an efficient signature scheme with rerandomizable keys [FKM+16] on top of ML-DSA [Nat24a], a recently standardized lattice-based signature scheme by NIST. Signature schemes with rerandomizable keys are cryptographic primitives that can be used in various applications such as deterministic wallets [DFL19] for cryptocurrencies, Tor hidden services [The25], etc. Before describing our construction, we will first recall the definition of signature schemes and signature schemes with rerandomizable keys, and the ML-DSA algorithms.

4.1 Signature schemes with rerandomizable keys

Definition 4.1 (signature scheme). *A signature scheme Σ is a tuple of algorithms* $(\Sigma.\mathsf{KGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ *defined as follows.*

- $\Sigma.\mathsf{KGen}(1^\lambda) \to (pk, sk)$*: is a randomized key generation algorithm which takes as input a security parameter* 1^λ *and outputs a key pair* (pk, sk) *with* pk *a public verification key and* sk *a secret signing key.*

- $\Sigma.\mathsf{Sign}(sk, m) \to \sigma$*: is a randomized (or deterministic) signing algorithm to sign a message* m *using* sk *and, producing a signature* σ *of* m*.*
- $\Sigma.\mathsf{Verify}(pk, m, \sigma) \to m$*: is a deterministic verification algorithm that takes as input a message* m*, a signature* σ *and a public key* pk*, and returns* 1 *(valid) or* 0 *(invalid).*

Definition 4.2 (Correctness of a signature scheme). *A signature scheme* $\Sigma := (\Sigma.\mathsf{KGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ *is correct if the following condition holds for all* 1^λ*:*

$$\Pr\left[\Sigma.\mathsf{Verify}\left(pk, m, \Sigma.\mathsf{Sign}(sk, m)\right) = 1 \,\middle|\, \begin{array}{l} m \leftarrow \{0,1\}^*, \\ (pk, sk) \leftarrow \Sigma.\mathsf{KGen}(1^\lambda) \end{array}\right] = 1$$

Definition 4.3 (signature scheme with rerandomizable keys [FKM+16]). *A signature scheme with rerandomizable keys (SWRK)* Σ_{RK} *is a tuple of algorithms* $(\Sigma_{\mathsf{RK}}.\mathsf{KGen}, \Sigma_{\mathsf{RK}}.\mathsf{Sign}, \Sigma_{\mathsf{RK}}.\mathsf{Verify}, \Sigma_{\mathsf{RK}}.\mathsf{RandSk}, \Sigma_{\mathsf{RK}}.\mathsf{RandPk})$ *where* $(\Sigma_{\mathsf{RK}}.\mathsf{KGen}, \Sigma_{\mathsf{RK}}.\mathsf{Sign}, \Sigma_{\mathsf{RK}}.\mathsf{Verify})$ *are the standard algorithms of a signature scheme and the two others algorithms are defined over a randomness space* $\mathcal{R}$ *as follows.*

- $\Sigma_{\mathsf{RK}}.\mathsf{RandSk}(sk, \rho) \to sk'$*: is a deterministic algorithm to update a private key which accepts as input* sk *and a randomness* $\rho \in \mathcal{R}$*, and produces as output a rerandomized secret key* sk'*.*
- $\Sigma_{\mathsf{RK}}.\mathsf{RandPk}(pk, \rho) \to pk'$*: is a deterministic algorithm to update a public key which accepts as input* pk *and a randomness* $\rho \in \mathcal{R}$*, and produces as output a rerandomized public key* pk'*.*

Definition 4.4 (Correctness of Σ_{RK}). *A signature scheme with rerandomizable keys* $\Sigma_{RK} = (\Sigma_{\mathsf{RK}}.\mathsf{KGen}, \Sigma_{\mathsf{RK}}.\mathsf{Sign}, \Sigma_{\mathsf{RK}}.\mathsf{Verify}, \Sigma_{\mathsf{RK}}.\mathsf{RandSk}, \Sigma_{\mathsf{RK}}.\mathsf{RandPk})$ *is correct if the following two conditions holds:*

