Loop-Abort Faults on Lattice-Based Signatures and Key Exchange Protocols

Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi

Abstract—Although postquantum cryptography is of growing practical concern, not many works have been devoted to implementation security issues related to postquantum schemes.

In this paper, we look in particular at fault attacks against implementations of lattice-based signatures and key exchange protocols. For signature schemes, we are interesting both in Fiat–Shamir type constructions (particularly BLISS, but also GLP, PASSSing and Ring-TESLA) and in hash-and-sign schemes (particularly the GPV-based scheme of Ducas–Prest–Lyubashevsky). For key exchange protocols, we study the implementations of New Hope and Frodo. These schemes include essentially all practical lattice-based signatures and key exchange protocols, and achieve the best efficiency to date in both software and hardware. We present several fault attacks against those schemes that recover the entire key recovery with only a few faulty executions (sometimes only one), and discuss possible countermeasures to protect against such attacks.

Keywords: Fault Attacks, Digital Signatures, Postquantum Cryptography, Lattices, BLISS, GPV.

1 Introduction

1.1 Lattice-based cryptography

Recent progress in quantum computation [**?**], the NSA advisory memorandum recommending the transition away from Suite B and to postquantum cryptography [**?**], as well as the announcement of the NIST standardization process for postquantum cryptography [**?**] all suggest that research on postquantum schemes, which is already plentiful but mostly focused on theoretical constructions and asymptotic security, should increasingly take into account real world implementation issues.

Among all postquantum directions, lattice-based cryptography occupies a position of particular interest, as it relies on well-studied problems and comes with uniquely strong security guarantees, such as worst-case to average-case reductions [**?**]. Various works have also focused on improving the performance of lattice-based schemes, and actual implementation results suggest that properly optimized schemes may be competitive with, or even outperform, classical factoring and discrete logarithm-based cryptography.

The literature on the underlying number-theoretic problems of lattice-based cryptography is extensive (even though concrete bit security is not nearly as well understood as for factoring and discrete logarithms; in addition, ring-based schemes have recently been subjected to new families of attacks that might eventually reduce their security, especially in the postquantum setting). On the other hand, there is currently a distinct lack of cryptanalytic results on the *physical* security of implementations of lattice-based schemes (or in fact, postquantum schemes in general! [**?**]). It is well-known that physical attacks, particularly against public-key schemes, are often simpler, easier to mount and more devastating than attacks targeting underlying hardness assumptions: it is often the case so that a few bits of leakage or a few fault injections can reveal an entire secret key (the well-known attacks from [**?**], [**?**] are typical instances). We therefore deem it important to investigate how fault attacks may be leveraged to recover secret keys in the lattice-based setting, particularly against signature schemes as being probably the most likely primitive to be deployed in a real-world setting where fault attacks are relevant; they have also received the most attention in terms of efficient implementations both in hardware and software.

1.2 Implementations of lattice-based signatures

Efficient signature schemes are typically proved secure in the random oracle model, and can be roughly divided in two families: the hash-and-sign family (which includes schemes like FDH and PSS), as well as signatures based on identification schemes, using the so-called Fiat–Shamir heuristic or a variant thereof. Efficient lattice-based signatures can also be divided along those lines, as observed for example in the survey of practical lattice-based digital signature schemes presented by O'Neill and Güneysu at the NIST workshop on postquantum cryptography [**?**], [**?**].

The Fiat–Shamir family is the most developed, with a number of schemes coming with concrete implementations in software, and occasionally in hardware as well. Most schemes in that family follow Lyubashevsky's "Fiat–Shamir with aborts" paradigm [11], which uses rejection sampling to ensure that the underlying identification scheme

- *T. Espitau is PhD student at Sorbonne Universités, UPMC Univ Paris 06, LIP6.*
  E-mail: thomas.espitau@lip6.fr
- *P.-A. Fouque is Professor at Université de Rennes I and researcher at Institut Universitaire de France & IRISA.*
  E-mail: pierre-alain.fouque@univ-rennes1.fr
- *B. Gérard is Researcher at IRISA.*
  E-mail: benoit.gerard@irisa.fr
- *M. Tibouchi is Researcher at NTT Secure Platform Laboratories.*
  E-mail: tibouchi.mehdi@lab.ntt.co.jp

Manuscript received May 15, 2017.

achieves honest-verifier zero-knowledge. Among lattice-based schemes, the exemplar in that family is Lyubashevsky's scheme from EUROCRYPT 2012 [12]. It is, however, of limited efficiency, and had to be optimized to yield practical implementations. This was first carried out by Güneysu et al., who described an optimized hardware implementation of it at CHES 2012 [9], and then to a larger extent by Ducas et al. in their scheme BLISS [**?**], which includes a number of theoretical improvements and is the top-performing lattice-based signature. It was also implemented in hardware by Pöppelmann et al. [**?**]. Other schemes in that family include Hoffstein et al.'s PASSSign [10], which incorporates ideas from NTRU, and Akleylek et al.'s Ring-TESLA [**?**], which boasts a tight security reduction.

On the hash-and-sign side, there were a number of early proposals with heuristic security (and no actual security proofs), particularly GGH [**?**] and NTRUSign [**?**], but despite several attempts to patch them[1] they turned out to be insecure. A principled, provable approach to designing lattice-based hash-and-sign signatures was first described by Gentry, Peikert and Vaikuntanathan in [7], based on discrete Gaussian sampling over lattices. The resulting scheme, GPV, is rather inefficient, even when using faster techniques for lattice Gaussian sampling [**?**]. However, Ducas, Lyubashevsky and Prest [**?**] later showed how it could be optimized and instantiated over NTRU lattices to achieve a relatively efficient scheme with particularly short signature size. The DLP scheme is somewhat slower than BLISS in software, but still a good contender for practical lattice-based signatures, and seemingly the only one in the hash-and-sign family.

1.3 Implementations of lattice-based key exchange

In the last few years, very efficient lattice-based key exchange protocols have been proposed at several security conferences [**?**], [**?**], [**?**], [**?**] and some of them have been field tested by Microsoft and Google as alternatives to the prequantum key agreements in the TLS handshake protocol. This has shown that lattice-based key exchange protocols can be practical in many contexts, and offer credible alternatives to schemes like ECDH, incurring only a 50% performance penalty or so compared to elliptic curves.

The various lattice-based key exchange protocols have a similar structure, relying on Peikert's reconciliation mechanism [**?**] that allows the two parties to recover the *exact same* secret even if they both have a *noisy* version of the common secret. They mostly differ on the underlying lattice assumptions they are based on. Similarly to the signature setting, one can in particular distinguish between ring-based constructions, like NewHope [**?**], and constructions using standard lattices, like Frodo [**?**].

1.4 Our contributions

In this paper, we initiate the study of fault attacks against lattice-based signatures and key exchange protocols, and obtain attacks against all the practical schemes above-mentioned.

As noted previously, early lattice-based signature schemes with heuristic security have been broken using standard attacks [6], [8], [16] but recent constructions including [**?**], [**?**], [7], [11], [12] are provably secure, and cryptanalysis therefore requires a more powerful attack model. In this work we consider fault attacks.

We present two attacks on signatures, both using a similar type of faults which allows the attacker to cause a loop inside the signature generation algorithm to abort early. Successful loop-abort faults have been described many times in the literature, including against DSA [**?**] and pairing computations [**?**], and in our attacks they can be used to recover information about the private signing key. The underlying mathematical techniques used to actually recover the key, however, are quite different in the two attacks.

Our first attack applies to the schemes in the Fiat–Shamir family: we describe it against BLISS [**?**], [**?**], and show how it extends to GLP [9], PASSSign [10] and Ring-TESLA [**?**]. In that attack, we inject a fault in the loop that generates the random "commitment value" $\mathbf{y}$ of the sigma protocol associated with the Fiat–Shamir signature scheme. That commitment value is a random polynomial generated coefficient by coefficient, and an early loop abort causes it to have abnormally low degree, so that the protocol is no longer zero-knowledge. In fact, this will usually leak enough information that *a single faulty signature is enough to recover the entire signing key*. More specifically, we show that the faulty signature can be used to construct a point that is very close to a vector in a suitable integer lattice of moderate dimension, and such that the difference is essentially (a subset of) the signing key, which can thus be recovered using lattice reduction. We show that this attack can also be used against the New Hope and Frodo schemes.

Our second attack targets the GPV-based hash-and-sign signature scheme of Ducas et al. [**?**]. In that case, we consider early loop abort faults against the discrete Gaussian sampling in the secret trapdoor lattice used in signature generation. The early loop abort causes the signature to be a linear combination of the last few rows of the secret lattice. A few faulty signatures can then be used to recover the span of those rows, and using the special structure of the lattice, we can then use lattice reduction to find one of the rows up to sign, which is enough to completely reconstruct the secret key. In practice, if we can cause loop aborts after up to m iterations, we find that $m + 2$ *faulty signatures are enough for full key recovery* with high probability.

In addition, we also describe loop-abort fault attacks on the two protocols that represent the state of the art for lattice-based key exchange, namely NewHope [**?**] and Frodo [**?**]. Although those schemes have a completely different overall structure than the signature schemes mentioned above, they also involve the sampling of random Gaussian secrets coefficient by coefficient during each execution of the protocol. Injecting a fault that causes this random sampling loop to abort early causes abnormally "low-dimensional" secrets to be generated, and this yields a key recovery attack very similar to the one we mount on BLISS and other Fiat–Shamir signatures.

All of our attacks are supported by extensive mathe-

1. There is a provably secure scheme due to Aguilar et al. [**?**] that claims to "seal the leak on NTRUSign", but it actually turns the construction into a Fiat–Shamir type scheme, using rejection sampling à la Lyubashevsky.

matical simulations in Sage [?]. We also take a close look at the concrete software and hardware implementations of the schemes above, and discuss the concrete feasibility of injecting the required loop-abort faults in practice. We find the attacks to be highly realistic. Moreover, we demonstrate the practicality of those attacks by simulating them against the emulated execution of actual compiled code for the 32-bit SPARC processor LEON3, using a readily available fault simulation tool for that architecture.

Finally, we discuss several possible countermeasures to protect against our attacks.

