Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate

Prasanna Ravi[1], Mahabir Prasad Jhanwar[2], James Howe[3], Anupam Chattopadhyay[4], and Shivam Bhasin[1]

[1] Temasek Laboratories, Nanyang Technological University, Singapore
[2] Ashoka University, India
[3] University of Bristol, UK
[4] School of Computer Science and Engineering
Nanyang Technological University, Singapore

prasanna.ravi@ntu.edu.sg mahavir.jhawar@ashoka.edu.in
james.howe@bristol.ac.uk anupam@ntu.edu.sg sbhasin@ntu.edu.sg

Abstract. The recent lattice-based signature scheme Dilithium, submitted as part of the CRYSTALS (Cryptographic Suite for Algebraic Lattices) package, is one of a number of strong candidates submitted for the NIST standardization process of post-quantum cryptography. The Dilithium signature scheme is based on the Fiat-Shamir paradigm and can be seen as a variant of the Bai-Galbraith scheme (BG) combined with several improvements from previous ancestor lattice-based schemes like GLP and BLISS signature schemes. One of the main features of Dilithium is the *compressed* public-key, which is a rounded version of the LWE instance. This implies that Dilithium is not breakable with the knowledge of only the secret or the error of the LWE instance, unlike its ancestor lattice-based signature schemes. In this paper, we investigate the security of Dilithium against a combination of side-channel and classical attacks. Side-channel attacks on schoolbook and optimised polynomial multiplication algorithms in the signing procedure are shown to extract the secret component of the LWE instance, which is just one among the multiple components of the secret-key of Dilithium. We then propose an alternative signing procedure, through which it is possible to forge signatures with only the extracted portion of the secret-key, without requiring the knowledge of all its elements. Thus showing that Dilithium too breaks on just knowing the secret portion of the LWE instance, similar to previous lattice-based schemes.

1 Introduction

Recently, NIST has called for proposals for standardization of post-quantum cryptographic schemes for public-key encryption, digital signatures, and key establishment protocols [18]. This initiative is partly driven by the onset of the era of practical and scalable quantum computers [3, 15, 26], which has motivated the cryptographic community to develop cryptographic schemes that are immune

to cryptanalytic efforts using quantum algorithms. Amongst the initial candidates from different types of post-quantum cryptography, lattice-based cryptography has emerged as the largest category in terms of submission numbers. NIST [19] state that "schemes that can be made resistant to side-channel attack at minimal cost are more desirable than those whose performance is severely hampered by any attempt to resist side-channel attacks. Thus, it has become essential to the standardisation process to analyse the implementation security of these proposals against active and passive side-channel attacks."

Over the years, significant research has shown that lattice-based cryptographic algorithms [13, 22] have the performance required to replace currently used classical cryptographic primitives such as RSA and ECC [14]. Most of the efficient lattice-based signature schemes [9, 10, 13] are based on the variant of the Fiat-Shamir with Aborts framework, first proposed by Lyubashevsky [16]. The BLISS signature scheme [9] based on the same framework attracted significant attention from the scientific community, owing to its security and efficiency guarantees. However, its physical security came under strong scrutiny, owing to the number of implementation attacks [6, 11, 12, 20, 21], which targetted its modules such as Gaussian sampling and rejection sampling.

These attacks have lead the cryptographic community to look beyond the BLISS signature scheme and recently Ducas et al. proposed Dilithium [10, 17], a lattice-based signature scheme submitted to NIST for post-quantum standardisation, which removes the Gaussian sampling attack vector by employing uniform sampling instead. The signature scheme derives its hardness guarantees from hard problems on modular lattices, which has a lesser exploitable structure compared to ideal lattices, which is used in BLISS. The scheme also derives certain properties from other previously proposed signature schemes [2, 9, 13] to improve upon the sizes of public-key and signature. Certain decisions on the algorithm and parameters have been taken to make sure that some implementation attacks reported on BLISS cannot be easily applied on the Dilithium signature scheme.

Most, if not all, lattice-based signature schemes partition their secret-keys in the form of two polynomials; $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, and are used to form a LWE instance $\boldsymbol{t} = \boldsymbol{a}\boldsymbol{s}_1 + \boldsymbol{s}_2$, which is declared as the public key. A scheme utilising this format thus breaks down with knowledge of either of the secrets, since the LWE instance becomes trivially solvable. While this is the case with Dilithium's ancestral schemes like BLISS, GLP and BG, Dilithium publishes only a rounded version of the LWE instance as its public key. Thus, unlike its ancestral scheems, Dilithium does not trivially break-down with the knowledge of the secret or the error. In this paper, we demonstrate a side-channel assisted existential forgery attack on the Dilithium signature scheme wherein we mainly bring to light the possibility of forging signatures with only the partial knowledge of the secret-key $\boldsymbol{s}_1$. We first demonstrate a side-channel attack on the polynomial multiplication component, to retrieve the secret of the LWE instance, which is only one of the many components of its secret-key, and further propose an alternate signing procedure that utilizes only the extracted portion of the secret-key ($\boldsymbol{s}_1$) to produce valid

signatures, thus showing that Dilithium too breaks with the knowledge of only the secret of the LWE instance $\boldsymbol{s}_1$.

1.1 Contribution

In this work, we propose a side-channel assisted existential forgery attack on the Dilithium signature scheme.

- The side-channel attack is designed to retrieve partial secret-key ($\bar{\boldsymbol{s}}_1$) through a power analysis attack on the polynomial multiplier in the signing procedure.
- We further exploit certain properties of the Dilithium signature scheme to show that one can forge signatures with only knowledge of the partial secret-key. This procedure can also be viewed as an alternative signing procedure which involves lesser computations compared to the original scheme, but accompanied with a certain failure probability.

The idea of the attack over the polynomial multiplier was proposed by Espitau et al. [12]. We modified the attack to suit the specifications of Dilithium and exploit the leakage in a different operation of the polynomial multiplication. We acknowledge another parallel work of Bruinderink and Pessl [7] which also proposes a forgery attack with the knowledge of the partial secret-key, however we provide a simpler alternate forgery scheme with a concrete analysis of the properties of the Dilithium scheme that facilitated our attack.

2 Preliminaries

Let $q \in \mathbb{N}$ be a prime. For an integer r and an even positive integer α, we define centered reduction modulo q, denote as $r \pmod^{\pm} \alpha$, to be the unique integer r_0 such that, $r \equiv r_0 \pmod{\alpha}$ and $\frac{\alpha}{2} < r_0 \leq \frac{\alpha}{2}$. The usual modulo reduction is denoted by $r \pmod{q}$. We let $\mathbb{Z}$ denote the ring of integers, and $\mathbb{Z}_q$ the ring of integers modulo q. The elements in $\mathbb{Z}$, $\mathbb{Z}_q$ are denoted by regular font letters, viz. $a, b \in \mathbb{Z}$ or $\mathbb{Z}_q$. We let R_q denote the ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$, $q, n \in \mathbb{N}$. Bold lower-case letters represents elements in R_q, viz. $\boldsymbol{a}, \boldsymbol{b} \in R_q$. Column vectors with coefficients in R_q are denoted by bold lower-case letters with an overhead bar, viz. $\bar{\boldsymbol{a}}, \bar{\boldsymbol{b}} \in R_q^\ell$, $\ell \in \mathbb{N}$. Note that, elements $\boldsymbol{a} \in R_q$ are polynomials of degree at most $n-1$ with coefficients in $\mathbb{Z}_q$ and therefore can be represented as n-length vectors $(a_0, \ldots, a_{n-1})$, $a_i \in \mathbb{Z}_q$. By default, all vectors will be column vectors. Matrices with coefficients in R_q are denoted by bold upper-case letters, viz. $\boldsymbol{A} \in R_q^{k \times \ell}$, $k, \ell \in \mathbb{N}$. Transpose of a vector $\bar{\boldsymbol{a}}$ is denoted by $\bar{\boldsymbol{a}}^\mathsf{T}$. For $q, n, \ell \in \mathbb{N}$, R_q^ℓ is an R module with scalar multiplication $R \times R_q^\ell \to R_q^\ell$ defined as follows: for $\boldsymbol{a} \in R, \bar{\boldsymbol{b}} = (\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{n-1})^\mathsf{T} \in R_q^\ell$, $\boldsymbol{a}\bar{\boldsymbol{b}} = (\boldsymbol{a}\boldsymbol{b}_0, \ldots, \boldsymbol{a}\boldsymbol{b}_{n-1})^\mathsf{T}$, where $\boldsymbol{a}\boldsymbol{b}_i$ is polynomial multiplication defined in the ring R_q. A definition for the length of elements in R_q is as follows. For an element $\boldsymbol{a} = (a_0, \ldots, a_{n-1})$, define $\|\boldsymbol{a}\|_\infty = \max_{0 \leq i \leq n-1} \|a_i\|_\infty$, where $\|a_i\|_\infty = |a_i \pmod^{\pm} q|$. Similarly, for $\bar{\boldsymbol{a}} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell)^\mathsf{T} \in R_q^\ell$, we define $\|\bar{\boldsymbol{a}}\|_\infty = \max_{1 \leq i \leq \ell} \|\boldsymbol{a}_i\|_\infty$. Finally, for a $\eta \in \mathbb{N}$,

define $S_\eta = \{\boldsymbol{a} \in R_q \mid \|\boldsymbol{a}\|_\infty \leq \eta\}$. For a set X, we write $x \xleftarrow{\$} X$ to denote that x is chosen uniformly at random from X, and $x \xleftarrow{D} X$ when $D : X \to [0,1]$ is a specific (non-uniform) probability distribution.