$$\Pr\left[\Sigma_{\mathsf{RK}}.\mathsf{Verify}\left(pk, m, \Sigma_{\mathsf{RK}}.\mathsf{Sign}(sk, m)\right) = 1 \,\middle|\, \begin{array}{l} m \leftarrow \{0,1\}^*, \\ (pk, sk) \leftarrow \Sigma_{\mathsf{RK}}.\mathsf{KGen}(1^\lambda) \end{array}\right] = 1$$

and for all $(pk, sk) \leftarrow \Sigma_{\mathsf{RK}}.\mathsf{KGen}(1^\lambda)$

$$\mathbb{P}\left[\Sigma_{\mathsf{RK}}.\mathsf{Verify}\left(pk', m, \Sigma_{\mathsf{RK}}.\mathsf{Sign}(sk', m)\right) = 1 \,\middle|\, \begin{array}{l} m \leftarrow \{0,1\}^*, \rho \leftarrow \mathcal{R} \\ pk' := \Sigma_{\mathsf{RK}}.\mathsf{RandPk}(pk, \rho) \\ sk' := \Sigma_{\mathsf{RK}}.\mathsf{RandSk}(sk, \rho) \end{array}\right] = 1$$

4.2 ML-DSA Algorithms

ML-DSA [Nat24a] is a lattice-based signature scheme that was recently standardized by NIST as FIPS 204. It is derived from the Dilithium signature scheme [BDK+21; DKL+18], and its security is related to the Module Learning With Errors (MLWE) problem. Let us recall the parameters and functions used in ML-DSA algorithms.

Parameters and Sets : In this section, we suppose that q is a prime such that $q \equiv 1 \pmod{\alpha}$ with $\alpha = 2\gamma_2 = 2^d$ for some positive integers γ_2 and d,

where d denotes the number of dropped bits in the public key. We also denote by $\mathcal{R} = \mathbb{Z}[X]/(X^{256}+1)$ and the quotient ring $\mathcal{R}_q = \mathcal{R}/(q\mathcal{R})$. For an element $y \in \mathcal{R}^k$ we denote by $\| y \|_\infty$ the infinity norm of the $256k-$dimensional vector of the coefficients of the polynomials of y. The scheme also uses an eXtendable-Output Function H instantiated by SHAKE-256.

Supporting algorithms : In addition to the function Hb and $\mathsf{Power2Round}$, the signature ML-DSA uses the following algorithms:

- $\mathsf{ExpandA}(\rho)$: is a deterministic algorithm that takes as input a seed ρ and outputs a matrix $\mathbf{A} \in \mathcal{R}_q^{k \times l}$.
- $\mathsf{ExpandS}(\rho')$: is a deterministic algorithm that takes as input a seed ρ' and outputs a couple of vectors $(s_1, s_2) \in \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$ each with polynomial coordinates whose coefficients are in the range $[-\eta, \eta]$.
- $\mathsf{HighBits}(r, \alpha)$: returns the corrected high bits of r i.e. r_1 if $r_1 \neq (q-q_0)/\alpha$ and 0 otherwise, where $r_1 = \mathsf{Hb}_{q,\alpha}(r)$ and $q_0 = q \bmod^{\pm}(\alpha)$. If the parameters q and α are clear from the context, we simply denote this function by $\mathsf{HighBits}(r)$.
- $\mathsf{LowBits}(r, \alpha)$: returns the corrected low bits of r i.e. r_0 if $r_1 \neq (q-q_0)/\alpha$ and $r_0 - 1$ otherwise. We denote this function by $\mathsf{LowBits}(r)$ if the parameters q and α are clear from the context.
- $\mathsf{MakeHint}(z, r)$: returns a hint $h \in \{0,1\}$ to recover the high bits of $z + r$ from the high bits r.
- $\mathsf{UseHint}(h, r)$: returns the high bits of $r + z$ using the binary hint h and the high bits of r.
- $\mathsf{ExpandMask}(\rho, \kappa)$: outputs a vector $y \in \mathcal{R}^l$ such that each polynomial of the components y has coefficients between $-\gamma_1 + 1$ and γ_1.
- $\mathsf{SampleInBall}(\rho)$: is a deterministic algorithm that takes as input a seed ρ and outputs a polynomial $c \in \mathcal{R}$ such that the coefficients of c are in $\{-1, 0, 1\}$ and their hamming weight τ is at most 60.
- $\mathsf{pkEncode}(\rho, t_1)$: encodes the public key components for ML-DSA into a bit string.
- $\mathsf{pkDecode}(pk)$: decodes the public key bit string back into its components.
- $\mathsf{skEncode}(\rho, K, tr, s_1, s_2, t_0)$: encodes the secret key components for ML-DSA into a bit string.
- $\mathsf{skDecode}(sk)$: decodes the secret key bit string back into its components.
- $\mathsf{sigEncode}(c, z, h)$: encodes the signature components for ML-DSA into a bit string.
- $\mathsf{sigDecode}(\sigma)$: decodes the signature bit string back into its components.