1.5 Related work

To the best of our knowledge, the first previous work on fault attacks against lattice-based signatures, and in particular the only one mentioned in the survey of Taha and Eisenbarth [?], is the fault analysis work of Kamal and Youssef on NTRUSign [?]. It is, however, of limited interest since NTRUSign is known to be broken [4], [16]; it also suffers from a very low probability of success.

In 2016, concurrently with our work [?], Bindel, Buchmann and Krämer [?] described various fault attacks against Fiat–Shamir type signature schemes. Most of the attacks, however, are either in a relatively contrived model (targeting key generation), or require unrealistically many faults and are arguably straightforward (bypassing rejection sampling in signature generation or size/correctness checks in signature verification). One attack described in the paper can be seen as posing a serious threat, namely the one in [?, Sec. IV-B], but it amounts to a weaker variant of our Fiat–Shamir attack, using simple linear algebra rather than lattice reduction. As a result, it requires several hundred faulty signatures, whereas our attack needs only one.

Another interesting concurrent work is the recent cache attack against BLISS of Groot Bruinderink et al. [?]. It uses cache side-channels to extract information about the coefficients of the commitment polynomial $\mathbf{y}$, and then lattice reduction to recover the signing key based on that side-channel information. In that sense, it is similar to our Fiat–Shamir attack. However, since the nature of the information is quite different than in our setting, the mathematical techniques are also quite different. In particular, again, in contrast with our fault attack, that cache attack requires many signatures for a successful key recovery.

A preliminary version of this paper has been published at SAC 2016 in [?]. We extend this version by attacking a larger number of signature schemes, and generalizing our original attacks to the key exchange settings. Our discussion of the practical implementations of the faults we consider has also been revised and improved; in particular, the fault simulation on LEON3 compiled code is a novel contribution.

2 Description of the Lattice-Based Signature and Key Exchange Schemes

2.1 Notation

For any integer q, we represent the ring $\mathbb{Z}_q$ by $[-q/2, q/2) \cap \mathbb{Z}$. Vectors are considered as column vectors and will be written in bold lower case letters and matrices with upper case letters. By default, we will use the ℓ_2 Euclidean norm, $\|\mathbf{v}\|_2 = (\sum_i v_i^2)^{1/2}$ and ℓ_∞-norm as $\|\mathbf{v}\|_\infty = \max_i |v_i|$.

The Gaussian distribution with standard deviation $\sigma \in \mathbb{R}$ and center $c \in \mathbb{R}$ at $x \in \mathbb{R}$, is defined by $\rho_{c,\sigma}(x) = \exp\left(\frac{-(x-c)^2}{2\sigma^2}\right)$ and more generally by $\rho_{\mathbf{c},\sigma}(\mathbf{x}) = \exp\left(\frac{-(\mathbf{x}-\mathbf{c})^2}{2\sigma^2}\right)$ and when $\mathbf{c} = \mathbf{0}$, by $\rho_\sigma(\mathbf{x})$. The discrete Gaussian distribution over $\mathbb{Z}$ centered at $\mathbf{0}$ is defined by $D_\sigma(x) = \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ (or $D_{\mathbb{Z},\sigma}$) and more generally over $\mathbb{Z}^m$ by $D_\sigma^m(\mathbf{x}) = \rho_\sigma(\mathbf{x})/\rho_\sigma(\mathbb{Z}^m)$, where $\rho_\sigma(\mathbb{Z}^m) = \sum_{\mathbf{x}\in\mathbb{Z}^m} \rho_\sigma(\mathbf{x})$.

2.2 Description of BLISS

The BLISS signature scheme [?] is possibly the most efficient lattice-based signature scheme so far. It has been implemented in both software [?] and hardware [?], and boasts performance numbers comparable to classical factoring and discrete-logarithm based schemes. BLISS can be seen as a ring-based optimization of the earlier lattice-based scheme of Lyubashevsky [12], sharing the same "Fiat–Shamir with aborts" structure [11]. One can give a simplified description of the scheme as follows: the public key is an NTRU-like ratio of the form $\mathbf{a}_q = \mathbf{s}_2/\mathbf{s}_1 \bmod q$, where the signing key polynomials $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{R} = \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$ are small and sparse. To sign a message μ, one first generates commitment values $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{R}$ with normally distributed coefficients, and then computes a hash $\mathbf{c}$ of the message μ together with $\mathbf{u} = -\mathbf{a}_q\mathbf{y}_1 + \mathbf{y}_2 \bmod q$. The signature is then the triple $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, with $\mathbf{z}_i = \mathbf{y}_i + \mathbf{s}_i\mathbf{c}$, and there is rejection sampling to ensure that the distribution of $\mathbf{z}_i$ is independent of the secret key. Verification is possible because $\mathbf{u} = -\mathbf{a}_q\mathbf{z}_1 + \mathbf{z}_2 \bmod q$. The real BLISS scheme, described in full in Figure 1, includes several optimizations on top of the above description. In particular, to improve the repetition rate, it targets a bimodal Gaussian distribution for the $\mathbf{z}_i$'s, so there is a random sign flip in their definition. In addition, to reduce key size, the signature element $\mathbf{z}_2$ is actually transmitted in compressed form $\mathbf{z}_2^\dagger$, and accordingly the hash input includes only a compressed version of $\mathbf{u}$. These various optimizations are essentially irrelevant for our purposes.

2.3 Description of the GPV-based scheme of [?]

The second signature scheme we consider is the one proposed by Ducas, Lyubashevsky and Prest at ASIACRYPT 2014 [?]. It is an optimization using NTRU lattices of the GPV hash-and-sign signature scheme of Gentry, Peikert and Vaikuntanathan [7], and has been implemented in software by Prest [?]. As in GPV, the signing key is a "good" basis of a certain lattice Λ (with short, almost orthogonal vectors), and the public key is a "bad" basis of the same lattice (with longer vectors and a large orthogonality defect). To sign a message μ, one simply hashes it to obtain a vector $\mathbf{c}$ in the ambient space of Λ, and uses the good, secret basis to sample $\mathbf{v} \in \Lambda$ according to a discrete Gaussian distribution of small variance supported on Λ and centered at $\mathbf{c}$. That vector $\mathbf{v}$ is the signature; it is, in particular, a lattice point very close to $\mathbf{c}$. That property can be checked using the bad, public basis, but that basis is too large to sample such close vectors (this, combined with the fact that the discrete Gaussian

function KEYGEN()
  sample $\mathbf{f}, \mathbf{g} \in \mathcal{R} = \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$, uniformly with $\lceil \delta_1 n \rceil$ coefficients in $\{\pm 1\}$, $\lceil \delta_2 n \rceil$ coefficients in $\{\pm 2\}$ and other equal to zero
  $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^T \leftarrow (\mathbf{f}, 2\mathbf{g} + 1)^T$
  if $N_\kappa(\mathbf{S}) \geq C^2 \cdot 5 \cdot (\lceil \delta_1 n \rceil + 4\lceil \delta_2 n \rceil) \cdot \kappa$ **then restart**
  $\mathbf{a}_q = (2\mathbf{g} + 1)/\mathbf{f} \bmod q$ (**restart** if $\mathbf{f}$ is not invertible)
  return $(pk = \mathbf{a}_1, sk = \mathbf{S})$ where $\mathbf{a}_1 = 2\mathbf{a}_q \bmod 2q$
end function

function VERIFY($\mu, pk = \mathbf{a}_1, (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$)
  if $\|(\mathbf{z}_1, 2^d \cdot \mathbf{z}_2^\dagger)\|_2 > B_2$ **then reject**
  if $\|(\mathbf{z}_1, 2^d \cdot \mathbf{z}_2^\dagger)\|_\infty > B_\infty$ **then reject**
  accept iff $\mathbf{c} = H(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu)$
end function

function SIGN($\mu, pk = \mathbf{a}_1, sk = \mathbf{S}$)
  $\mathbf{y}_1 \leftarrow D^n_{\mathbb{Z},\sigma}$, $\mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$
  $\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$
  $\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$
  choose a random bit b
  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
  rejection sampling: **restart** to step 2 except with probability $1/\big(M \exp(-\|\mathbf{Sc}\|/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle / \sigma^2)\big)$
  $\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
end function

Fig. 1. Description of the BLISS signature scheme. The random oracle H takes its values in the set of polynomials in $\mathcal{R}$ with $0/1$ coefficients and Hamming weight exactly κ, for some small constant κ. The value ζ is defined as $\zeta \cdot (q-2) = 1 \bmod 2q$. The authors of [**?**] propose four different sets of parameters with security levels at least 128 bits. The interesting parameters for us are: $n = 512$, $q = 12289$, $\sigma \in \{215, 107, 250, 271\}$, $(\delta_1, \delta_2) \in \{(0.3, 0), (0.42, 0.03), (0.45, 0.06)\}$ and $\kappa \in \{23, 30, 39\}$. We refer to the original paper for other parameters and for the definition of notation like N_κ and $\lfloor \cdot \rceil_d$, as they are not relevant for our attack. The instruction in red (sampling of $\mathbf{y}_1$) is where we introduce our faults.