2.1 Underlying Lattice Problems

Definition 1 (MLWE). *The module learning with error (MLWE) assumption states, roughly, that the distribution* $(\boldsymbol{A}, \bar{\boldsymbol{t}})$ *and* $(\boldsymbol{A}, \boldsymbol{A}\bar{\boldsymbol{s}}_1 + \bar{\boldsymbol{s}}_2)$ *are computationally indistinguishable when* $\boldsymbol{A} \xleftarrow{\$} R_q^{k\times\ell}$ *($k, \ell \in \mathbb{N}$),* $\bar{\boldsymbol{t}} \xleftarrow{\$} R_q^k$, $\bar{\boldsymbol{s}}_1 \xleftarrow{\$} S_\eta^\ell$, *and* $\bar{\boldsymbol{s}}_2 \xleftarrow{\$} S_\eta^k$ *($\eta \in \mathbb{N}$). In particular, for positive integers k, ℓ, η, the advantage of an adversary $\mathcal{A}$ in solving a decisional* $\mathsf{MLWE}_{k,\ell,\eta}$ *problem instance over the ring R_q is*

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}.\mathsf{MLWE}_{k,\ell,\eta} = |\mathbb{P}[\mathcal{A}(\boldsymbol{A}, \bar{\boldsymbol{t}}) = 1 \mid \boldsymbol{A} \xleftarrow{\$} R_q^{k\times\ell}; \bar{\boldsymbol{t}} \xleftarrow{\$} R_q^k] \\ - \mathbb{P}[\mathcal{A}(\boldsymbol{A}, \boldsymbol{A}\bar{\boldsymbol{s}}_1 + \bar{\boldsymbol{s}}_2) = 1 \mid \boldsymbol{A} \xleftarrow{\$} R_q^{k\times\ell}; \bar{\boldsymbol{s}}_1 \xleftarrow{\$} S_\eta^\ell; \bar{\boldsymbol{s}}_2 \xleftarrow{\$} S_\eta^k]|\end{aligned}$$

and the MLWE hardness assumption says that the advantage $\mathsf{Adv}^{\mathcal{A}}.\mathsf{MLWE}_{k,\ell,\eta}$ *for any probabilistic polynomial time* (PPT) *adversary $\mathcal{A}$ is negligible.*

Definition 2 (MSIS). *The MSIS hardness assumptions states that given* $(\boldsymbol{A}, \bar{\boldsymbol{t}}, \beta)$, *where* $\boldsymbol{A} \xleftarrow{\$} R_q^{k\times\ell}$, $\bar{\boldsymbol{t}} \in R_q^k$, *and* $\beta \in \mathbb{N}$, *it is hard to compute an* $\bar{\boldsymbol{x}} \in R_q^{\ell+k}$ *such that* $\|x\|_\infty \leq \beta$ *and* $[\boldsymbol{A}|\boldsymbol{I}]\bar{\boldsymbol{x}} = \bar{\boldsymbol{t}}$, *where* $\boldsymbol{I} = \boldsymbol{I}_{k\times k}$, *$k$ order identity matrix. In particular, for positive integers k, ℓ, β, the advantage of an adversary $\mathcal{A}$ in solving a* $\mathsf{MSIS}_{k,\ell,\beta}$ *problem instance over the ring R_q is*

$$\mathsf{Adv}^{\mathcal{A}}.\mathsf{MSIS}_{k,\ell,\beta} = \mathbb{P}[\|\bar{\boldsymbol{x}}\|_\infty \leq \beta \wedge [\boldsymbol{A}|\boldsymbol{I}]\bar{\boldsymbol{x}} = \bar{\boldsymbol{t}} \mid \boldsymbol{A} \xleftarrow{\$} R_q^{k\times\ell}; \bar{\boldsymbol{t}} \in R_q^k; \bar{\boldsymbol{x}} \leftarrow \mathcal{A}(\boldsymbol{A}, \bar{\boldsymbol{t}}, \beta)]$$

and the MSIS hardness assumption says that the advantage $\mathsf{Adv}^{\mathcal{A}}.\mathsf{MSIS}_{k,\ell,\beta}$ *for any probabilistic polynomial time* (PPT) *adversary $\mathcal{A}$ is negligible.*

2.2 Rounding Algorithms

This section describes several rounding procedures used in Dilithium. The purpose of them is to provide better compression of Dilithium's public-key and signature. Figure 1 lists the full details of these algorithms. For non-negative integers r, α, the procedure $r \pmod{}^{\pm} \alpha$ outputs unique integer r_0 in $-\alpha/2 < r_0 \leq \alpha/2$ such that $r \equiv r_0 \pmod{\alpha}$. The procedure $\mathsf{D}_q(r, \alpha)$ decomposes r into a pair of integers (r_1, r_0). The procedures $\mathsf{HB}_q(r, \alpha)$ and $\mathsf{LB}_q(r, \alpha)$ extracts r_1 and r_0 respectively from $\mathsf{D}_q(r, \alpha)$. The procedure $\mathsf{MH}_q(u, r, \alpha)$ produces a bit $h \in \{0, 1\}$. The procedure $\mathsf{UH}_q(h, r, \alpha)$ shows how to use h as a hint to derive $\mathsf{HB}_q(u + r, \alpha)$ when z is not known and its correctness is stated in the following lemma.

Lemma 1 ([17]). *For $r, z \in \mathbb{Z}_q$ with $||z||_\infty \leq \alpha/2$, we have*

$$\mathsf{UseHint}_q\left(\mathsf{MakeHint}_q(z, r, \alpha), r, \alpha\right) = \mathsf{HighBits}_q(r + z, \alpha) \tag{1}$$

Function CenteredModulo $r(\text{mod}^{\pm}\ \alpha)$**:**
  Compute $r_0 = r \pmod{\alpha}$
  if $r_0 > \alpha/2$ **then**
    $r_0 = r_0 - \alpha$
  end
  return $r \pmod{\pm}\ \alpha) = r_0$

Function Decompose $\mathsf{D}_q(r, \alpha)$**:**
  Compute $r = r \pmod{q}$
  Compute $r_0 = r \pmod{\pm}\ \alpha)$
  if $r - r_0 = q - 1$ **then**
    $r_1 = 0$
    $r_0 = r_0 - 1$
  else
    $r_1 = (r - r_0)/\alpha$
  end
  return $\mathsf{D}_q(r, \alpha) = (r_1, r_0)$

Function HighBits $\mathsf{HB}_q(r, \alpha)$**:**
  Compute $(r_1, r_0) = \mathsf{D}_q(r, \alpha)$
  return $\mathsf{HB}_q(r, \alpha) = r_1$

Function LowBits $\mathsf{LB}_q(r, \alpha)$**:**
  Compute $(r_1, r_0) = \mathsf{D}_q(r, \alpha)$
  return $\mathsf{LB}_q(r, \alpha) = r_0$

Function MakeHint $\mathsf{MH}_q(u, r, \alpha)$**:**
  Compute $r_1 = \mathsf{HB}_q(r, \alpha)$
  Compute $v_1 = \mathsf{HB}_q(u + r, \alpha)$
  if $r_1 = v_1$ **then**
    $h = 0$
  else
    $h = 1$
  end
  return $\mathsf{MH}_q(u, r, \alpha) = h$

Function UseHint $\mathsf{UH}_q(h, r, \alpha)$**:**
  Compute $m = (q - 1)/\alpha$
  Compute $(r_1, r_0) = \mathsf{D}_q(r, \alpha)$
  if $h = 1$ *and* $r_0 > 0$ **then**
    $r_1 = (r_1 + 1) \pmod{m}$
  end
  else if $h = 1$ *and* $r_0 \leq 0$ **then**
    $r_1 = (r_1 - 1) \pmod{+}\ m)$
  else
    $r_1 = r_1$
  end
  return $\mathsf{UH}_q(h, r, \alpha) = r_1$

Fig. 1: Rounding Algorithms

All these procedures extend naturally to ring elements in R_q. For example, if $\boldsymbol{r} = (r_0, \ldots, r_{n-1}) \in R_q$ then $\mathsf{D}_q(\boldsymbol{r}, \alpha) = (\mathsf{D}_q(r_0), \ldots, \mathsf{D}_q(r_{n-1}))$. Finally the procedures are extended to vectors in R_q^ℓ and matrices in $R_q^{k \times \ell}$: for $\bar{\boldsymbol{a}} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell)^\mathsf{T} \in R_q^\ell$, $\mathsf{HB}_q(\bar{\boldsymbol{a}}, \alpha) = (\mathsf{HB}_q(\boldsymbol{a}_1, \alpha), \ldots, \mathsf{HB}_q(\boldsymbol{a}_\ell, \alpha))$.