We recall the key generation, signing and verification algorithms of ML-DSA in the following. We refer the reader to [Nat24a] for more details about the scheme and its security analysis.

ML-DSA.KGen($pp = \zeta \in \{0,1\}^{256}$) :

$(\rho, \rho', K) := \mathsf{H}(\zeta)$;
$(s_1, s_2) \in \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k := \mathsf{ExpandS}(\rho')$;
$\mathbf{A} \sim \mathcal{R}_q^{k \times l} := \mathsf{ExpandA}(\rho)$;
$t := \mathbf{A}s_1 + s_2$;
$(t_1, t_0) := \mathsf{Power2Round}(t)$;
$tr := \mathsf{H}(\rho, t_1, 64)$;
$pk := \mathsf{pkEncode}(\rho, t_1)$;
$sk := \mathsf{skEncode}(\rho, K, tr, s_1, s_2, t_0)$;
return (pk, sk)

ML-DSA.Verify(m, pk, σ) :

$\mathbf{A} \sim \mathcal{R}_q^{k \times l} := \mathsf{ExpandA}(\rho)$;
$(\rho, t_1) := \mathsf{pkDecode}(pk)$;
$(c, z, h) := \mathsf{sigDecode}(\sigma)$;
$tr := \mathsf{H}(\rho, t_1)$; $\mu := \mathsf{H}(tr, m)$;
$c := \mathsf{SampleInBall}(c)$;
$w_1' := \mathsf{UseHint}(h, \mathbf{A}z - ct_1 \cdot \alpha)$;
$b_1 := [\| z \|_\infty \leq \gamma_1 - \beta]$;
$b_2 := [c = \mathsf{H}(\mu, w_1')]$;
$b_3 := [|h|_{h_j=1} \leq \omega]$;
return $b_1 \wedge b_2 \wedge b_3$

ML-DSA.Sign(m, sk) :

$\mathbf{A} \sim \mathcal{R}_q^{k \times l} := \mathsf{ExpandA}(\rho)$;
$(\rho, K, tr, s_1, s_2, t_0) := \mathsf{skDecode}(sk)$;
$\mu := \mathsf{H}(tr, m)$;
$\kappa := 0$; $(z, h) :=\perp$;
$\rho' := \mathsf{H}(K, \mu)$; ▷ (or $\rho' \leftarrow \{0,1\}^{512}$)
while $(z, h) =\perp$ **do**
  $y \in \mathcal{S}_{\gamma_1 - 1}^l := \mathsf{ExpandMask}(\rho', \kappa)$;
  $w := \mathbf{A}y$;
  $w_1 := \mathsf{HighBits}(w)$;
  $c :=\in \mathsf{H}(\mu, w_1)$;
  $c \in B_\tau := \mathsf{SampleInBall}(c)$;
  $z := y + cs_1$;
  $r_0 := \mathsf{LowBits}(w - cs_2)$;
  if $\| z \|_\infty \geq \gamma_1 - \beta$ or $\| r_0 \|_\infty \geq \gamma_2 - \beta$ **then**
    $(z, h) :=\perp$;
  else
    $h := \mathsf{MakeHint}(-ct_0, w - cs_2 + ct_0)$;
    if $\| ct_0 \|_\infty > \gamma_2$ or $|h|_{h_j=1} > \omega$ **then**
      $(z, h) :=\perp$;
    $\kappa := \kappa + 1$
return $\sigma = \mathsf{sigEncode}(c, z, h)$

Fig. 1: ML-DSA: key generation, signature and verification algorithms.