function KEYGEN(n, q)
  $\mathbf{f} \leftarrow D^n_{\sigma_0}$, $\mathbf{g} \leftarrow D^n_{\sigma_0}$ ▷ $\sigma_0 = 1.17\sqrt{q/2n}$
  if $\|(\mathbf{g}, -\mathbf{f})\|_2 > \sigma$ **then restart** ▷ $\sigma = 1.17\sqrt{q}$
  if $\left\| \left(\frac{q\bar{\mathbf{f}}}{\mathbf{f}\bar{\mathbf{f}} + \mathbf{g}\bar{\mathbf{g}}}, \frac{q\bar{\mathbf{g}}}{\mathbf{f}\bar{\mathbf{f}} + \mathbf{g}\bar{\mathbf{g}}} \right) \right\|_2 > \sigma$ **then restart**
  using the extended Euclidean algorithm, compute $\rho_f, \rho_g \in \mathcal{R}$ and $R_f, R_g \in \mathbb{Z}$ s.t. $\rho_f \cdot \mathbf{f} = R_f \bmod \mathbf{x}^n + 1$ and $\rho_g \cdot \mathbf{g} = R_g \bmod \mathbf{x}^n + 1$
  if $\gcd(R_f, R_g) \neq 1$ or $\gcd(R_f, q) \neq 1$ **then restart**
  using the extended Euclidean algorithm, compute $u, v \in \mathbb{Z}$ s.t. $u \cdot R_f + v \cdot R_g = 1$
  $\mathbf{F} \leftarrow qv\rho_g$, $\mathbf{G} \leftarrow -qu\rho_f$
  repeat
    $\mathbf{k} \leftarrow \left\lfloor \frac{\mathbf{F} \cdot \bar{\mathbf{f}} + \mathbf{G} \cdot \bar{\mathbf{f}}}{\mathbf{f}\bar{\mathbf{f}} + \mathbf{g}\bar{\mathbf{g}}} \right\rceil \in \mathcal{R}$
    $\mathbf{F} \leftarrow \mathbf{F} - \mathbf{k} \cdot \mathbf{f}, \mathbf{G} \leftarrow \mathbf{G} - \mathbf{k} \cdot \mathbf{g}$
  until $\mathbf{k}$=0
  $\mathbf{h} \leftarrow \mathbf{g} \cdot \mathbf{f}^{-1} \bmod q$
  $\mathbf{B} \leftarrow \begin{pmatrix} \mathbf{M_g} & -\mathbf{M_f} \\ \mathbf{M_G} & -\mathbf{M_F} \end{pmatrix} \in \mathbb{Z}^{2n \times 2n}$ ▷ short lattice basis
  return $sk = \mathbf{B}$, $pk = \mathbf{h}$
end function

function GAUSSIANSAMPLER($\mathbf{B}, \sigma, \mathbf{c}$) ▷ $\mathbf{b}_i$ (resp. $\widetilde{\mathbf{b}}_i$) denote the rows of $\mathbf{B}$ (resp. of its Gram–Schmidt matrix $\widetilde{\mathbf{B}}$)
  $\mathbf{v} \leftarrow \mathbf{0}$
  for $i = 2n$ **down to** 1 **do**
    $c' \leftarrow \langle \mathbf{c}, \widetilde{\mathbf{b}_i} \rangle / \|\widetilde{\mathbf{b}_i}\|_2^2$
    $\sigma' \leftarrow \sigma / \|\mathbf{b}_i\|_2$
    $r \leftarrow D_{\mathbb{Z},\sigma',c'}$
    $\mathbf{c} \leftarrow \mathbf{c} - r\mathbf{b}_i$ and $\mathbf{v} \leftarrow \mathbf{v} + r\mathbf{b}_i$
  end for
  return $\mathbf{v}$ ▷ $\mathbf{v}$ sampled according to the lattice Gaussian distribution $D_{\Lambda,\sigma,\mathbf{c}}$
end function

function SIGN($\mu, sk = \mathbf{B}$)
  $\mathbf{c} \leftarrow H(\mu) \in \mathbb{Z}_q^n$
  $(\mathbf{y}, \mathbf{z}) \leftarrow (\mathbf{c}, \mathbf{0}) - \text{GAUSSIANSAMPLER}(\mathbf{B}, \sigma, (\mathbf{c}, \mathbf{0}))$
    ▷ $\mathbf{y}, \mathbf{z}$ are short and satisfy $\mathbf{y} + \mathbf{z} \cdot \mathbf{h} = \mathbf{c} \bmod q$
  return $\mathbf{z}$
end function

function VERIFY($\mu, pk = \mathbf{h}, \mathbf{z}$)
  accept iff $\|\mathbf{z}\|_2 + \|H(\mu) - \mathbf{z} \cdot \mathbf{h}\|_2 \leq \sigma\sqrt{2n}$
end function

Fig. 2. Description of the GPV-based signature scheme of Ducas–Lyubashevsky–Prest. The random oracle H takes its values in $\mathbb{Z}_q^n$. We denote by $\mathbf{f} \mapsto \bar{\mathbf{f}}$ the conjugation involution of $\mathcal{R} = \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$, i.e. for $\mathbf{f} = \sum_{i=0}^{n-1} f_i x^i$, $\bar{\mathbf{f}} = f_0 - \sum_{i=1}^{n-1} f_{n-i} x^i$. $\mathbf{M_a}$ represents the matrix of the multiplication by $\mathbf{a}$ in the polynomial basis of $\mathcal{R}$, which is anticirculant of dimension n. For 128 bits of security, the authors of [**?**] recommend the parameters $n = 256$ and $q \approx 2^{10}$. The constant 1.17 is an approximation of $\sqrt{e/2}$. The steps in red (main loop of the Gaussian sampler) is where we introduce our faults.

leaks no information about the secret basis, is what makes it possible to prove security). The actual scheme of Ducas–Lyubashevsky–Prest, described in Figure 2, uses a lattice of the same form as NTRU: $\Lambda = \{(\mathbf{y}, \mathbf{z}) \in \mathcal{R}^2 \mid \mathbf{y} + \mathbf{z} \cdot \mathbf{h} = 0\}$, where the public key $\mathbf{h}$ is again a ratio $\mathbf{g}/\mathbf{f} \bmod q$ of small, sparse polynomials in $\mathcal{R} = \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$. The use of such a lattice yields a very compact representation of the keys, and makes it possible to compress the signature as well by publishing only the second component of the sampled vector $\mathbf{v}$. As a result, this hash-and-sign scheme is very space efficient (even more than BLISS). However, the use of lattice Gaussian sampling makes signature generation significantly slower than BLISS at similar security levels.

Alice		Bob
$seed \leftarrow^{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \mathsf{SHAKE\text{-}128}(seed)$		
$\mathbf{s}, \mathbf{e} \leftarrow \psi_{16}^{n}$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \leftarrow \psi_{16}^{n}$
$\mathbf{b} \leftarrow \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \mathsf{SHAKE\text{-}128}(seed)$
		$\mathbf{u} \leftarrow \mathbf{a} \cdot \mathbf{s}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{b} \cdot \mathbf{s}' + \mathbf{e}''$
$\mathbf{v}' \leftarrow \mathbf{u} \cdot \mathbf{s}$	$\xleftarrow{(\mathbf{u}, \mathbf{r})}$	$\mathbf{r} \leftarrow^{\$} \mathrm{HelpRec}(\mathbf{v})$
$\nu \leftarrow \mathrm{Rec}(\mathbf{v}', \mathbf{r})$		$\nu \leftarrow \mathrm{Rec}(\mathbf{v}, \mathbf{r})$
$\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$		$\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$

Fig. 3. Description of the NewHope scheme. The security parameters given are $q = 12289 < 2^{14}, n = 1024$. The REC and HELPREC subprocedures are encoding and decoding functions between bits and coordinates in a small dimension lattice, as fully analyzed in [**?**]. Note that computations are taken $\mathrm{mod} q$

Alice		Bob
$seed \leftarrow^{\$} \mathcal{U}(\{0,1\}^{s})$		
$\mathbf{A} \leftarrow \mathrm{Gen}(seed)$		
$\mathbf{S}, \mathbf{E} \leftarrow \chi(\mathbb{Z}_q^{n \times \bar{n}})$		$\mathbf{S}', \mathbf{E}', \mathbf{E}'' \leftarrow^{\$} \chi(\mathbb{Z}_q^{n \times \bar{n}})$
$\mathbf{B} \leftarrow \mathbf{A} \cdot \mathbf{S} + \mathbf{E}$	$\xrightarrow{(\mathbf{B}, seed)}$	$\mathbf{A} \leftarrow \mathrm{Gen}(seed)$
		$\mathbf{B}' \leftarrow \mathbf{S}' \cdot \mathbf{A} + \mathbf{E}'$
		$\mathbf{V} \leftarrow \mathbf{S}' \cdot \mathbf{B} + \mathbf{E}''$
	$\xleftarrow{(\mathbf{B}', \mathbf{C})}$	$\mathbf{C} \leftarrow \langle \mathbf{V} \rangle_{2^B}$
$K \leftarrow \mathrm{Rec}(\mathbf{B}' \cdot \mathbf{S}, \mathbf{C})$		$K \leftarrow \lfloor \mathbf{V} \rceil_{2^B}$

Fig. 4. Description of the Frodo scheme with parameters (n, q, χ), and protocol specific parameters $\bar{n}, \bar{m}, \bar{B} \in \mathbb{Z}$. The matrix $\mathbf{A} \in \mathbb{Z}^{n \times \bar{n}}$ is generated from seed via a pseudo-random function GEN. Recommended parameters for instanciation are $n = 752, q = 2^{15}$, χ an approximate discrete Gaussian distribution of variance 1.75. Note that computations are taken $\mathrm{mod} q$.

2.4 Description of NewHope

The NewHope key exchange protocol is a variation on the protocol of Ding, Xie, and Lini, itself based on the so-called Peikert tweak, getting the 2016 Internet Defense Prize award of USENIX conference [**?**]. Its major improvements w.r.t. its successor are both theoretical and practical. On the one hand the authors proposed a refined analysis of the failure probability of the exchange and its resistance towards quantum adversaries. On the other hand, various tweaks were introduced in the design of the scheme: like Peikert, the authors use the KEM framework, defined by the algorithms (Setup, Gen, Encaps, Decaps); after a successful protocol run both parties share an ephemeral secret key that can be used to protect further communication: the reconciliation allows both parties to derive the session key from an approximately agreed pseudorandom ring element. On Alice's side, this element can be written as $\mathbf{u} \cdot \mathbf{s} = \mathbf{a} \cdot \mathbf{s} \cdot \mathbf{s} + \mathbf{e} \cdot \mathbf{s}$ and on Bob's side: $\mathbf{s} \cdot \mathbf{v} = \mathbf{b} \cdot \mathbf{s} + \mathbf{e} = \mathbf{a} \cdot \mathbf{s} \cdot \mathbf{s} + \mathbf{e} \cdot \mathbf{s} + \mathbf{e}$, where $\mathbf{s}, \mathbf{s}'$ are the respective secrets of Alice and Bob, taken as element in the convolution ring $\mathcal{R}$. The full outline of the NEW HOPE protocol is given in Figure 3.