2.3 Dilithium

In the following, we recall the Dilithium signature scheme [10,17]. The underlying approach of the scheme is based on the "Fiat-Shamir with Aborts" approach [16] and is an improved variant of the scheme proposed by Bai and Galbraith [2]. Dilithium depends on parameters $q, n, k, \ell, \eta, d, \gamma_1, \gamma_2, \beta, \omega$. Various constraints and recommended values for these parameters are provided in the specifications [17]. While reading the scheme below, the reader may keep in mind the following set of recommended parameters: ($q = 8380417 = 2^{23} - 2^{13} + 1, n = 256, k = 3, \ell = 2, \eta = 7, d = 14, \gamma_1 = (q-1)/16, \gamma_2 = \gamma_1/2, \beta = 375, \omega = 64$). In addition, the scheme also uses:

- $H : \{0,1\}^* \to B_h$, a hash function, where B_h denote the set of elements of R_q that have h coefficients that are either -1 or 1 and the rest are 0;

– $\mathsf{ExpandA} : \{0,1\}^{256} \to \{\boldsymbol{A} \mid \boldsymbol{A} \in R_q^{k\times\ell}\}$;
– $\mathsf{ExpandMask} : \{0,1\}^{256} \times \{0,1\}^* \times \mathbb{N} \to S^{\ell}_{\gamma_1-1}$; and
– a collision resistant hash function $\mathsf{CRH} : \{0,1\}^* \to \{0,1\}^{384}$.

For complete details of these function please refer to [17]. The key generation, signing and verification algorithms for Dilithium are presented in Figure 2.

– **Key Generation**: The key generation algorithm, KeyGen, begins by choosing $\rho, K \xleftarrow{\$} \{0,1\}^{256}$. It then computes a matrix $\boldsymbol{A} \in R_q^{k\times\ell} = \mathsf{ExpandA}(\rho)$. Next, it samples secret-key vectors $\bar{\boldsymbol{s}}_1 \in S_\eta^\ell$, $\bar{\boldsymbol{s}}_2 \in S_\eta^k$. It then computes $\bar{\boldsymbol{t}} = \boldsymbol{A}\bar{\boldsymbol{s}}_1 + \bar{\boldsymbol{s}}_2$ and decomposes it into $\mathsf{HB}_q(\bar{\boldsymbol{t}}, 2^d) = \bar{\boldsymbol{t}}_1$ and $\mathsf{LB}_q(\bar{\boldsymbol{t}}, 2^d) = \bar{\boldsymbol{t}}_0$. Finally, $\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2, \bar{\boldsymbol{t}}_0, K$ are set as part of the secret-key; and $\bar{\boldsymbol{t}}_1$ is set as part of public-key.
– **Signing**: The signing algorithm, Sign, proceeds by generating a masking vector $\bar{\boldsymbol{y}} \in S^{\ell}_{\gamma_1-1}$ using ExpandMask function. It then computes $\bar{\boldsymbol{w}} = \boldsymbol{A}\bar{\boldsymbol{y}}$ and sets $\bar{\boldsymbol{w}}_1$ to be the $\mathsf{HB}_q(\bar{\boldsymbol{w}}, 2\gamma_2)$. The challenge, $\boldsymbol{c}$, is created as the hash of μ and $\bar{\boldsymbol{w}}_1$. The potential signature is then computed as $\bar{\boldsymbol{z}} = \bar{\boldsymbol{y}} + \boldsymbol{c}\bar{\boldsymbol{s}}_1$. To avoid the dependency of $\bar{\boldsymbol{z}}$ on the secret-key $\bar{\boldsymbol{s}}_1$, rejection sampling on $\bar{\boldsymbol{z}}$ is used unless $\|\bar{\boldsymbol{z}}\|_\infty < \gamma_1 - \beta$ (the probability of the later event is high as β is so chosen that $\|c\bar{\boldsymbol{s}}_2\|_\infty \le \beta$). The rejection sampling on $\bar{\boldsymbol{z}}$ also checks if $\mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) < \gamma_2 - \beta$ (note that, $\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}} = \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2$), this check is to ensure both security and correctness. In order to further proceed with the sampled $\bar{\boldsymbol{z}}$ it finally checks if $\mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) = \bar{\boldsymbol{w}}_1$. At this moment, the tuple $(\bar{\boldsymbol{z}}, \boldsymbol{c})$, together with $\bar{\boldsymbol{t}}$, can be used to verify if $c \stackrel{?}{=} H(\mu||\mathsf{HB}_q(\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}, 2\gamma_2))$. But the public-key of the verifier only has $\bar{\boldsymbol{t}}_1 = \mathsf{HB}_q(\bar{\boldsymbol{t}}, 2^d)$. To overcome this problem, the signing algorithm provides a hint vector $\bar{\boldsymbol{h}}$ (in fact, a binary matrix) which will allow the verifier to compute $\bar{\boldsymbol{w}}_1$. This is done in Step 23 (Figure 2) with some other additional checks carried out in Step 24.
– **Verification**: The verifier computes $\bar{\boldsymbol{w}}'_1 = \mathsf{UH}_q(\bar{\boldsymbol{h}}, \boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 \cdot 2^d, 2\gamma_2)$, and then accepts if $\|\bar{\boldsymbol{z}}\|_\infty < \gamma_1 - \beta$, $\boldsymbol{c} = H(\mathsf{CRH}(\mathsf{CRH}(\rho||\bar{\boldsymbol{t}}_1)||M), \bar{\boldsymbol{w}}'_1)$, and $\mathsf{wt}(\bar{\boldsymbol{h}}) \le \omega$. Let us look at why verification works, in particular as to why $\bar{\boldsymbol{w}}_1 = \mathsf{UH}_q(\bar{\boldsymbol{h}}, \boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 \cdot 2^d, 2\gamma_2)$. Notice that $\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 \cdot 2^d = \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0$. Therefore,

$$
\begin{aligned}
\bar{\boldsymbol{w}}_1 &= \mathsf{HB}_q(\bar{\boldsymbol{w}}, 2\gamma_2) \\
&\stackrel{*}{=} \mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) \\
&= \mathsf{HB}_q((-\boldsymbol{c}\bar{\boldsymbol{t}}_0) + (\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0), 2\gamma_2) \\
&\stackrel{**}{=} \mathsf{UH}_q(\mathsf{MH}_q(-\boldsymbol{c}\bar{\boldsymbol{t}}_0, \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0, 2\gamma_2), \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0, 2\gamma_2) \\
&= \mathsf{UH}_q(\bar{\boldsymbol{h}}, \boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 \cdot 2^d, 2\gamma_2).
\end{aligned}
$$

The * equality is true as follows: $\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 = \bar{\boldsymbol{r}}_1 \times 2\gamma_2 + \mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2)$ such that $\|\mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2)\|_\infty \le \gamma_2 - \beta$ (ensured during signing). This implies, $\bar{\boldsymbol{w}} = (\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2) + \boldsymbol{c}\bar{\boldsymbol{s}}_2 = \bar{\boldsymbol{r}}_1 \times 2\gamma_2 + \mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) + \boldsymbol{c}\bar{\boldsymbol{s}}_2$. But, as $\|\boldsymbol{c}\bar{\boldsymbol{s}}_2\|_\infty \le \beta$, $\|\mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) + \boldsymbol{c}\bar{\boldsymbol{s}}_2\|_\infty \le \|\mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2)\|_\infty + \|\boldsymbol{c}\bar{\boldsymbol{s}}_2\|_\infty \le \gamma_2 - \beta + \beta =$

γ_2. Thus, from the definition of centered reduction modulo $2\gamma_2$, it follows that $\mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) = \mathsf{HB}_q(\bar{\boldsymbol{w}}, 2\gamma_2) = \bar{\boldsymbol{w}}_1$. The ** equality follows from Lemma 1, as $\| - \boldsymbol{c}\bar{\boldsymbol{t}}_0\|_\infty \leq 2^d \leq \gamma_2$.

2.4 Security of Dilithium

The security of Dilithium is derived from MLWE and MSIS problems. In particular, security against key-recovery attack under classical random oracle model is based on the hardness assumption of the MLWE problem; and the security against existential signature forgery is due to MSIS hardness assumption. The scheme's security against strong signature forgery attack, under quantum random oracle is also discussed by the authors [17].