4.3 A SWRK from ML-DSA

By Corollary 3.7, we have $\mathsf{Hb}(t + r) = \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r)) \cdot \alpha) + t^h$ with $t^h \in \{-1, 0, 1, -q_1, -q_1 + 1, q_1 - 1\}$ for any $t, r \in [0, q)$. We will use this result to construct an efficient lattice-based signature scheme with rerandomizable keys on top of ML-DSA. Our construction is similar to DilithiumRKC proposed by Seck and Roux-Langlois [SR25], but for the sake of completeness, we will give the details of the different algorithms of our scheme. We first describe the encoding and decoding algorithms for the hint t^h in the following. Notice that the hint t^h can be encoded with 3 bits since it takes at most 6 different values.

$\underline{\mathsf{hintEncode}(t^h, q, \alpha) :}$

1: $q_0 := q \bmod^{\pm}(\alpha)$;
2: $q_1 = (q - q_0)/\alpha$;
3: **If** $t^h = -1$: **return** 010;
4: **If** $t^h = 0$: **return** 000;
5: **If** $t^h = 1$: **return** 001;
6: **If** $t^h = -q_1 + 1$: **return** 011;
7: **If** $t^h = -q_1$: **return** 100;
8: **If** $t^h = q_1 - 1$: **return** 101;

$\underline{\mathsf{hintDecode}(\mathsf{hint}, q, \alpha) :}$

1: $q_0 := q \bmod^{\pm}(\alpha)$;
2: $q_1 = (q - q_0)/\alpha$;
3: **If** hint $=$ 010 : **return** -1;
4: **If** hint $=$ 000 : **return** 0;
5: **If** hint $=$ 001 : **return** 1;
6: **If** hint $=$ 011 : **return** $-q_1 + 1$;
7: **If** hint $=$ 100 : **return** $-q_1$;
8: **If** hint $=$ 101 : **return** $q_1 - 1$;

Fig. 2: Encoding and decoding algorithms for the hint t^h

Now, let us describe the encoding and decoding functions for ML-DSA-RK.

- $\mathsf{pkEncode}'(\rho, t_1')$: outputs a bit string encoding the public key for ML-DSA-RK.
- $\mathsf{pkDecode}'(pk)$: outputs the components ρ and t_1' of a public key for ML-DSA-RK.
- $\mathsf{skEncode}'(\rho, K, tr, s_1, s_2, t_0, t^h)$: outputs a bit string encoding the secret key for ML-DSA-RK. Notice that

$$\mathsf{skEncode}'(\rho, K, tr, s_1, s_2, t_0, t^h) := \mathsf{skEncode}(\rho, K, tr, s_1, s_2, t_0) \| \mathsf{hintEncode}(t^h, q, \alpha)$$

  where skEncode is the secret key encoding function of ML-DSA.
- $\mathsf{skDecode}'(sk)$: outputs the components ρ, K, tr, s_1, s_2, t_0 and t^h of a secret key for ML-DSA-RK.
- $\mathsf{sigEncode}'(c, z, h, t^h)$: outputs a bit string encoding the signature for ML-DSA-RK. Notice that

$$\mathsf{sigEncode}'(c, z, h, t^h) := \mathsf{sigEncode}(c, z, h) \| \mathsf{hintEncode}(t^h, q, \alpha)$$

  where sigEncode is the signature encoding function of ML-DSA.
- $\mathsf{sigDecode}'(\sigma)$: outputs the components c, z, h and t^h of a signature for ML-DSA-RK.

In the following, we define the randomized algorithms RandPk and RandSk for our SWRK for updating the public and private keys.