2.5 Description of Frodo

The second key exchange scheme has been proposed by Bos et al. at CCS 2016 [**?**]. This scheme is a practical demonstration of an efficient lattice scheme based on the hardness of LWE (on the contrary of New Hope, which relies on the hardness of the *Ring*-LWE problem). Its performances have been measured in a "real-world" setting by benchmarking its implantation within the TLS protocol, when coupled with ECDSA certificates. Like in New Hope key exchange, the respective reconciliation elements can be written as $\mathbf{u} \cdot \mathbf{s} = \mathbf{a} \cdot \mathbf{s} \cdot \mathbf{s}' + \mathbf{e}' \cdot \mathbf{s}$ and $\mathbf{s} \cdot \mathbf{v} = \mathbf{b} \cdot \mathbf{s}' + \mathbf{e}'' = \mathbf{a} \cdot \mathbf{s} \cdot \mathbf{s}' + \mathbf{e} \cdot \mathbf{s}' + \mathbf{e}''$ for Alice and Bob, but on the contrary of this latter scheme, the secrets are no more elements of the convolution ring $\mathcal{R}$, but vectors of integers. The full outline of the NEW HOPE protocol is given in Figure 4.

3 Attack on Fiat–Shamir type lattice-based signatures

The first fault attack that we consider targets the lattice-based signature schemes of Fiat–Shamir type, and specifically the generation of the random "commitment" element in the underlying sigma protocols, which is denoted by $\mathbf{y}$ in our descriptions. That element consists of one or several polynomials generated coefficient by coefficient, and the idea of the attack is to introduce a fault in that random sampling to obtain a polynomial of abnormally small degree, in which case signatures will leak information about the private signing key. For simplicity's sake, we introduce the attack against BLISS in particular, but it works against the other Fiat–Shamir type schemes (GLP, PASSSign and Ring-TESLA) with almost no changes: see the full version of this paper [**?**] for details.

In BLISS, the commitment element actually consists of two polynomials $(\mathbf{y}_1, \mathbf{y}_2)$, and it suffices to attack $\mathbf{y}_1$. Intuitively, $\mathbf{y}_1$ should mask the secret key element $\mathbf{s}_1$ in the relation $\mathbf{z}_1 = \pm \mathbf{s}_1 \mathbf{c} + \mathbf{y}_1$, and therefore modifying the distribution of $\mathbf{y}_1$ should cause some information about $\mathbf{s}$ to leak in signatures. The actual picture in the Fiat–Shamir with aborts paradigm is in fact slightly different (namely, rejection sampling ensures that the distribution of $\mathbf{z}_1$ is independent of $\mathbf{s}_1$, but only does so under the assumption that $\mathbf{y}_1$ follows the correct distribution), but the end result is the same: perturbing the generation of $\mathbf{y}_1$ should lead to secret key leakage.

Concretely speaking, in BLISS, $\mathbf{y}_1 \in \mathcal{R}_q$ is a ring element generated according to a discrete Gaussian distribution[2], and that generation is typically carried out coefficient by coefficient in the polynomial representation. Therefore, if we can use faults to cause an early termination of that generation process, we should obtain signatures in which the element $\mathbf{y}_1$ is actually a low-degree polynomial. If the

2. In the other Fiat–Shamir schemes such as [9], the distribution of each coefficient is uniform in some interval rather than Gaussian, but this doesn't affect our attack strategy at all.

degree is low enough, we will see that this reveals the whole secret key right away, from a single faulty signature!

Indeed, suppose that we can obtain a faulty signature obtained by forcing a termination of the loop for sampling $\mathbf{y}_1$ after the m-th iteration, with $m \ll n$. Then, the resulting polynomial $\mathbf{y}_1$ is of degree at most $m-1$. As part of the faulty signature, we get the pair $(\mathbf{c}, \mathbf{z}_1)$ with $\mathbf{z}_1 = (-1)^b \mathbf{s}_1 \mathbf{c} + \mathbf{y}_1$. Without loss of generality, we may assume that $b = 0$ (we will recover the whole secret key only up to sign, but in BLISS, $(\mathbf{s}_1, \mathbf{s}_2)$ and $(-\mathbf{s_1}, -\mathbf{s}_2)$ are clearly equivalent secret keys). Moreover, with high probability, $\mathbf{c}$ is invertible: if we heuristically assume that $\mathbf{c}$ behaves like a random element of the ring from that standpoint, we expect it to be the case with probability about $(1-1/q)^n$, which is over 95% for all proposed BLISS parameters. We thus get an equation of the form:

$$\mathbf{c}^{-1}\mathbf{z}_1 - \mathbf{s}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 \equiv \sum_{i=0}^{m-1} y_{1,i}\mathbf{c}^{-1}\mathbf{x}^i \pmod{q} \quad (1)$$

Thus, the vector $\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1$ is very close to the sublattice of $\mathbb{Z}^n$ generated by $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i \bmod q$ for $i = 0, \ldots, m-1$ and $q\mathbb{Z}^n$, and the difference should be $\mathbf{s}_1$.

The previous lattice is of full rank in $\mathbb{Z}^n$, so the dimension is too large to apply lattice reduction directly. However, the relation given by equation (1) also holds for all subsets of indices. More precisely, let I be a subset of $\{0, \ldots, n-1\}$ of cardinality ℓ, and $\varphi_I \colon \mathbb{Z}^n \to \mathbb{Z}^I$ be the projection $(u_i)_{0 \le i < n} \mapsto (u_i)_{i \in I}$. Then we also have that $\varphi_I(\mathbf{z}_1)$ is a close vector to the sublattice L_I of $\mathbb{Z}^I$ generated by $q\mathbb{Z}^I$ and the images under φ_I of the $\mathbf{w}_i$'s; and the difference should be $\varphi_I(\mathbf{s}_1)$.

Equivalently, using Babai's nearest plane approach to the closest vector problem, we hope to show that $(\varphi_I(\mathbf{s}_1), B)$, for a suitably chosen positive constant B, is the shortest vector in the sublattice L'_I of $\mathbb{Z}^I \times \mathbb{Z}$ generated by $(\varphi_I(\mathbf{v}), B)$ as well as the vectors $(\varphi_I(\mathbf{w_i}), 0)$ and $q\mathbb{Z}^I \times \{0\}$.

The volume of L'_I is given by:

$$\mathrm{vol}(L'_I) = B \cdot \mathrm{vol}(L_I) = B \cdot \frac{\mathrm{vol}(q\mathbb{Z}^I)}{[L_I : q\mathbb{Z}^I]} = Bq^{\ell - r}$$

where r is the rank of the family $(\varphi_I(\mathbf{w}_0), \ldots, \varphi_I(\mathbf{w}_{m-1}))$ in $\mathbb{Z}_q^I$, which is at most m. Hence $\mathrm{vol}(L'_I) \ge Bq^{\ell-m}$, and the Gaussian heuristic predicts that the shortest vector should be of norm:

$$\lambda_I \approx \sqrt{\frac{\ell+1}{2\pi e}} \cdot \mathrm{vol}(L'_I)^{1/(\ell+1)} \gtrsim \sqrt{\frac{\ell+1}{2\pi e}} \cdot B^{1/(\ell+1)} q^{1-(m+1)/(\ell+1)}$$

Thus, we expect that $(\varphi_I(\mathbf{s}_1), B)$ will actually be the shortest vector of L'_I provided that its norm is significantly smaller than this bound λ_I. Now $\varphi_I(\mathbf{s}_1)$ has roughly $\delta_1 \ell$ entries equal to ± 1, $\delta_2 \ell$ entries equal to ± 2 and the rest are zeroes; therefore, the norm of $(\varphi_I(\mathbf{s}_1), B)$ is around $\sqrt{(\delta_1 + 4\delta_2)\ell + B^2}$. Let us choose $B = \lceil \sqrt{\delta_1 + 4\delta_2} \rceil$. The condition for $\mathbf{s}_1$ to be the shortest vector L_I can thus be written as:

$$\sqrt{(\delta_1 + 4\delta_2) \cdot (\ell+1)} \ll \sqrt{\frac{\ell+1}{2\pi e}} \cdot B^{1/(\ell+1)} q^{1-(m+1)/(\ell+1)}$$

or equivalently:

$$\ell + 1 \gtrsim \frac{m + 1 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}. \quad (2)$$

The denominator of the right-hand side of (6) ranges from about 0.91 for the BLISS–I and BLISS–II parameter sets down to about 0.87 for BLISS–IV. In all cases, we thus expect to recover $\varphi_I(\mathbf{s}_1)$ if we can solve the shortest vector problem in a lattice of dimension slightly larger than m. This is quite feasible with the LLL algorithm for m up to about 50, and with BKZ for m up to 100 or so.

To complete the attack, it suffices to apply the above to a family of subsets I of $\{0, \ldots, n-1\}$ covering the whole set of indices, which reveals the entire vector $\mathbf{s}_1$. The second component of the secret key is then obtained as $\mathbf{s}_2 = \mathbf{a}_1 \mathbf{s}_1 / 2 \bmod q$.

Simulations using our Sage implementation confirm the theoretical estimates, and show that full key recovery can be achieved in practice in a time ranging from a few seconds to a few hours depending on m. Detailed experimental results are reported in Table 1.

Remark 1. A variant of that attack which is possibly slightly simpler consists in observing that $\varphi_I(\mathbf{s}_1)$ should be the shortest vector in the lattice generated by L_I and $\varphi_I(\mathbf{v})$. The bound on the lattice dimension becomes essentially the same as (6). The drawback of that approach, however, is that we obtain each $\varphi_I(\mathbf{s}_1)$ up to sign, and so one needs to use overlapping subsets I to ensure the consistency of those signs.

Remark 2. Note that a single *faulty* signature is enough to recover the entire secret key with this attack, a successful key recovery may require several *fault injections*. This is due to rejection sampling: after a faulty $\mathbf{y}_1$ is generated, the whole signature may be thrown away in the rejection step. On average, the fault attacker may thus need to inject the same number of faults as the repetition rate of the scheme, which is a small constant ranging from 1.6 to 7.4 depending on chosen parameters [**?**], and even smaller with the improved analysis of BLISS–B [**?**].

Remark 3. Finally, we note that in certain hardware settings, fault injection may yield a faulty value of $\mathbf{y}_1$ in which all coefficients upwards of a certain degree bound are non zero but equal to a common constant (see the discussion in Section 7.3). Our attack adapts to that setting in a straightforward way: that simply means that $\mathbf{y}_1$ is a linear combination of the $\mathbf{x}^i$ for small i and of the all-one vector $(1, \ldots, 1)$, so it suffices to add that vector to the set of lattice generators.