2.5 Side-Channel Attacks on Lattice-Based Cryptography

Lattice-based cryptography has gained significant attention from the perspective of SCA and fault attacks. Especially, predecessing lattice-based signature schemes like GLP [13], BLISS [9], and Ring-Tesla [1], which have been scrutinized through a number of active and passive implementation attacks, such as power analysis [20, 27], fault attacks [5, 11], branch-tracing [12], and cache-timing [6, 21]. These signature schemes opted to sample their error vectors from a Gaussian distribution and used rejection sampling to hide the information about the secret-key in the signature. Most of the side channel analysis research targetted the data dependent side-channel leakage from these Gaussian sampling and the rejection sampling components.

The attacks on the Gaussian samplers and rejection sampling techniques prompted the designers of the newer signature schemes, like Dilithium and qTESLA, to instead sample from uniform distributions during signing. This both simplifies the rejection step in addition to simplifying the sampling procedure, albeit with the cost of increased size of the signatures.

Lattice-based signature schemes have also been the target of a number of fault attacks [5, 7, 11]. Bindel et al. [5] proposed a variety of randomization, zeroing, and skipping fault attacks that lead to either key recovery or forging of signatures, applicable to BLISS, GLP, and Ring-TESLA. This was followed by a generic and much stronger fault attack on lattice-based signature schemes by Espitau *et al.* [11], based on loop abort faults. They were able to show that the signing step $\bar{\boldsymbol{z}} = \bar{\boldsymbol{s}}_1\boldsymbol{c} + \bar{\boldsymbol{y}}$, can be converted into a solvable CVP instance, provided the masking polynomials are limited to low degrees using loop abort faults. The first differential style fault attacks on Dilithium and qTESLA, signatures that are potential NIST standards, have been reported by Bruinderink and Pessl [7], which targets the generation of the $\bar{\boldsymbol{z}}$ component of the signature and work with a fault model of injecting random faults using clock glitches on an ARM Cortex-M4F based microcontroller.

Function $\mathsf{KeyGen}(q, n, k, \ell, \eta, d, \gamma_1, \gamma_2, \beta, \omega)$**:**

- $\rho \leftarrow \{0,1\}^{256}$
- $K \leftarrow \{0,1\}^{256}$
- $\boldsymbol{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$
- $(\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2) \leftarrow S_\eta^\ell \times S_\eta^k$
- $\bar{\boldsymbol{t}} = \boldsymbol{A}\bar{\boldsymbol{s}}_1 + \bar{\boldsymbol{s}}_2$
- $(\bar{\boldsymbol{t}}_1, \bar{\boldsymbol{t}}_0) = \mathsf{D}_q(\bar{\boldsymbol{t}}, 2^d)$
- $\mathsf{tr} = \mathsf{CRH}(\rho || \bar{\boldsymbol{t}}_1)$
- **Return :** $\mathsf{pk} = (\bar{\boldsymbol{t}}_1, \rho, n, q, k, d, \ell, \gamma_1, \gamma_2, \beta, \omega), \mathsf{sk} = (K, \mathsf{tr}, \bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2, \bar{\boldsymbol{t}}_0, \mathsf{pk})$

Function $\mathsf{Sign}(\mathsf{sk}, M)$**:**

- $\boldsymbol{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$
- $\mu = \mathsf{CRH}(\mathsf{tr} \| M)$
- $\kappa = 0, (\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}) = \perp$
- **while** $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}) = \perp$ **do**
  - $\bar{\boldsymbol{y}} \in S_{\gamma_1 - 1}^\ell := \mathsf{ExpandMask}(K \| \mu \| \kappa)$
  - $\bar{\boldsymbol{w}} = \boldsymbol{A}\bar{\boldsymbol{y}}$
  - $\bar{\boldsymbol{w}}_1 = \mathsf{HB}_q(\bar{\boldsymbol{w}}, 2\gamma_2)$
  - $\boldsymbol{c} \in B_{60} = H(\mu \| \bar{\boldsymbol{w}}_1)$
  - $\bar{\boldsymbol{z}} = \bar{\boldsymbol{y}} + \boldsymbol{c}\bar{\boldsymbol{s}}_1$
  - $(\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_0) := \mathsf{D}_q(\bar{\boldsymbol{w}} - c\bar{\boldsymbol{s}}_2, 2\gamma_2)$
  - **if** $\|\bar{\boldsymbol{z}}\|_\infty \geq \gamma_1 - \beta$ or $\|\bar{\boldsymbol{r}}_0\|_\infty \geq \gamma_2 - \beta$ or $\bar{\boldsymbol{r}}_1 \neq \bar{\boldsymbol{w}}_1$ **then**
    - $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}) = \perp$
  - **else**
    - $\bar{\boldsymbol{h}} = \mathsf{MH}_q(-\boldsymbol{c}\bar{\boldsymbol{t}}_0, \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0, 2\gamma_2)$
    - **if** $\|\boldsymbol{c}\bar{\boldsymbol{t}}_0\|_\infty \geq \gamma_2$ or $\mathsf{wt}(\bar{\boldsymbol{h}}) > \omega$ **then**
      - $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}) = \perp$
  - **end**
  - $\kappa = \kappa + 1$
- **end**
- **Return :** $\sigma = (\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}, \boldsymbol{c})$

Function $\mathsf{Verify}(\mathsf{pk}, M, \sigma = (\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}, \boldsymbol{c}))$**:**

- $\boldsymbol{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$
- $\mu = \mathsf{CRH}\left(\mathsf{CRH}(\rho \| \bar{\boldsymbol{t}}_1) \| M\right)$
- $\bar{\boldsymbol{w}}_1 := \mathsf{UH}_q(\bar{\boldsymbol{h}}, \boldsymbol{A}\bar{\boldsymbol{z}} - c\bar{\boldsymbol{t}}_1 \cdot 2^d, 2\gamma_2)$
- **if** $\boldsymbol{c} = H(\mu, \bar{\boldsymbol{w}}_1)$ and $\|\bar{\boldsymbol{z}}\|_\infty < \gamma_1 - \beta$ and $\mathsf{wt}(\bar{\boldsymbol{h}}) \leq \omega$ **then**
  - return 1
- **else**
  - return 0
- **end**

Fig. 2: Dilithium Signature Scheme

3 Motivation

In this section, we motivate the idea of existential forgery attacks with only knowledge of the partial secret-key. We first investigate the key features that distinguish Dilithium from some of the previous practical lattice-based signature schemes like BLISS [9], GLP [13], and Ring-TESLA [1] signature schemes. We provide some intuition which lead to the idea of signature forgery using partial knowledge of the secret-key. Additionally, we also highlight the importance of the signature forgery technique from the perspective of a side-channel adversary.

3.1 A few observations on Dilithium

Although most of the features in Dilithium are similar to those in its predecessors, like GLP or BLISS, one stand out property that differentiates it is the public-key being a compressed version of the LWE instance. The reason for this is to compress the size of the public-key. As noted earlier, only the d higher-order bits (i.e., 14 out of the 23 bits) of every coefficient of $\bar{\boldsymbol{t}}$ are revealed as the public-key. Alternatively, this can be seen as an additional error of about $(\pm 2^{14})$ to every coefficient, thus making it a LWE instance with an additional error. With such an instance at our disposal, even retrieval of the complete $\bar{\boldsymbol{s}}_1$ does not result in a straight-forward retrieval of the error $\bar{\boldsymbol{s}}_2$. But, previous schemes like GLP, BLISS, and BG signature schemes would be completely broken with the knowledge of either the secret or the error of the LWE instance. Thus, preliminary observations suggest that the *compressed* public key offers increased protection against signature forgery though the authors do not claim any additional security due to the compression of the public-key [17].

Another difference is that the $\bar{\boldsymbol{z}}_2$ component of the signature, that directly depends on the value of $\bar{\boldsymbol{s}}_2$, previously present in schemes like GLP and BLISS, has been completely removed owing to the effort to decrease the signature size, an idea derived from the BG signature scheme. Thus, Dilithium only consists of three components, $(\bar{\boldsymbol{z}}, \boldsymbol{c})$ and the additional hint vector $\bar{\boldsymbol{h}}$. One can generate the signature components $\bar{\boldsymbol{z}}$ and $\boldsymbol{c}$ with only the knowledge of $\bar{\boldsymbol{s}}_1$. As stated before, preliminary observations suggest that the $\bar{\boldsymbol{s}}_2$ component is required to generate the hint vectors. But, another parallel observation of the signing and verification procedures suggest that both the signer and verifier only commit over the higher-order bits of a particular intermediate value $(\bar{\boldsymbol{w}}_1)$. With $\bar{\boldsymbol{s}}_2$ being small and only used in the product of $\boldsymbol{c} \cdot \bar{\boldsymbol{s}}_2$, which is again a module with very small coefficients, it leads us to think if $\bar{\boldsymbol{s}}_2$ will have any observable impact over the committed value $(\bar{\boldsymbol{w}}_1)$, which also naturally leads us to think if it is possible to bypass the use of $\bar{\boldsymbol{s}}_2$ to forge signatures. A concrete analysis of this idea is followed in Section 4, where we show that it is indeed possible to forge signatures with only the knowledge of $\bar{\boldsymbol{s}}_1$.