$\underline{\mathsf{RandSk}(sk, \mu) :}$
1: $(\rho, K, tr, s_1, s_2, t_0, _) := \mathsf{skDecode}(sk)$;
2: $\mathbf{A} \sim R_q^{k \times l} := \mathsf{ExpandA}(\rho)$;
3: $t_1 := \mathsf{Hb}(\mathbf{A}s_1 + s_2)$;
4: $\rho_\mu := \mathsf{H}(\rho, t_1, \mu)$;
5: $(s_1', s_2') := \mathsf{ExpandS}(\rho_\mu)$;
6: $r := \mathbf{A}s_1' + s_2'$; $r_1 := \mathsf{Hb}(r)$;
7: $s_1^+ := s_1 + s_1'$; $s_2^+ := s_2 + s_2'$;
8: $t^+ := \mathbf{A}s_1^+ + s_2^+$;
9: $(t_1^+, t_0^+) := \mathsf{Power2Round}(t^+)$;
10: $tr' := \mathsf{H}(\rho, t_1^+, 64)$;
11: $t^h := t_1^+ - \mathsf{Hb}((t_1 + r_1) \cdot \alpha)$;
12: $sk' := \mathsf{skEncode}'(\rho, K, tr', s_1^+, s_2^+, t_0^+, t^h)$;
13: **return** sk'

$\underline{\mathsf{RandPk}(pk, \mu) :}$
1: $(\rho, t_1) := \mathsf{pkDecode}(pk)$;
2: $\mathbf{A} \sim R_q^{k \times l} := \mathsf{ExpandA}(\rho)$;
3: $\rho_\mu := \mathsf{H}(\rho, t_1, \mu)$;
4: $(s_1', s_2') := \mathsf{ExpandS}(\rho_\mu)$;
5: $r := \mathbf{A}s_1' + s_2'$;
6: $r_1 := \mathsf{Hb}(r)$;
7: $t_1' := \mathsf{Hb}((t_1 + r_1) \cdot \alpha)$;
8: $pk' := \mathsf{pkEncode}'(\rho, t_1')$;
9: **return** pk'

$\underline{\mathsf{KGen}(\zeta \in \{0,1\}^{256}) :}$
1: $(pk, \widehat{sk}) := \mathsf{ML\text{-}DSA.KGen}(\zeta)$;
2: $t^h := 0 \in \mathcal{R}_q^k$; $sk := (\widehat{sk}, t^h)$;
3: **return** (pk, sk)

Fig. 3: Key generation, public and private keys rerandomization algorithms of ML-DSA-RK.

Remark 4.5. Notice that the value $pk' := (\rho, t_1')$ obtained from the randomized public key algorithm is not the correct public key corresponding to the rerandomized secret key $sk' := (\rho, K, tr', s_1^+, s_2^+, t_0^+, t^h)$. Concretely, the correct public key corresponding to sk' is $(\rho, t_1' + t^h)$. However, the hint t^h cannot be publicly computed from pk and the randomness μ since it depends on the secret elements s_1 and s_2. To handle this problem, the hint t^h which is included in the rerandomized secret key sk' is encoded using the hintEncode algorithm and will be included as a part of the signature. During the verification, the hint will be decoded using the hintDecode algorithm and added to t_1' to obtain the correct public key corresponding to sk'.

Now let us describe the signing and verification algorithms of our SWRK.

$\underline{\mathsf{ML\text{-}DSA\text{-}RK.Sign}(m, sk) :}$
1: $(\rho, K, tr, s_1, s_2, t_0, t^h) := \mathsf{skDecode}'(sk)$;
2: $sk' := \mathsf{skEncode}(\rho, K, tr, s_1, s_2, t_0)$;
3: $(c, z, h) := \mathsf{ML\text{-}DSA.Sign}(m, sk')$;
4: $h' := (h, t^h)$;
5: $\sigma := \mathsf{sigEncode}'(c, z, h')$;
6: **return** σ

$\underline{\mathsf{ML\text{-}DSA\text{-}RK.Verify}(m, \sigma, pk) :}$
1: $(\rho, t_1) := \mathsf{pkDecode}'(pk)$;
2: $(c, z, (h, t^h)) := \mathsf{sigDecode}'(\sigma)$;
3: $\sigma' := \mathsf{sigEncode}(c, z, h)$;
4: $pk' := \mathsf{pkEncode}'(\rho, t_1 + t^h)$;
5: $b := \mathsf{ML\text{-}DSA.Verify}(m, \sigma', pk')$;
6: **return** b

Fig. 4: Signing and verification algorithms of ML-DSA-RK constructed from the sign and verify algorithms of ML-DSA.