4 Attack on Hash-and-Sign Type Lattice-Based Signatures

Our second attack targets the practical hash-and-sign signature scheme of Ducas, Lyubashevsky and Prest [**?**], which is based on GPV-style lattice trapdoors. More precisely, the faults we consider are again early loop aborts, this time in the lattice-point Gaussian sampling routine used in signature generation.

TABLE 1
Experimental success rate of the attack and average CPU time for key recovery for several values of m, the iteration after which the loop-abort fault is injected. We attack the BLISS–II parameter set $(n, q, \sigma, \delta_1, \delta_2, \kappa) = (512, 12289, 10, 0.3, 0, 23)$ from [**?**]. Since the choice of ℓ has no effect on the concrete fault injection (e.g. it does not affect the required number of faulty signatures, which is always 1), we did not attempt to optimize it very closely. The simulation was carried out using our Sage implementation on a single core of an Intel Xeon E5-2697v3 workstation, using 100 trial runs for each value of m.

Fault after iteration number $m =$	2	5	10	20	40	60	80	100
Theoretical minimum dimension $\ell_{\min}$	3	6	11	22	44	66	88	110
Dimension ℓ in our experiment	3	6	12	24	50	80	110	150
Lattice reduction algorithm	LLL	LLL	LLL	LLL	BKZ–20	BKZ–25	BKZ–25	BKZ–25
Success probability (%)	100	99	100	100	100	100	100	98
Avg. CPU time to recover ℓ coeffs. (s)	0.002	0.005	0.022	0.23	7.3	119	941	33655
Avg. CPU time for full key recovery	0.5 s	0.5 s	1 s	5 s	80 s	14 min	80 min	38 h

4.1 Description of the attack

The attack can be described as follows. A correctly generated signature element is of the form $\mathbf{z} = \mathbf{R} \cdot \mathbf{f} + \mathbf{r} \cdot \mathbf{F} \in \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$, where the short polynomials $\mathbf{f}$ and $\mathbf{F}$ are components of the secret key, and $\mathbf{r}, \mathbf{R}$ are short random polynomials sampled in such a way that $\mathbf{z}$ follows a suitable Gaussian distribution. In fact, $\mathbf{r}, \mathbf{R}$ are generated coefficient by coefficient, in a single loop with $2n$ iterations, going from the top-degree coefficient of $\mathbf{r}$ down to the constant coefficient of $\mathbf{R}$.

Therefore, if we inject a fault aborting the loop after $m \leq n$ iterations (in the first half of the loop), the resulting signature simply has the form:

$$\mathbf{z} = r_0 \mathbf{x}^{n-1} \mathbf{F} + r_1 \mathbf{x}^{n-2} \mathbf{F} + \cdots + r_{m-1} \mathbf{x}^{n-m} \mathbf{F}.$$

Any such faulty signature is, in particular, in the lattice L of rank m generated by the vectors $\mathbf{x}^{n-i}\mathbf{F}$, $i = 1, \ldots, m$, in $\mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$.

Suppose then that we obtain several signatures $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(\ell)}$ of the previous form. If ℓ is large enough (slightly more than m is sufficient; see §4.2 below for an analysis of success probability depending on ℓ), the corresponding vectors will then generate the lattice L. Assuming the lattice dimension is not too large, we should then be able to use lattice reduction to recover a shortest vector in L, which is expected to be one of the signed shifts $\pm\mathbf{x}^{n-i}\mathbf{F}$, $i = 1, \ldots, m$, since the polynomial $\mathbf{F}$ is constructed in a such a way as to make it quite short relative to the Gram–Schmidt norm of the ideal lattice it generates. Hence, we can recover $\mathbf{F}$ among a small set of at most $2m$ candidates.

And recovering $\mathbf{F}$ is actually sufficient to reconstruct the entire secret key $(\mathbf{f}, \mathbf{g}, \mathbf{F}, \mathbf{G})$, and hence completely break the scheme. This is due to the particular structure of the NTRU lattice. On the one hand, $\mathbf{G}$ is linked to $\mathbf{F}$ via the public key polynomial $\mathbf{h}$: $\mathbf{G} = \mathbf{F} \cdot \mathbf{h} \bmod q$, so we obtain it directly. On the other hand, the basis completion algorithm of Hoffstein et al. [**?**] allows to recover the pair $(\mathbf{f}, \mathbf{g})$ from $(\mathbf{F}, \mathbf{G})$ via the defining relation $\mathbf{f} \cdot \mathbf{G} - \mathbf{g} \cdot \mathbf{F} = q$. This is actually used in the opposite direction in the key generation algorithm of the scheme of Ducas et al. (i.e. they construct $(\mathbf{F}, \mathbf{G})$ from $(\mathbf{f}, \mathbf{g})$: see steps 5–12 of KeyGen in Figure 2), but applying [**?**, Theorem 1], the technique is easily seen to work in both ways.

Moreover, if we start from a polynomial of the form $\zeta\mathbf{F}$ where ζ is of the form $\pm\mathbf{x}^\alpha$, then applying the previous steps yields the quadruple $(\zeta\mathbf{f}, \zeta\mathbf{g}, \zeta\mathbf{F}, \zeta\mathbf{G})$, which is also a valid secret key equivalent to $(\mathbf{f}, \mathbf{g}, \mathbf{F}, \mathbf{G})$, in the sense that signing with either keys produces signatures with exactly the same distributions. Thus, we don't even need to carry out an exhaustive search on several possible values of $\mathbf{F}$ after the lattice reduction step: it suffices to use the first vector of the reduced basis directly.

4.2 How many faults do we need?

Let us analyze the probability of success of the attack depending on the iteration m at which the iteration is inserted and the number $\ell > m$ of faulty signatures $\mathbf{z}^{(i)}$ available. As we have seen, a sufficient condition for the attack to succeed (provided that our lattice reduction algorithm actually finds a shortest vector) is that the ℓ faulty signatures generate the rank-m lattice L defined above. This is not actually necessary (the attack works as soon as *one* of the shifts of $\mathbf{F}$ is in sub-lattice generated by the signatures, rather than all of them), but we will be content with a lower bound on the probability of success.

Now, that condition is equivalent to saying that the vectors $(r_0^{(i)}, \ldots, r_{m-1}^{(i)}) \in \mathbb{Z}^m$ (sampled according to the distribution given by the GPV algorithm) that define the faulty signatures:

$$\mathbf{z}^{(i)} = r_0^{(i)} \mathbf{x}^{n-1} \mathbf{F} + \cdots + r_{m-1}^{(i)} \mathbf{x}^{n-m} \mathbf{F}$$

generate the whole integer lattice $\mathbb{Z}^m$. But the probability that $\ell > m$ random vectors generate $\mathbb{Z}^m$ has been computed by Maze, Rosenthal and Wagner [**?**] (see also [**?**]), and is asymptotically equal to $\prod_{k=\ell-m+1}^{\ell} \zeta(k)^{-1}$. In particular, if $\ell = m + d$ for some integer d, it is bounded below by:

$$p_d = \prod_{k=d+1}^{+\infty} \frac{1}{\zeta(k)}.$$

Thus, if we take $\ell = m + 1$ (resp. $\ell = m + 2$, $\ell = m + 3$), we expect the attack to succeed with probability at least $p_1 \approx 43\%$ (resp. $p_2 \approx 71\%$, $p_3 \approx 86\%$).

As shown in Table 2, this is well verified in practice (and the lower bound is in fact quite pessimistic). Moreover, the attack is quite fast even for relatively large values of m: only a couple of minutes for full key recovery for $m = 100$.

TABLE 2
Experimental success probability of the attack and average CPU time for key recovery for several values of m, the iteration after which the loop-abort fault is injected. We consider the attack with $\ell = m + 1$ and $\ell = m + 2$ faulty signatures. The attacked parameters are $(n, q) = (256, 1021)$ as suggested in [**?**] for signatures. The simulation was carried out using our Sage implementation (see the full version of this paper [**?**]) on a single core of an Intel Xeon E5-2697v3 workstation, using 100 trial runs for each pair (ℓ, m).

Fault after iteration number $m =$	2	5	10	20	40	60	80	100
Lattice reduction algorithm	LLL	LLL	LLL	LLL	LLL	LLL	BKZ–20	BKZ–20
Success probability for $\ell = m + 1$ (%)	75	77	90	93	94	94	95	95
Avg. CPU time for $\ell = m + 1$ (s)	0.001	0.003	0.016	0.19	2.1	8.1	21.7	104
Success probability for $\ell = m + 2$ (%)	89	95	100	100	99	99	100	100
Avg. CPU time for $\ell = m + 2$ (s)	0.001	0.003	0.017	0.19	2.1	8.2	21.6	146

5 Attack on New Hope Key Exchange

We present an attack against the NewHope key exchange protocol, and more specifically the generation of the "commitment" elements $\mathbf{e}, \mathbf{e}'$. By the inherent symmetry of the protocol, we describe the attack when mounted on Alice's side. The adaptation to Bob's side is then straightforward.

In this scheme, the commitment element consists of a polynomial $\mathbf{e}$, which acts intuitively as an additive mask to the secret key element $\mathbf{s}$ in the relation:

$$\mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}.$$

As a consequence, tampering the distribution of $\mathbf{e}$ should cause some information leakage in when sending the element $\mathbf{b}$ to Bob.

More formally, $\mathbf{e} \in \mathcal{R}_q$ is a ring element drawn under the center binomial distribution ψ_{16}.[3]. Its generation is typically carried out coefficient by coefficient in a polynomial representation. Thus, if one can use faults to cause an early termination of that generation process, we should obtain elements in which the element $\mathbf{e}$ is actually a low-degree polynomial. If the degree is low enough, we will see that this reveals the whole secret key right away, from a single faulty element!