3.2 Analysis from the perspective of a side-channel adversary

Analysis of previous works on implementation security of lattice-based signatures [5,7,11,12] suggest that most of the attacks only concentrated on recovering

$\bar{\boldsymbol{s}}_1$, as the retrieval of $\bar{\boldsymbol{s}}_2$ from the completely available LWE instance directly followed. Thus, the step of calculating the $\bar{\boldsymbol{z}}$ component of the signature, which involves the only computation using $\bar{\boldsymbol{s}}_1$ in the signing procedure, has been greatly scrutinized from a SCA and fault attack perspective to a great extent. A similar attack can also be mounted on Dilithium to retrieve $\bar{\boldsymbol{s}}_1$ (Step 19 of signing in Figure 2) provided the implementation is not sufficiently hardened against the previously known attacks. Upon recovering $\bar{\boldsymbol{s}}_1$, there are two possible methods that an adversary can follow to forge signatures.

The first technique is to try to recover the other secret $\bar{\boldsymbol{s}}_2$ using statistical analysis. With the knowledge of $\bar{\boldsymbol{s}}_1$, the adversary can trivially calculate the $\bar{\boldsymbol{w}}$ component for all the valid signatures. Since we also know that all the signatures output adhere to the condition $\mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) \leq \gamma_2 - \beta$, it is possible to derive certain conditions of the possible values of $\bar{\boldsymbol{s}}_2$. But as stated by Bruinderink and Pessl [7], it would require a large number of signatures and a very high computational effort, thus increasing the attacker's complexity. Additionally, an adversary cannot use this technique in an ephemeral key setting as they cannot expect to observe suitably many signatures to be generated using the same secret-key.

Additionally, the authors of Dilithium do not assume any secrecy with respect to the lower order bits of $\bar{\boldsymbol{t}}$ (i.e, $\bar{\boldsymbol{t}}_0$). In fact, the security proof of Dilithium is sketched under the assumption that the whole of $\bar{\boldsymbol{t}}$ is known to the adversary. Thus, it might indeed be possible that the whole of $\bar{\boldsymbol{t}}$ leaks as part of the signature and observations of sufficiently many signatures might lead to the recovery of the complete LWE instance, $\bar{\boldsymbol{t}}$. But again, we expect the number of signatures and the computational effort to be very high, which cannot be expected in a practical SCA setting.

The other technique the SCA adversary can attempt is to recover the other secret, $\bar{\boldsymbol{s}}_2$ or $\bar{\boldsymbol{t}}_0$, through side channels. There have been previous works on SCA and fault attacks over the polynomial multiplication operation $\boldsymbol{c}\bar{\boldsymbol{s}}_i$, for $i = \{1, 2\}$ [12, 27]. Since both the products are computed during the signing procedure, it is possible to mount similar SCA attacks to recover both $\bar{\boldsymbol{s}}_1$ and $\bar{\boldsymbol{s}}_2$ and also $\bar{\boldsymbol{t}}_0$.

But, in a practical SCA setting, it might not be always possible to retrieve the secrets completely and might pose a significant challenge owing to the requirement for the attacker to either analyse a large number of observations [12] for a given secret-key or inject a large number of successful faults [5]. Thus, an adversary might have to brute-force over the remaining coefficients of the attacked component ($\bar{\boldsymbol{s}}_2$ or $\bar{\boldsymbol{t}}_0$) to completely recover the key.

We analysed this case, with the recommended parameter set of Dilithium, to estimate the brute-force complexity. Figure 3 shows the brute-force complexity of finding the remaining coefficients of both the secrets, $\bar{\boldsymbol{s}}_1$ and $\bar{\boldsymbol{s}}_2$ plotted against the success rates in retrieving coefficients. It shows that the attack's complexity increases substantially, compared to when brute-forcing just one of the secrets. Thus, we can see that even for a success rate in retrieving the coefficients that is as high as 98%, the brute-force complexity is well over the 128 bit security level.

Even brute-forcing over one of the secrets with very high success rates of over 97% also crosses the 128-bit security level. Thus, in a practical setting, trying to retrieve both $\bar{\boldsymbol{s}}_1$ and $\bar{\boldsymbol{s}}_2$ might not be feasible even if success rate is very high and only very small percentage of coefficients are left unrecovered.

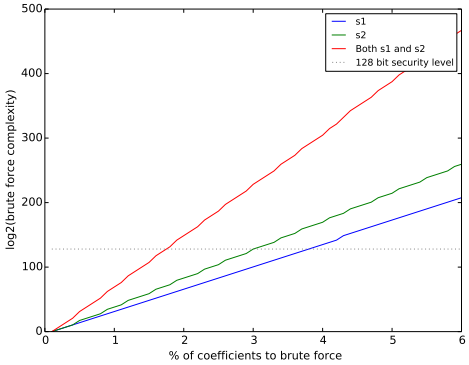


Fig. 3: Brute-force complexity versus the % of coefficients yet to be found.

Alternatively, the adversary can try to retrieve the remaining coefficients of $\bar{\boldsymbol{s}}_1$ using algorithms that solve lattice problems. A similar technique was proposed by Bindel et al. [5], who propose a hybrid approach to reduce the number of fault attacks to retrieve a certain number of coefficients, leaving the remaining coefficients to be retrieved using LWE solvers. This technique reduces the attacker's complexity in retrieving the whole of $\bar{\boldsymbol{s}}_1$. But, even if the whole of $\bar{\boldsymbol{s}}_1$ is retrieved, the remaining coefficients of $\bar{\boldsymbol{s}}_2$ still remain unknown. Referring to Figure 3, we again see that brute-forcing over remaining coefficients of one of the secrets requires a very high computation cost, even with a very high success rate.

Given all the above scenarios from an SCA perspective, we try to motivate the importance of our forgery signature scheme which requires only the partial knowledge of the secret-key. We do acknowledge another very recent parallel work by Bruinderink and Pessl [7], which proposes a different strategy to forge signatures in Dilithium with only the knowledge of $\bar{\boldsymbol{s}}_1$. However, the research lacks a clear analysis of the underlying factors that enabled signature forgery, with only the knowledge of $\bar{\boldsymbol{s}}_1$. However, we propose an alternate simpler forgery signature scheme backed up by a careful analysis along with a novel EM-based side channel attack strategy to recover $\bar{\boldsymbol{s}}_1$.

4 Existential Forgery Assuming Partial Secret-Key Knowledge

We now present an existential forgery attack on Dilithium assuming the knowledge of partial secret-key, $\bar{\boldsymbol{s}}_1$. For the following attack, refer to the function Sign in Figure 2. We consider an adversary $\mathcal{A}$, who has knowledge of $\bar{\boldsymbol{s}}_1$, and whose goal is to produce a valid signature of a new message. Execution of Lines 10 through 18 require knowledge of K (Line 14) and $\bar{\boldsymbol{s}}_1$ (Line 18) the partial secret-key. The security of Dilithium is unaffected if K is made public, indeed K is only used to deterministically generate randomness $\bar{\boldsymbol{y}}$. Thus, $\mathcal{A}$ proceeds through Lines 10 to 18, choosing $\bar{\boldsymbol{y}}$ uniformly at random from $S^{\ell}_{\gamma_1-1}$ in Line 14 and computing $\bar{\boldsymbol{z}}$ in Line 18 using $\bar{\boldsymbol{s}}_1$.