Impact of the Hint on Security

After updating the secret key using the RandSk algorithm, the hint t^h is included in the signature and becomes publicly available. The hint t^h is computed using MLWE samples derived from the secret key and the randomness used for rerandomization, i.e.:

$$t^h = \mathsf{Hb}(t + r) - \mathsf{Hb}((\mathsf{Hb}(t) + \mathsf{Hb}(r)) \cdot \alpha)$$

where $t = \mathbf{A}s_1 + s_2$ is a master public key component computed from master secret elements s_1 and s_2, and r is an MLWE sample computed from the randomness used for rerandomization.

One may wonder whether the hint t^h could leak information about the secret key, or be used by an adversary to forge signatures or break the unlinkability property. Unlinkability [ELW23; CGH+25] ensures that randomized public keys cannot be linked to the master public keys.

- **Impact of t^h on secret key recovery:** Assuming the hardness of the MLWE problem, an adversary cannot recover any information about the secret elements s_1 and s_2 when given as input the element t. Therefore, the hint t^h, which is computed from t and an independent randomness, cannot leak any information about the secret key.

- **Impact of t^h on signature forgery:** In the unforgeability game (see [SR25, Fig. 2]), the adversary is given a master public key pk (and thus t) and is asked to find a valid randomness r, a message $m^\star$, and a signature $\sigma^\star$ such that $\mathsf{ML\text{-}DSA\text{-}RK.Verify}(m^\star, \sigma^\star, \mathsf{RandPk}(pk, r)) = 1$. Since t^h is computed from t and r, both of which are accessible to the adversary, this hint does not provide any additional advantage for forging signatures. Moreover, t is considered to be public in the security analysis of $\mathsf{ML\text{-}DSA}$. Hence, without loss of generality, we can assume t is publicly available in the security analysis of $\mathsf{ML\text{-}DSA\text{-}RK}$. In this case, it is trivial to see that no further information is learned from t^h.

- **Impact of t^h on unlinkability:** In the game of unlinkability, the adversary is given two session public keys pk_r and $pk_{\hat{r}}$ and is asked to guess whether they are derived from the same master public key or not. The adversary can also ask for signatures on messages of its choice that verify under the session public keys. Then the adversary has access to the hints t^h and $\hat{t}^h$ corresponding to the two session public keys. Suppose that the randomness r and $\hat{r}$ used for deriving the two session public keys are sampled uniformly at random over the randomness space and are not known to the adversary. Then the distributions of the hints t^h and $\hat{t}^h$ computed from (t, r) (i.e. $t^h = \mathsf{Hb}(t+r) - \mathsf{Hb}((\mathsf{Hb}(t)+\mathsf{Hb}(r))\cdot\alpha)$) and $(\hat{t}, \hat{r})$ (i.e. $\hat{t}^h = \mathsf{Hb}(\hat{t}+\hat{r}) - \mathsf{Hb}((\mathsf{Hb}(\hat{t})+\mathsf{Hb}(\hat{r}))\cdot\alpha)$) respectively are indistinguishable even if t and $\hat{t}$ are the same. Therefore, the hints t^h and $\hat{t}^h$ do not leak any additional advantage to an adversary attempting to break the unlinkability property.

Remark 4.6. The security analysis of a signature scheme with rerandomizable keys built on top of a simplified version of ML-DSA was presented in [SR25]. We refer the reader to [SR25] for further details.

References

[BCG+24] Daniel J. Bernstein, Jolijn Cottaar, Emanuele Di Giandomenico, Kathrin Hövelmanns, Andreas Hülsing, Mikhail Kudinov, Tanja Lange, Mairon Mahzoun, Matthias Meijers, Alex Pellegrini, Alberto Ravagnani, Silvia Ritsch, Sven Schäge, Tianxin Tang, Monika Trimoska, Marc Vorstermans, and Fiona Johanna Weber. *Report on evaluation of KpqC Round-2 candidates*. Cryptology ePrint Archive, Paper 2024/2077. 2024. URL: https://eprint.iacr.org/2024/2077.