Indeed, suppose that we can obtain a faulty element $\mathbf{b}$ obtained by forcing a termination of the loop for sampling $\mathbf{e}$ after the m-th iteration, with $m \ll n$, that is, the resulting polynomial $\mathbf{e}$ is of degree at most $m - 1$. As part of the commitment message, we get the pair $(seed, \mathbf{b})$, and since the generation of $\mathbf{a}$ is both deterministic and public, the pair $(\mathbf{a}, \mathbf{b})$ with

$$\mathbf{b} = \mathbf{s} \cdot \mathbf{a} + \mathbf{e}.$$

With high probability, $\mathbf{a}$ is invertible: if we heuristically assume that $\mathbf{a}$ behaves like a random element of the ring from that standpoint, we expect it to be the case with probability about $(1 - 1/q)^n$, which is slightly over 92% for the proposed NewHope parameters. We thus get an equation of the form:

$$\mathbf{a}^{-1}\mathbf{b} \equiv \mathbf{s} + \mathbf{a}^{-1}\mathbf{e} \equiv \mathbf{s} + \underbrace{\sum_{i=0}^{m-1} e_i \mathbf{a}^{-1}\mathbf{x}^i}_{\in \mathrm{Span}_{\mathbb{Z}}((\mathbf{a}^{-1}\mathbf{x}^i)_{0 \le i \le m-1})} \pmod q \quad (3)$$

3. On the contrary of signatures schemes, where zero-knowledgness is required, one can use different error distributions than the Discrete Gaussian.

for integers $e_0, \ldots, e_{m-1}$. Thus, the vector $\mathbf{v} = \mathbf{a}^{-1}\mathbf{b}$ is close to the sublattice $\mathcal{L}$ of $\mathbb{Z}^n$ generated by the elements $\mathbf{w}_i = \mathbf{a}^{-1}\mathbf{x}^i \bmod q$ for $i = 0, \ldots, m-1$ and $q\mathbb{Z}^n$. Then the difference between $\mathcal{L}$ and $\mathbf{v}$ should be precisely $\mathbf{s}$. The rest of the analysis is similar to the attack against BLISS. The main difference lies in the fact that $\mathbf{s}$ is sampled according to the n-dimensional centered binomial distribution of parameter $k = 16$ instead of a discrete Gaussian distribution. All in all the computations makes arise the following condition on ℓ:

$$\ell + 1 \gtrsim \frac{m + 1 + \frac{\log \sqrt{k}/2}{\log q}}{1 - \frac{\log \sqrt{k}/2\sqrt{2\pi e}}{\log q}}. \quad (4)$$

The denominator of the right-hand side of (6) is roughly about 0.70 for the parameters sets, We thus expect to recover $\varphi_I(\mathbf{s})$ — we recall that ϕ_I denotes the projection operator on a subset $I \subset \{1, \ldots, n\}$ — if we can solve the shortest vector problem in a lattice of dimension slightly larger than $1.4 \cdot m$. This is quite feasible with the LLL algorithm for m up to about 40, and with BKZ for m up to 100 or so.

6 Attack on Frodo Key Exchange

Our second attack targets the Frodo key exchange protocol. Like in the previous attack we tamper the generation of the "commitment" vectors $\mathbf{E}, \mathbf{E}'$. By the inherent symmetry of the protocol, we describe the attack when mounted on Alice's side. The adaptation to Bob's side is straightforward.

In this scheme, the commitment vector consists on a vector $\mathbf{E}$, which acts as an additive mask to the secret key vector $\mathbf{S}$ in the relation:

$$\mathbf{B} = \mathbf{S} \cdot \mathbf{A} + \mathbf{E}.$$

As a consequence, tampering the distribution of $\mathbf{E}$ leaks some information when sending the vector $\mathbf{B}$ to Bob.

More formally, $\mathbf{E} \in \mathcal{R}_q$ is a vector of integers drawn under a Gaussian distribution, its generation is then once again typically carried out coefficient, allowing us to use faults to cause an early abort of this process, and thus we should obtain a commitment vector $\mathbf{B}$ in which the vector $\mathbf{E}$ is sparse.

Formally, let consider a vector $\mathbf{B}$ obtained by forcing a termination of the loop for sampling $\mathbf{E}$ after the m-th iteration, with $m \ll n$ — that is, the resulting vector $\mathbf{E}$ has at most $m - 1$ non-zero coefficients —. Without loss of generality we can suppose the non-zero coefficients to be the first ones. As part of the commitment message, we

get the pair $(seed, \mathbf{B})$, and since the generation of $\mathbf{A}$ is both deterministic and public, the pair $(\mathbf{A}, \mathbf{B})$, satisfying the relation taken modulo q:

$$\left[\begin{array}{c|c} B_1 & B_2 \end{array}\right] = \mathbf{B} = \mathbf{S}\cdot\mathbf{A} + \mathbf{E}$$
$$= \left[\begin{array}{c|c} S_1 & S_2 \end{array}\right] \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right] + \left[\begin{array}{c|c} E_1 & \mathbf{0} \end{array}\right].$$

As such we have:

$$\left\{\begin{array}{rcl} S_1\cdot A_1 + S_2\cdot A_3 &=& B_1 - E_1 \\ S_1\cdot A_2 + S_2\cdot A_4 &=& B_2 \end{array}\right. \bmod q.$$

Let suppose at that point that A_4 is invertible[4] mod q.

$$\left\{\begin{array}{rcl} S_1\cdot A_1 + S_2\cdot A_3 &=& B_1 - E_1 \\ S_2 &=& (B_2 - S_1\cdot A_2)\cdot A_4^{-1} \end{array}\right. \bmod q, \tag{5}$$

yielding by replacement in the first equation:

$$S_1\cdot(A_1 - A_2\cdot A_4^{-1}\cdot A_3) + E_1 = B_1 - B_2\cdot A_4^{-1}\cdot A_3 \bmod q$$

This equation is then a m-dimensional instance of LWE, of the shape $S_1\cdot\tilde{A} + E_1 = \tilde{B}$.

Suppose now that $\tilde{A} = (A_1 - A_2\cdot A_4^{-1}\cdot A_3)$ is also invertible. Then performing the inversion trick described in the attack against NewHope yields:

$$\tilde{A}^{-1}\cdot\tilde{B} = S + \underbrace{A^{-1}\cdot E_1}_{\in \mathrm{Span}_{\mathbb{Z}}((\tilde{A}^{-1}(E_1)_i)_{0\le i\le m-1})} \bmod q$$

The attack is then mounted in the same fashion as before, considering the lattice generated by $\mathbf{w}_i = \tilde{A}^{-1}(E_1)_i \bmod q$ for $i = 0, \ldots, m-1$ and $q\mathbb{Z}^n$. Carrying the computations like in the NEW HOPE attack yields eventually the condition:

$$\ell + 1 \gtrsim \frac{m + 1 + \frac{\log\sqrt{\zeta}}{\log q}}{1 - \frac{\log\sqrt{\zeta}\sqrt{2\pi e}}{\log q}}, \tag{6}$$

for ζ the standard deviation of the distribution χ. Once S_1 is recovered, Equation 5, allows to recover directly S_2 by matricial manipulations and linear algebra.

7 IMPLEMENTATION OF THE FAULTS

Once again, due to the obvious similarities between the four instances of the Fiat–Shamir family that we choose to attack, we only give details of the attack on the BLISS scheme. We also give details for the GPV scheme but they are essentially the same as the one for BLISS since the underlying fault introduced is strictly identical.

In this section we investigate how an attacker may obtain helpful faulty signatures for the proposed attacks. We base our discussion on two available implementations of BLISS signature, namely the software implementation from Ducas and Lepoint [**?**] and the FPGA implementation by Pöppelmann *et al.* [**?**], and on Prest's software implementation of the GPV-based scheme of Ducas et al. [**?**]. Notice that the discussion on the hardware implementation is also valid for the implementation of [9] since both share some common components and architecture that we exploit (for instance BRAM storage).

We emphasize the fact that those three implementations were not supposed to have any resilience with respect to fault attacks and were only developed as proofs of concept to illustrate the efficiency properties of the schemes. The point here is to show that the fault attacks presented in this paper are relevant based on the analysis of freely available and published implementations to put forward the need of dedicated protections against faults attacks (when attackers have such abilities).

7.1 Classical fault models

Faults during a computation may be induced by different means as a laser beam shot, electromagnetic injection, under-powering, glitches, etc. These faults are mainly characterized by their

- range: impacting a single bit or many bits (e.g. register or memory word);
- effect: typically target chunk is set to a chosen value, random value or all-zero/all-one value;
- persistence: a fault may only modify the target for a short period or it may be definitive.

Obviously, some fault models are close from being purely theoretical: it is very unlikely to be able to set a 32-bit register to `0xbad00dad` during precisely 2 cycles. Nevertheless many recent works have been published showing that some faults models that seemed overdone are actually obtained during lab experiments with good reproducibility. That is true for single-bit faults both using EM [**?**] or laser injections [**?**]. There are also many reported skip instruction faults in the literature. One example of such faults is the work of Rivière et al. on ARMv7 intruction cache [**?**].

In the next subsections we discuss which fault models[5] may lead to faulty signatures that are relevant with respect to the attacks presented in this paper. We did not investigate clock glitches or under-powering which induce violation of the setup time and which actual side-effects are implementation and compilation-dependent (with large ranges of possible parameters to test). Nevertheless, they may not be overseen in the evaluation of a chip since they may also lead to the generation of exploitable faulty signatures.

7.2 Fault attacks on software implementations

Polynomial $\mathbf{y}_1$ can be generated using a loop over the n coefficients. This is indeed how the implementation in [**?**] is made: a loop is constructing polynomials $\mathbf{y}_1$ and $\mathbf{y}_2$ coefficient by coefficient using a Gaussian sampler (function `Sign::signMessage`). The condition to perform the attack is rather few restrictive since we only require $\mathbf{y}_1$ to have at most (roughly) a quarter of unknown coefficients. Such result can be obtain by going out of the loop after a few iterations. A random fault on the loop counter or skipping the jump operation will lead to such result.

Notice here that it is not trivial to decide whether a faulty signature will be helpful or not. Hopefully the required timing precision is much less important in this setting since

4. Seeing this matrix as a random uniform $m \times m$ matrix over $\mathbb{F}_q$, which is invertible with probability $\prod_{i=1}^{m}(1 - q^{-i})$.

5. We only focus on single fault attacks here.

the attack will succeed even with 50 unknown coefficients out of 512. This means that the time-window for the fault to occur is composed of decades of loop iterations. Moreover, we may use side-channel analysis to detect the loop iteration pattern to trigger the fault injection. Such pattern is likely to be detected after much less than 50 iterations and thus it seems that the synchronization here will be relatively easy.