Signature verification requires knowledge of $\bar{\boldsymbol{w}}_1$. In the specifications [17], it is proven that the $\mathbb{P}[\bar{\boldsymbol{w}}_1 = \mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2)] \approx e^{-n\beta(\frac{\ell}{\gamma_1}+\frac{k}{\gamma_2})}$, very close to 1. Now $\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 = (-\boldsymbol{c}\bar{\boldsymbol{t}}_0) + (\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0)$. Writing $\bar{\boldsymbol{u}} = -\boldsymbol{c}\bar{\boldsymbol{t}}_0$, $\bar{\boldsymbol{r}} = \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0$, and the fact that $\mathbb{P}[\|\bar{\boldsymbol{u}}\|_\infty \leq \gamma_2] \approx 1$ (indeed, $\|-\boldsymbol{c}\bar{\boldsymbol{t}}_0\|_\infty \leq \|\bar{\boldsymbol{t}}_0\|_\infty < 2^d < \gamma_2$) together ensure (Lemma 1) that $\bar{\boldsymbol{w}}_1$ can be computed as follows:

$$\begin{aligned}\mathsf{UH}_q\left(\mathsf{MH}_q(\bar{\boldsymbol{u}}, \bar{\boldsymbol{r}}, 2\gamma_2), \bar{\boldsymbol{r}}, 2\gamma_2\right) &= \mathsf{HB}_q(\bar{\boldsymbol{u}} + \bar{\boldsymbol{r}}, 2\gamma_2) \\ &= \mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) \\ &= \bar{\boldsymbol{w}}_1.\end{aligned}$$

But computation of $\mathsf{MH}_q(\bar{\boldsymbol{u}}, \bar{\boldsymbol{r}}, 2\gamma_2)$ requires knowledge of $\bar{\boldsymbol{u}} = -\boldsymbol{c}\bar{\boldsymbol{t}}_0$, and therefore subsequently the partial secret-key $\bar{\boldsymbol{t}}_0$. *The problem for $\mathcal{A}$ is now to compute hint matrix $\bar{\boldsymbol{h}} = \mathsf{MH}_q(\bar{\boldsymbol{u}}, \bar{\boldsymbol{r}}, 2\gamma_2)$ without the knowledge of $\bar{\boldsymbol{t}}_0$.* In the following we first show that UH_q function could be inverted to produce the correct hint. In particular $\mathsf{UH}_q(h, r, \alpha)$, given q, α, $r \in \mathbb{Z}_q$ and $\mathsf{HB}_q(u + r, \alpha)$, is invertible on h if $\|u\|_\infty \leq \alpha/2$. Lemma 2 summarizes this fact.

Lemma 2. *Let $u \in \mathbb{Z}_q$ with $\|u\|_\infty \leq \alpha/2$, where $\alpha > 0$. Then, given any $r \in \mathbb{Z}_q$ and $\mathsf{HB}_q(u + r, \alpha)$, it is easy to compute $\mathsf{MH}_q(u, r, \alpha)$.*

Proof: The algorithm in Figure 4, given below, illustrates this claim. The correctness of the algorithm follows immediately from Lemma 1 and Lemma 3.

Lemma 3 ([17]). *Let $r \in \mathbb{Z}_q$ and $h, h' \in \{0, 1\}$. If $\mathsf{UH}_q(h, r, \alpha) = \mathsf{UH}_q(h', r, \alpha)$, then $h = h'$.*

Thus, in order to compute $\bar{\boldsymbol{h}}$, $\mathcal{A}$ must ensure that it has access to $\mathsf{HB}_q(\bar{\boldsymbol{u}} + \bar{\boldsymbol{r}}, 2\gamma_2) = \mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2)$, and $\bar{\boldsymbol{r}} = \bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0$. Computing $\bar{\boldsymbol{r}}$ is easy as $\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2 + \boldsymbol{c}\bar{\boldsymbol{t}}_0 = \boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d$. To compute $\mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2)$ all $\mathcal{A}$ has access to is $\bar{\boldsymbol{w}}_1$. But it is known that $\mathsf{HB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2) = \bar{\boldsymbol{w}}_1$ provided $\|\mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2)\|_\infty \leq \gamma_2 - \beta$. A valid signer could ensure this check, but for $\mathcal{A}$ it is not possible as it requires knowledge of $\bar{\boldsymbol{s}}_2$. But, this is not a problem as it is proven in [17] that, for a $\bar{\boldsymbol{y}} \xleftarrow{\$} S^{\ell}_{\gamma_1}$, $\mathbb{P}[\|\mathsf{LB}_q(\bar{\boldsymbol{w}} - \boldsymbol{c}\bar{\boldsymbol{s}}_2, 2\gamma_2)\|_\infty \leq \gamma_2 - \beta] \approx e^{\frac{-n\beta k}{\gamma_2}}$, very close to 1. We now finally present complete details of our existential forgery attack in Figure 5.

input : $q, r, \alpha, \theta = \mathsf{HB}_q(u + r, \alpha)$
output : $\mathsf{MH}_q(u, r, \alpha)$
Set $h = 0$;
Compute $\phi = \mathsf{UH}_q(h, r, \alpha)$;
if $\phi = \theta$ **then**
  return h
else
  return 1
end

Fig. 4: Inverting UH_q for the hint bit h.

It is important to note that the attack works for the recommended parameter sets of Dilithium.

We implemented our forgery signing procedure by modifying the reference implementation of Dilithium that was submitted to the NIST standardization process. The results were obtained on an Intel Core-i5 (Haswell) processor running at 2.6 GHz with Turbo Boost and Hyperthreading and compiled with gcc-4.2.1 without modifying the compiler flags set for the reference implementation. We obtained an average signing time of about 0.3253 msecs for our forgery signing procedure which is about 2.67 times faster than the signing time for the original signing procedure of Dilithium which runs at an average time of 0.8689 msec. The improved speed is due to the reduced number of operations and also the removal of rejection conditions over the norms of $\bar{\boldsymbol{r}}_0$ and $\boldsymbol{c}\bar{\boldsymbol{t}}_0$. We have uploaded the C code of our forgery signing procedure in the github link mentioned below[5].

It is natural for our forgery scheme to produce invalid signatures since the conditional checks on the norm of $\bar{\boldsymbol{r}_0}$ and $\boldsymbol{c}\bar{\boldsymbol{t}}_0$ are skipped. We attempted to empirically compute the failure probability of our forgery signing procedure. We could run our signature scheme for a total of 2^{28} times on the same platform with all the signatures verified correctly. This along with its increased signing rate leads us to hypothesize if our forgery signing procedure can be used as an alternative signing procedure with improved performance, provided it does not leak any information about the secret key. Concrete analysis of the same is left for future work.

4.1 Implications of the forgery signature scheme

Thus, we have shown that it is indeed possible to forge signatures, with only the knowledge of $\bar{\boldsymbol{s}}_1$, by successfully bypassing the use of the other elements of the secret-key. We do not claim any major break of the Dilithium signature scheme, but show that the knowledge of just one component, $\bar{\boldsymbol{s}}_1$, of the secret-key is enough to forge signatures which will be verified correctly with very high probability. But, it is important to state that the secret-key components; $\bar{\boldsymbol{s}}_1$, $\bar{\boldsymbol{s}}_2$,

[5] `https://github.com/jameshoweee/dilithium_forgery`

input : public-key $pk = (q, \rho, \bar{\boldsymbol{t}}_1)$, Partial secret-key $= \mathsf{s}_1$, A message M
output : A forged Dilithium signature

$\boldsymbol{A} \sim R_q^{k\times\ell} := \mathsf{Sam}(\rho)$;
$\mu = H\left(H(\rho \| \bar{\boldsymbol{t}}_1) \| M\right)$;
$\bar{\boldsymbol{y}} \xleftarrow{\$} S_{\gamma_1-1}^{\ell}$;
$\bar{\boldsymbol{w}} = \boldsymbol{A}\bar{\boldsymbol{y}}$;
$\bar{\boldsymbol{w}}_1 = \mathsf{HB}_q(\bar{\boldsymbol{w}}, 2\gamma_2)$;
$\boldsymbol{c} = H(\mu, \bar{\boldsymbol{w}}_1)$;
$\bar{\boldsymbol{z}} = \bar{\boldsymbol{y}} + \boldsymbol{c}\bar{\boldsymbol{s}}_1$;
$\bar{\boldsymbol{h}} = \begin{bmatrix} h_{10} \ldots h_{1(n-1)} \\ h_{20} \ldots h_{2(n-1)} \\ \vdots \quad \vdots \quad \vdots \\ h_{k0} \ldots h_{k(n-1)} \end{bmatrix} = \begin{bmatrix} 0 \ldots 0 \\ 0 \ldots 0 \\ \vdots \quad \vdots \quad \vdots \\ 0 \ldots 0 \end{bmatrix}$;
$\boldsymbol{\theta} = \begin{bmatrix} \theta_{10} \ldots \theta_{1(n-1)} \\ \theta_{20} \ldots \theta_{2(n-1)} \\ \vdots \quad \vdots \quad \vdots \\ \theta_{k0} \ldots \theta_{k(n-1)} \end{bmatrix} =$
$\begin{bmatrix} \mathsf{UH}_q(h_{10}, [\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d]_{10}, 2\gamma_2) \ldots \mathsf{UH}_q(h_{1(n-1)}, [\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d]_{1(n-1)}, 2\gamma_2) \\ \mathsf{UH}_q(h_{20}, [\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d]_{20}, 2\gamma_2) \ldots \mathsf{UH}_q(h_{2(n-1)}, [\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d]_{2(n-1)}, 2\gamma_2) \\ \vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ \mathsf{UH}_q(h_{k0}, [\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d]_{k0}, 2\gamma_2) \ldots \mathsf{UH}_q(h_{k(n-1)}, [\boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d]_{k(n-1)}, 2\gamma_2) \end{bmatrix}$;
for $i = 1$ *to* k **do**
  for $j = 0$ *to* $n-1$ **do**
    if $\theta_{ij} \neq [\bar{\boldsymbol{w}}_1]_{ij}$ **then**
      Set $h_{ij} = 1$
    end
  end
end
if $\mathsf{UH}_q(\boldsymbol{h}, \boldsymbol{A}\bar{\boldsymbol{z}} - \boldsymbol{c}\bar{\boldsymbol{t}}_1 2^d, 2\gamma_2) \neq \bar{\boldsymbol{w}}_1$ *or* $\|\bar{\boldsymbol{z}}\|_\infty \geq \gamma_1 - \beta$ *or* $\mathsf{wt}(\boldsymbol{h}) > \omega$ **then**
  Go to 3
else
  return $(\sigma = (\bar{\boldsymbol{z}}, \bar{\boldsymbol{h}}, \boldsymbol{c}))$
end