[BCP+25] Giacomo Borin, Sofía Celi, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Threshold Signatures Reloaded: ML-DSA and Enhanced Raccoon with Identifiable Aborts*. Cryptology ePrint Archive, Paper 2025/1166. 2025. URL: https://eprint.iacr.org/2025/1166.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS–Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018.

[BDK+21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS–Dilithium: Algorithm Specification and Supporting Documentation (Version 3.1)*. Round-3 submission to the NIST Post-Quantum Cryptography Standardization Project. 2021. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[BG14] Shi Bai and Steven D. Galbraith. "An Improved Compression Technique for Signatures Based on Learning with Errors". In: *Topics in Cryptology – CT-RSA 2014*. 2014.

[BT23] Katharina Boudgoust and Akira Takahashi. "Sequential half-aggregation of lattice-based signatures". In: *European Symposium on Research in Computer Security*. 2023.

[CCD+24] Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. "HAETAE: Shorter lattice-based fiat-shamir signatures". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 3 (2024).

[CGH+25] Sofía Celi, Scott Griffy, Lucjan Hanzlik, Octavio Perez-Kempner, and Daniel Slamanig. "SoK: Signatures with Randomizable Keys".

In: *Financial Cryptography and Data Security*. 2025. ISBN: 978-3-031-78679-2.

[CGT+23] Jean-Sébastien Coron, François Gérard, Matthias Trannoy, and Rina Zeitoun. "Improved gadgets for the high-order masking of dilithium". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 4 (2023).

[DFL19] Poulami Das, Sebastian Faust, and Julian Loss. "A formal treatment of deterministic wallets". In: *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2019.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "Crystals-dilithium: A lattice-based digital signature scheme". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018).

[DPKM+24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarinen. "Threshold raccoon: Practical threshold signatures from standard lattice assumptions". In: *Annual international conference on the theory and applications of cryptographic techniques*. 2024.

[ELW23] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. *Security Analysis of Signature Schemes with Key Blinding*. Cryptology ePrint Archive, Paper 2023/380. 2023. URL: https://eprint.iacr.org/2023/380.

[FH20] Masaya Fukumitsu and Satoru Hasegawa. "A Lattice-Based Provably Secure Multisignature Scheme in the Quantum Random Oracle Model". In: *Provable Security – ProvSec 2020*. 2020.

[FKM+16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. "Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys". In: *Public-Key Cryptography – PKC 2016*. 2016.

[Mil86] Victor S. Miller. "Use of Elliptic Curves in Cryptography". In: *Advances in Cryptology — CRYPTO '85*. 1986.

[Nat16] National Institute of Standards and Technology. *Call for Proposals for Public-Key Post-Quantum Cryptographic Algorithms*. Federal Register Notice. Dec. 2016. URL: https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[Nat24a] National Institute of Standards and Technology (NIST). *Module-Lattice-Based Digital Signature Standard*. Federal Information Processing Standards Publication 204. U.S. Department of Commerce, 2024. URL: https://csrc.nist.gov/pubs/fips/204/final.

[Nat24b] National Institute of Standards and Technology (NIST). *Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)*. Federal Information Processing Standards Publication 203. Stan-

dardization of CRYSTALS-Kyber. U.S. Department of Commerce, 2024.

[PKP+24] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. "Raccoon: A masking-friendly signature proven in the probing model". In: *Annual International Cryptology Conference*. 2024.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 2 (1978).

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994.

[SR25] Michel Seck and Adeline Roux-Langlois. "Towards Post-Quantum Bitcoin Blockchain using Dilithium Signature". In: *IACR Communications in Cryptology* 3 (Oct. 6, 2025). ISSN: 3006-5496.

[SVL24] Nikita Snetkov, Jelizaveta Vakarjuk, and Peeter Laud. "TOPCOAT: towards practical two-party Crystals-Dilithium". In: *Discover Computing* 1 (2024).

[The25] The Tor Project. *Tor Rendezvous Specification - Version 3*. Tech. rep. The Tor Project, 2025. URL: https://spec.torproject.org/rend-spec/.