Similarly, the short random polynomials $\mathbf{R}$ and $\mathbf{r}$ used in the GPV scheme are generated in a single loop [**?**] ranging from leading coefficient of $\mathbf{r}$ to the constant term in $\mathbf{R}$ which allows to fault both polynomials using a single fault. Again, a random fault on the counter or skipping a jump makes it work and the time-window large according to the results shown in Table 2.

To conclude, these attacks seems to be a real threat since synchronization (which is a major difficulty when performing fault attacks) is eased by the loose condition on the number of known coefficients in faulted polynomials.

7.3 Fault attacks on hardware implementations

Generation of polynomial $\mathbf{y}_1$ requires n random coefficients. It is very unlikely that all these coefficients are obtained at the same time (n is too large) thus $\mathbf{y}_1$ generation will be sequential. This is the case in the implementation we took as example where the super memory is linked to the sampler through a 14-bit port. We may fault a flag or a state register to fool the control logic (here the bliss processor) and keep part of the BRAM cells to their initial state. If this initial state is known then we know all the corresponding coefficients and hopefully the number of unknown ones will be small enough for the attack to work. Again, the large number of unknown coefficients handled by the attack helps the attacker by providing a large time window for the fault to occur. The feasibility of the attack will mostly depend on the precise flag/state implementation and the knowledge of memory cells previous/initial value.

There is a second way of performing the fault injection here. The value of $\mathbf{y}_1$ has to be stored somehow until the computation of $\mathbf{z}_1$ (close to the end of the signature generation). In the example implementation a BRAM is used. We may fault BRAM accesses to fix some coefficients to a known value. A possible fault would be to set the `rstram` or `rstreg` signal to one (Xilinx's nomenclature). Indeed, when set to one, this will set the output latches (*resp.* register) of the RAM block to some fixed value `SRVAL` defined by the designer. We may notice two points to understand why this kind of fault enables the proposed attack.

(i) The value $\mathbf{y}_1$ used to compute $\mathbf{u}$ will not be the faulted one but this has no impact on the attack.

(ii) If we do not know the default value for the output register, all coefficients are unknown but a big part of them are equal to the same unknown default value. In that case, the attack is still applicable by adding one generator to the constructed lattice: see Remark 3 in Section 3.

Again a large time window is given to the attacker due to sequential read induced by the size of $\mathbf{y}_1$.

The BRAM storage of $\mathbf{y}_1$ helps the attacker since a single bit-set fault may have effects on many coefficients. The only difficulty seems to be able to perform a single-bit fault — which is a real threat as stated in Section 7.1 — and the `rstram` signal localization[6].

7.4 Simulated faults

To confirm that a skipping-instruction fault actually leads to a secret recovery, fault attacks on NewHope have been simulated (both on s and e) using the tool made available by authors of [**?**]. They propose a python script that makes calls to the Aeroflex Gaisler's LEON3 CPU simulator (namely `tsim`) to simulate the replacement of one or more instruction by a `nop` (or to modify some data but it is out of our interest here). The LEON3 is a 32-bit processor (SPARC architecture) which source code is freely available. Thus, it can be used as a soft-core (i.e. a processor instantiated on an FPGA) or directly integrated as an IP for an ASIC. This target is the only one for which a fault simulator is available thus it is a perfect candidate to simulate the proposed attack.

For both polynomials e and s, the generation from the "small distribution" is performed by first getting some random bits from Chacha20 then processing each 32-bit word to obtain as many small coefficients (function `poly_get_noise`). We targeted this loop on the n coefficients computation (where n is a parameter of the scheme) to inject faults.

More precisely, enough random bytes are obtained and stored in a 32-bit word buffer. They are then processed to derive polynomial coefficients which are stored in a 16-bit word array. We first directly targeted the second loop (where coefficients are derived from random words) and managed to obtain two relevant faults in a few hours of simulation. This was mainly due to the fact that the granularity of the simulator is a range of address to fault. Due to the binary code structure, a jumping instruction was present present in this range leading to a huge amount of executed code between the first and the last potentially faulted instruction. This resulted in a large simulation time. Due to time limitations we did not try to modify the script from [**?**] to take this particular situation into account. Indeed, two faulty outputs corresponding to loop aborts in the first and second iterations were observed and allowed key-recovery and this was enough to confirm that the attack was an actual threat when instruction-skip faults can be injected on the device.

Nevertheless, and for the sake of completeness, the code has been modified to first derive coefficients in place and then to copy coefficients in the polynomial structure. This latter copy has been moved into a dedicated function to narrow the step-by-step simulation region. Using this code modification faults have been obtained for all abort iteration indexes. This faulty outputs have then been used to feed a `sage` attack script recovering the key (for small enough number of executed iterations indeed).

This confirms that skipping an instruction will lead to the expected behaviour according to the widely deployed simulator for LEON3 core. Particularly, this shows that there

6. Since $\mathbf{y}_1$ is not directly outputted checking if the attack actually worked is a bit more tricky. Again side-channel collision analysis may help here. We may also notice that if the faulty $\mathbf{y}_1$ is sparse (that is known coefficients have been set to zero) then the number of non-zero coefficients in the corresponding $\mathbf{z}_1$ should be significantly smaller then for a $\mathbf{z}_1$ corresponding to a dense $\mathbf{y}_1$.

is no side-effect of the fault nor of the targeted code that render the proposed attack more complicated or inefficient.

8 Conclusion and countermeasures

We have shown that unprotected implementations of lattice-based signature and key exchange schemes are vulnerable to fault attacks, in fault models that our analysis suggests are quite realistic: the faulty signatures required by our attacks can be obtained on actual implementations. As a result, countermeasures should be added in applications where such a physical attacker is relevant to the threat model.

Simple countermeasures exist to thwart the single fault attacks proposed. There are simple, non-cryptographic countermeasures that consist in validating that the full loop have been correctly performed. This can be achieved for instance by adding a second loop counter and doing a consistency check after exiting the loop. Such a countermeasure is very cheap and we therefore recommend introducing it in all deployed implementations.

Nevertheless, it will only detect early-abort faults while an attacker may succeed in getting the same kind of faulty signature using another technique. For instance, we mentioned the possibility of faulting BRAM blocks so that they output a fixed value. For software implementations, the compiler may decide to put the coefficient in some RAM location which address could be faulted to point to another part of the memory leading in many coefficients having the same value. A single fault may also alter instruction cache leading to a `nop` operation instead of a load from memory and thus not updating the coefficient. We propose now other countermeasures that may deal with this issue for both types of signature schemes we considered.

We have described our attack on the Fiat–Shamir schemes in a setting where the attacker can obtain a commitment polynomial $\mathbf{y}$ of low degree, and it works more generally with a sparse $\mathbf{y}$, provided that the attackers *knows* where the non zero coefficients are located. If the locations are unknown, however, the attack does not work, so one possible countermeasure is to randomize the order of the loop generating $\mathbf{y}$. One should be careful that this may not protect against faults introduced after the very first few iterations, however: in the case of BLISS, for example, we have seen that we could easily attack polynomials $\mathbf{y}$ in which the non zero coefficients are located in the 20% lower degree coefficients, say; then, if a fault attacker can collect a few hundred faulty signatures with $\mathbf{y}$ of very low Hamming weight (say 3 or 4) at random positions, they have a good chance of finding one fault with all non zero coefficients in the lower 20%, and hence be able to attack.

Another possible approach for the Fiat–Shamir schemes is to check that the degree of the generated $\mathbf{y}$ is not too low. One cannot demand that all its coefficients are non zero, as this would skew the distribution and invalidate the security argument, but verifying that the top $\varepsilon \cdot n$ coefficients of $\mathbf{y}$ are not all zero for some small constant $\varepsilon > 0$, say $\varepsilon = 1/16$, would be a practical countermeasure that does not affect the security proof. Indeed, in the case of BLISS for example, the probability that all of these coefficients vanish is roughly $(1/\sigma\sqrt{2\pi})^{\varepsilon n}$, which is exponentially small. Thus, the resulting distribution of $\mathbf{y}$ after this check is statistically indistinguishable from the original distribution, and security is therefore preserved. Moreover, the lattice dimension required to mount our fault attack is then greater than $(1-\varepsilon)n$, so it will not work. An additional advantage of that countermeasure is that it also adapts easily to thwart faults that cause all the top coefficients of $\mathbf{y}$ to be equal to some constant non-zero value.

Regarding the hash-and-sign signature of Ducas et al., one possible countermeasure is to simply check the validity of generated signatures. This will usually work due to the fact that a faulty signature generated from an early loop abort from the GaussianSampler algorithm is of significantly larger norm than a valid signature: a rough estimate of the norm after $m \leq n$ iterations is $\|\mathbf{F}\|_2\sqrt{mq/12}$ (as $q/12$ is the variance of a uniform random variable in $\{-(q-1)/2, \ldots, (q-1)/2\}$), which is too large for correct verification even for very small values of m. An added benefit of that countermeasure is that even the correct signature generation algorithm has a very small but non zero probability of generating an invalid signature, so this countermeasure doubles up as a safeguard against those rare accidental failures.

References

[1] Commercial national security algorithm suite and quantum computing FAQ. Technical report, National Security Agency, Jan. 2016. Available at https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm.

[2] S. Akleylek, N. Bindel, J. Buchmann, J. Krämer, and G. A. Marson. *Progress in Cryptology – AFRICACRYPT 2016: 8th International Conference on Cryptology in Africa, Fes, Morocco, April 13-15, 2016, Proceedings*, chapter An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation. 2016.

[3] S. Akleylek, N. Bindel, J. A. Buchmann, J. Krämer, and G. A. Marson. An efficient lattice-based signature scheme with provably secure instantiation. In D. Pointcheval, A. Nitaj, and T. Rachidi, editors, *AFRICACRYPT*, volume 9646 of *LNCS*, pages 44–60. Springer, 2016.

[4] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343, 2016.

[5] J. Benaloh, editor. *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *LNCS*. Springer, 2014.

[6] I. Biehl, B. Meyer, and V. Müller. Differential fault attacks on elliptic curve cryptosystems. In M. Bellare, editor, *CRYPTO*, volume 1880 of *LNCS*, pages 131–146. Springer, 2000.