Fig. 5: $\mathsf{Forgery}(\mathsf{pk}, \bar{\boldsymbol{s}}_1, M)$

and $\bar{\boldsymbol{t}}_0$, are related to one another and it is public knowledge that knowing any of the two aforementioned components will result in a direct break of the signature scheme. In this research, we have shown that just the knowledge of $\bar{\boldsymbol{s}}_1$ is sufficient for an existential forgery attack.

A natural question arises as to whether the knowledge of $\bar{\boldsymbol{s}}_2$ only will be sufficient to forge signatures. But, according to the claim of Bai and Galbraith [2], the signature does not send any information about the error $\bar{\boldsymbol{s}}_2$ and thus does proves only the knowledge of $\bar{\boldsymbol{s}}_1$, but not $\bar{\boldsymbol{s}}_2$. Thus with only the knowledge of $\bar{\boldsymbol{s}}_2$, it should be impossible to forge signatures given the fact that $\bar{\boldsymbol{s}}_1$ and $\bar{\boldsymbol{s}}_2$ are independent. Thus, we show that the among all the elements of the secret key $(\rho, K, tr, \bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2, \bar{\boldsymbol{t}}_0)$, just knowing $\bar{\boldsymbol{s}}_1$ is sufficient to forge signatures, thus showing that the secrecy of $\bar{\boldsymbol{s}}_1$ is pivotal for the secrurity of Dilithium.

5 Retrieval of $\bar{\boldsymbol{s}}_1$

To complete our existential forgery attack on Dilithium, we demonstrate a power side-channel attack to retrieve $\bar{\boldsymbol{s}}_1$ by targetting the polynomial multiplication operation $\boldsymbol{c}\bar{\boldsymbol{s}}_1$. during the signing procedure. Recent public implementations of Dilithium utilise a number theoretic transform (NTT) for polynomial multiplication. Side-channel vulnerability of NTT-based polynomial multiplications is already known [27]. Designers can use other alternatives for computing multiplication, but the choice of the multiplication architecture does not necessarily protect against SCA. There are several works that acknowledge the fact that both the schoolbook and the sparse polynomial multiplier might yield better performance in certain corner cases over area-constrained devices, due to its simplified control logic [13, 23, 24]. Additionally, since the signature component has to be generated in the normal domain, use of the schoolbook or the sparse polynomial multiplier might yield better performance results compared to the use of the NTT polynomial multiplier. The sparse polynomial multiplier has been used in the reference implementation of another candidate lattice-based signature scheme, qTESLA [4], which further motivates us analyse the schoolbook polynomial multiplier and its efficient variant.

We use power analysis to demonstrate two types of attacks to recover $\bar{\boldsymbol{s}}_1$, targetting the polynomial multiplication operation $\bar{\boldsymbol{s}}_1 \times \boldsymbol{c}$, that is Step 19 of the signing procedure in Figure 2. SCA on two multiplier architectures are considered. The first one is a conventional DPA style attack on a schoolbook polynomial multiplier, while the second is a two stage horizontal and vertical DPA style attack on the efficient sparse polynomial multiplier protected with the countermeasure of randomized shuffling of non-zero indices of $\boldsymbol{c}$. The latter attack is based on implementation by Pöppelmann et al. [25] on 8-bit AVR microcontrollers. Both the schoolbook and sparse polynomial multiplication algorithms have been used for design of compact multipliers in a number of reported implementation works on both hardware and software platforms [8, 24].

We would like to clarify that $\bar{\boldsymbol{s}}_1 \in R_q^{\ell}$, while $\boldsymbol{c} \in B_{60}$ and hence can be assumed to be an element of R_q. The operation of $\bar{\boldsymbol{s}}_1 \times \boldsymbol{c}$ will be repeated ℓ

times with each polynomial of module $\bar{\boldsymbol{s}}_1$. But for notational convenience in this section, $\bar{\boldsymbol{s}}_1 \times \boldsymbol{c}$ is used to denote a single polynomial multiplication.

5.1 Attacking the schoolbook polynomial multiplier

The operands of the target polynomial multiplication operation ($\bar{\boldsymbol{s}}_1 \times \boldsymbol{c}$) are $\bar{\boldsymbol{s}}_1$, which has small coefficients in $[-\eta, \eta]$, and $\boldsymbol{c}$, which is a sparse polynomial with coefficients in $\{0, \pm 1\}$. If a schoolbook algorithm is used for the polynomial multiplier, then $\boldsymbol{c}(x) = \boldsymbol{a}(x) \cdot \boldsymbol{b}(x)$ in an ideal lattice setting can be described by the equation below as follows:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \left[\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \boldsymbol{a}_i \cdot \boldsymbol{b}_j x^{i+j} \right] \bmod \langle x^n + 1 \rangle = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (-1)^{\lfloor \frac{i+j}{n} \rfloor} \boldsymbol{a}_i \cdot \boldsymbol{b}_j x^{i+j \bmod n} \quad (2)$$

The intermediate result $\boldsymbol{a}_i \cdot \boldsymbol{b}_j$ is targeted (seen in Equation 2), wherein one of the operands $\boldsymbol{c}$ is known while the other operand $\boldsymbol{s}_1$ is unknown. This is an ideal setting to mount DPA attacks using a divide-and-conquer approach, and the secret-key can be retrieved one coefficient at a time. Each coefficient of $\boldsymbol{s}_1$ is multiplied with all the coefficients of $\boldsymbol{c}$ and observing one signature provides 256 points on the trace for the attack. Hence, information about a particular coefficient can also be retrieved from multiple points on a single trace, in addition to getting information about the same coefficient by observing multiple signatures.

5.2 Attacking the sparse polynomial multiplier

The sparse polynomial multiplier implementation we use [25] is a much more optimized algorithm, which takes advantage of the fact that one of the operands is a sparse polynomial. Only the indices of the non-zero coefficients of $\boldsymbol{c}$ are stored along with their sign. This restricts the multiplication of the coefficients of $\bar{\boldsymbol{s}}_1$ to only those that are non-zero, thus saving computational time. As a preliminary countermeasure, the non-zero coefficients of $\boldsymbol{c}$ are also shuffled and processed in a random order for each multiplication operation. Figure 6 describes the code corresponding to the computation of the sparse polynomial multiplier. Though $\boldsymbol{c}$ is still known, the random shuffling of its coefficients prevents from mounting a straightforward DPA attack. An attack on a sparse polynomial multiplier was reported by Espitau et al. [12]. We use a similar technique to recover $\bar{\boldsymbol{s}}_1$, albeit with modifications to suit Dilithium, while also targeting a different source of leakage compared to the original attack. The first step of the attack works by retrieving the order in which $\boldsymbol{c}$ is processed during every polynomial multiplication. This is done by performing a horizontal DPA attack on Step 5 of Figure 6. Since, one of the inputs (q) is known, a hypothesis over all the possible values of $\boldsymbol{c}[j]$ can be constructed to perform a horizontal style DPA attack using information available from a single trace, as the coefficients of $\boldsymbol{c}$ are shuffled after each multiplication operation. Once the order of indices for each polynomial multiplication is revealed, this information can be utilized to retrieve coefficients of $\bar{\boldsymbol{s}}_1$ using a DPA attack over computation in Step 9, since the value of val is dependent on $\boldsymbol{c}[j]$.

```
1  for (q=0; q<N; q++){
2          res[q] = 0;
3          for(j=0; j<N; j++){
4                  int8_t val = 1;
5                  int16_t i = (q - c[j]);
6                  if(i < 0){
7                          i += N;
8                          val = -val;}
9                  val *= s[i];
10                 res[q] += val;}
11         }
```

Fig. 6: Code snippet of optimised sparse polynomial multiplier [22]

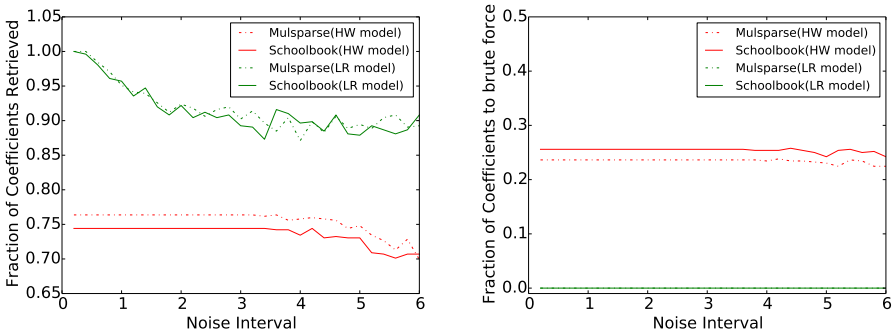
5.3 Experimental results

The above mentioned attacks were performed in a simulated setting, where uniform noise (over a given zero centered interval) is added to the traces, to test the robustness of the attack in a noisy environment. On assuming an 8-bit Hamming weight (HW) model leakage, there is a very limited number of observable hamming weights on the trace, given the fact that $\boldsymbol{s}_1$ is small and coefficients of $\boldsymbol{c} \in \{0, 1\}$. In addition to this, the distance between the various observable hamming weights is also very small (at least for values with the same sign). One can easily distinguish between whether the correct guess is a positive or negative value (due to the leading ones in the negative value). But distinguishing between different positive values is very tricky, as there is very little information that helps the attacker in distinguishing different key hypotheses. We used the least square error as the statistical distinguisher for both attacks, owing to the decreased number of observable hamming weights. We observed that some of the coefficients could not be retrieved even in a perfect noise free setting. For example, pairs $(3, 5)$ and $(-3, -5)$ are indistinguishable from each other, since they have the same hamming weights over 8 bits. For $\eta = 6$ and $N = 256$ (N being the degree of polynomial), the possible values of the coefficient are in $[-6, 6]$, and thus we cannot distinguish four out of the possible eleven cases. This boils down to a brute-force complexity of around $\approx 2^{79}$, on average, for each polynomial in the secret-key $\bar{\boldsymbol{s}}_1$, which renders the attack impractical. A work by Bindel et al. [5] showed that it is enough to reveal only 67% of coefficients in order to reveal the complete secret polynomial $\bar{\boldsymbol{s}}_1$ for BLISS. It is possible that a similar technique could be used for $\boldsymbol{s}_1$ in Dilithium to recover the remaining coefficients.

But, it is known that real devices do not necessarily leak in a perfect HW model. Realistically speaking, each bit has a certain bias associated with it and hence each bit weighs differently in leakage. This model of leakage is known as the linear regression model, where leakage corresponding to each bit i is weighed with

a particular coefficient α_i [28]. Both attacks were repeated in a simulated linear regression leakage (LR) model. We found that an introduction of a very small bias that is random in $[-0.01, 0.01]$ to each bit, introduced enough information that resulted in the recovery of almost all coefficients. But to demonstrate a practical case, we use the α coefficients that were profiled using an AES S-box on an AVR microcontroller. An important point to note is that the α coefficients relate to the characteristics of the device and hence any non-sensitive algorithm can also be used for profiling the device. In the case of the LR leakage model, we observed that almost all coefficients were retrieved with very high confidence (with clear distinction between the correct and wrong key hypotheses). The different bias associated with each bit helped distinguish the correct hypothesis from the wrong ones much more easily compared to the HW leakage setting. We also added uniform noise in interval $[-x, x]$ to test the robustness of both the attacks in a noisy environment.

Figures 7a and 7b show the plots for the fraction of coefficients retrieved successfully and the fraction of coefficients to be brute-forced, respectively, against an increasing value of noise interval. Note that the value of x in the x-axis corresponds to uniform noise in the interval $[-x, x]$. It can be seen that the success rate for both the attacks in the HW model is around 75% with high confidence and a brute-force on 25% of the coefficients is required on average. Both attacks in the LR model retrieve almost all coefficients and have a very high success of 90% retrieval, even in presence of noise, with no coefficients to brute-force.



(a) Fraction of coefficients retrieved against uniform noise.

(b) Fraction of coefficients to brute-force against uniform noise.

Fig. 7: Practical results for DPA attack against Noise

Comparing the results in Figure 7b with Figure 3, it can be seen that even with very low noise levels, the brute-force complexity can go beyond polynomial time. In such cases, the attacker runs at an advantage if only one of $\bar{\boldsymbol{s}}_1$ or $\bar{\boldsymbol{s}}_2$ has to be recovered.

6 Conclusion

In this work, we have demonstrated a side channel assisted forgery attack on the Dilithium signature scheme which is one of the leading candidates to the NIST call for standardization of post-quantum cryptography. Our attack proceeds in two stages, wherein the the partial secret-key $\bar{\boldsymbol{s}}_1$ is retrieved through a novel power analysis attack on the polynomial multiplier. We follow it up with a forgery signature scheme, wherein we show that it is indeed possible to forge signatures with only the knowledge of $\bar{\boldsymbol{s}}_1$ while bypassing the use of the other parts of the secret-key. We again stress that we neither claim a major break of the Dilithium signature scheme nor any reduction in security of the same. But, we show that the Dilithium scheme breaks on only the knowledge of $\bar{\boldsymbol{s}}_1$, which is otherwise not very intuitive given the fact that only a fraction of the LWE instance is revealed as the public-key. Thus the secrecy of $\bar{\boldsymbol{s}}_1$ is pivotal in ensuring the security of the Dilithium signature scheme.

References

1. Sedat Akleylek, Nina Bindel, Johannes Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. An efficient lattice-based signature scheme with provably secure instantiation. In *International Conference on Cryptology in Africa*, pages 44–60. Springer, 2016.
2. Shi Bai and Steven D Galbraith. An Improved Compression Technique for Signatures Based on Learning with Errors. In *CT-RSA*, volume 8366, pages 28–47, 2014.
3. Rami Barends, Julian Kelly, Anthony Megrant, Andrzej Veitia, Daniel Sank, Evan Jeffrey, Ted C White, Josh Mutus, Austin G Fowler, Brooks Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, 2014.
4. Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.
5. Nina Bindel, Johannes Buchmann, and Juliane Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on*, pages 63–77. IEEE, 2016.
6. Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, Gauss, and Reload–a cache attack on the BLISS lattice-based signature scheme. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 323–345. Springer, 2016.
7. Leon Groot Bruinderink and Peter Pessl. Differential Fault Attacks on Deterministic Lattice Signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3), 2018. `https://eprint.iacr.org/2018/355.pdf`.
8. Johannes Buchmann, Florian Göpfert, Tim Güneysu, Tobias Oder, and Thomas Pöppelmann. High-performance and lightweight lattice-based public-key encryption. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 2–9. ACM, 2016.

9. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In *Advances in Cryptology–CRYPTO 2013*, pages 40–56. Springer, 2013.
10. Léo Ducas, Eike Kiltz, Tancrde Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehl. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018.
11. Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Loop-abort faults on lattice-based Fiat-Shamir and hash-and-sign signatures. In *International Conference on Selected Areas in Cryptography*, pages 140–158. Springer, 2016.
12. Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1857–1874. ACM, 2017.
13. Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 530–547. Springer, 2012.
14. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *International Conference on Cryptographic Hardware and Embedded Systems*, volume 4, pages 119–132. Springer, 2004.
15. TP Harty, DTC Allcock, C J Ballance, L Guidoni, HA Janacek, NM Linke, DN Stacey, and DM Lucas. High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit. *Physical review letters*, 113(22):220501, 2014.
16. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 598–616. Springer, 2009.
17. Vadim Lyubashevsky, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS-Dilithium. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.
18. NIST. Post-Quantum Crypto Project. `http://csrc.nist.gov/groups/ST/post-quantum-crypto/`, 2016.
19. NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. `https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf`, 2016.
20. Peter Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In *INDOCRYPT 2016*, pages 153–170. Springer, 2016.
21. Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongSwan's Implementation of Post-Quantum Signatures. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1843–1855. ACM, 2017.
22. Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 353–370. Springer, 2014.

23. Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In *International Conference on Cryptology and Information Security in Latin America*, pages 139–158. Springer, 2012.
24. Thomas Pöppelmann and Tim Güneysu. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 2796–2799. IEEE, 2014.
25. Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. High-performance ideal lattice-based cryptography on 8-bit atxmega microcontrollers. In *International Conference on Cryptology and Information Security in Latin America*, pages 346–365. Springer, 2015.
26. John Preskill. Reliable quantum computers. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 454, pages 385–410. The Royal Society, 1998.
27. Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 513–533. Springer, 2017.
28. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 30–46. Springer, 2005.