[7] N. Bindel, J. A. Buchmann, and J. Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. In *FDTC 2016*, pages 63–77. IEEE Computer Society, 2016.

[8] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001.

[9] J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018, 2016.

[10] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570, 2015.

[11] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 323–345, 2016.

[12] C. Champeix, N. Borrel, J. Dutertre, B. Robisson, M. Lisart, and A. Sarafianos. SEU sensitivity and modeling using pico-second pulsed laser stimulation of a D flip-flop in 40 nm CMOS technology. In *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFTS 2015, Amherst, MA, USA, October 12-14, 2015*, pages 177–182. IEEE Computer Society, 2015.

[13] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on post-quantum cryptography. Technical report, National Institute of Standards and Technology, Feb. 2016. Available at http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf.

[14] Ö. Dagdelen, R. E. Bansarkhani, F. Göpfert, T. Güneysu, T. Oder, T. Pöppelmann, A. H. Sánchez, and P. Schwabe. High-speed signatures from standard lattices. In D. F. Aranha and A. Menezes, editors, *LATINCRYPT*, volume 8895 of *LNCS*, pages 84–103. Springer, 2014.

[15] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven. What is the Computational Value of Finite Range Tunneling? *ArXiv e-prints*, Dec. 2015.

[16] L. Ducas. Accelerating BLISS: the geometry of ternary polynomials. Cryptology ePrint Archive, Report 2014/874, 2014. http://eprint.iacr.org/.

[17] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013.

[18] L. Ducas and T. Lepoint. A proof-of-concept implementation of BLISS. Available at http://bliss.di.ens.fr.

[19] L. Ducas, V. Lyubashevsky, and T. Prest. Efficient identity-based encryption over NTRU lattices. In P. Sarkar and T. Iwata, editors, *ASIACRYPT*, volume 8874 of *LNCS*, pages 22–41. Springer, 2014.

[20] L. Ducas and P. Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In X. Wang and K. Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 433–450. Springer, 2012.

[21] T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi. Loop abort faults on lattice-based fiat-shamir & hash'n sign signatures. In *Selected Areas in Cryptology (SAC)*, page 449, 2016.

[22] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[23] F. Fontein and P. Wocjan. On the probability of generating a lattice. *Journal of Symbolic Computation*, 64:3–15, 2014.

[24] C. Gentry, J. Jonsson, J. Stern, and M. Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 1–20. Springer, 2001.

[25] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In C. Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[26] C. Gentry and M. Szydlo. Cryptanalysis of the revised NTRU signature scheme. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 299–320. Springer, 2002.

[27] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In B. S. K. Jr., editor, *CRYPTO*, volume 1294 of *LNCS*, pages 112–131. Springer, 1997.

[28] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *CHES*, volume 7428 of *LNCS*, pages 530–547. Springer, 2012.

[29] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSign: Digital signatures using the NTRU lattice. In M. Joye, editor, *CT-RSA*, volume 2612 of *LNCS*, pages 122–140. Springer, 2003.

[30] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, and W. Whyte. Practical signatures from the partial fourier recovery problem. In I. Boureanu, P. Owesarski, and S. Vaudenay, editors, *ACNS*, volume 8479 of *LNCS*, pages 476–493. Springer, 2014.

[31] J. Howe, T. Pöppelmann, M. O'Neill, E. O'Sullivan, and T. Güneysu. Practical lattice-based digital signature schemes. *ACM Trans. Embedded Comput. Syst.*, 14(3):41, 2015.

[32] J. Howe, T. Pöppelmann, M. O'Neill, E. O'Sullivan, T. Güneysu, and V. Lyubashevsky. Practical lattice-based digital signature schemes. Slides of the presentation at the NIST Workshop of Cybersecurity in a Post-Quantum World, 2015. Available at http://csrc.nist.gov/groups/ST/post-quantum-2015/presentations/session9-oneill-maire.pdf.

[33] A. A. Kamal and A. M. Youssef. Fault analysis of the NTRUSign digital signature scheme. *Cryptography and Communications*, 4(2):131–144, 2012.

[34] V. Lyubashevsky. Fiat–Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009.

[35] V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.

[36] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP*, volume 4052 of *LNCS*, pages 144–155. Springer, 2006.

[37] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In R. Canetti, editor, *TCC*, volume 4948 of *LNCS*, pages 37–54. Springer, 2008.

[38] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.

[39] G. Maze, J. Rosenthal, and U. Wagner. Natural density of rectangular unimodular integer matrices. *Linear Algebra and its Applications*, 434(5):1319–1324, 2011.

[40] C. A. Melchor, X. Boyen, J. Deneuville, and P. Gaborit. Sealing the leak on classical NTRU signatures. In M. Mosca, editor, *PQCrypto*, volume 8772 of *LNCS*, pages 1–21. Springer, 2014.

[41] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012.

[42] D. Naccache, P. Q. Nguyen, M. Tunstall, and C. Whelan. Experimenting with faults, lattices and the DSA. In S. Vaudenay, editor, *PKC*, volume 3386 of *LNCS*, pages 16–28. Springer, 2005.

[43] P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.

[44] S. Ordas, L. Guillaume-Sage, K. Tobich, J. Dutertre, and P. Maurine. Evidence of a larger EM-induced fault model. In M. Joye and A. Moradi, editors, *CARDIS*, volume 8968 of *LNCS*, pages 245–259. Springer, 2014.

[45] D. Page and F. Vercauteren. A fault attack on pairing-based cryptography. *IEEE Trans. Computers*, 55(9):1075–1080, 2006.

[46] C. Patrick, B. Yuce, N. F. Ghalaty, and P. Schaumont. Lightweight fault attack resistance in software using intra-instruction redundancy. In *Selected Areas in Cryptography, 19th International Conference, SAC 2016, Revised Selected Papers*, 2016.

[47] C. Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 197–219, 2014.

[48] C. Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. http://eprint.iacr.org/.

[49] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *LNCS*, pages 145–166. Springer, 2006.

[50] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[51] T. Pöppelmann, L. Ducas, and T. Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In L. Batina and M. Robshaw, editors, *CHES*, volume 8731 of *LNCS*, pages 353–370. Springer, 2014.

[52] T. Prest. Implementation of the GPV-based scheme of Ducas et al. Available at https://github.com/tprest/Lattice-IBE.

[53] L. Rivière, Z. Najm, P. Rauzy, J. Danger, J. Bringer, and L. Sauvage. High precision fault injections on the instruction cache of armv7-m architectures. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 62–67. IEEE Computer Society, 2015.

[54] W. Stein et al. *Sage Mathematics Software (Version 7.0)*, 2016. `http://www.sagemath.org`.

[55] M. Taha and T. Eisenbarth. Implementation attacks on post-quantum cryptographic schemes. In E. A. Aleisa, editor, *ICACC*. IEEE Social Implications of Technology Society, 2015.

APPENDIX

Description of the other Fiat–Shamir schemes. In [12], Lyubashevsky describes a signature scheme proved secure in the random-oracle model which is an alternative to hash-and-sign methodology of Gentry et al. in [7]. Gentry, Peikert and Vaikuntanathan were the first to propose a signature scheme whose security is based on the hardness of worst-case lattice problems, while Lyubashevsky and Micciancio present a one-time signature scheme based on the hardness of worst-case ideal lattice problems [14]. Lyubashevsky propose a Fiat–Shamir framework [5] using rejection sampling technique in [11]. Both signature schemes are inefficient in practice: [7] requires megabytes long signature and [11] needs 60,000 bits for reasonable parameters.

Many previous lattice-based signature schemes have been broken since information about the secret key leaks in every signature [4], [6], [8], [16]. Consequently, the basic idea of the Lyubashevsky and BLISS signature schemes is to use the rejection sampling so that the distribution output is independent of the secret key. This signature scheme is proved secure on the hardness of the ring version of $\ell_2 - \mathrm{SIS}_{q,n,m,\beta}$.

In the Figure 5, we describe the version of Güneysu et al. in [9] which is a particular instantiation of the ring version of Lyubashevsky signature as presented in Section 7 in [12]. We denote by $\mathcal{R}_{q,k}$ the subset of $\mathcal{R}_q$ that consists of all polynomials with integer coefficients in the interval $[-k; k]$. The hardness assumption of [9] is that $(\mathbf{a}, \mathbf{t}) \in \mathcal{R}_q \times \mathcal{R}_q$ where $\mathbf{a}$ is chosen uniformly in $\mathcal{R}_q$ and $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$ with $\mathbf{s}_1$ and $\mathbf{s}_2$ uniformly chosen in $\mathcal{R}_{q,k}$ is indistinguishable from $(\mathbf{a}, \mathbf{t})$ uniformly chosen in $\mathcal{R}_q \times \mathcal{R}_q$. When $\sqrt{q} < k$, the solution $(\mathbf{s}_1, \mathbf{s}_2)$ is not unique and finding one of them is as hard as worst-case lattice problems in ideal lattices [13], [17]. In [15], it was shown that if $\mathbf{s}_i$ are chosen according a Gaussian distribution instead of a uniform one, then recovering the $\mathbf{s}_i$ given $(\mathbf{a}, \mathbf{t})$ is as hard as solving worst-case lattice problems using a quantum computer. In the following our attacks do not take into account the way the secret key is generated and work in all cases.

Description of the PASSSign signature scheme. PASSSign is a signature scheme introduced by Hoffstein et al. in [10]. This scheme is a variant of the PASS and PASS-2 scheme from the same authors, adding the *rejection sampling* technique of Lyubashevsky from 2009. Its hardness is based on the problem of recovering a ring element with small norm from an incomplete description of its Chinese remainder representation.

We follow in its description the original presentation and notation of [10]. Computations are made in the ring $\mathbb{Z}_q[x]/(x^N - 1)$. On that ring, we define $\mathcal{B}_q^\infty$ the subset of polynomials whose coefficients lie in $[-k, k]$. Given g a primitive $N-th$ root of unity in $\mathbb{Z}_q$, Ω a subset of $\{g^i | 1 \leq i \leq N-1\}$, we define the mapping $\mathcal{F}_\Omega : \mathbb{Z}_q[x]/(x^N - 1) \to \mathbb{Z}_q^{|\Omega|}$ consisting in the multi-evaluation of a polynomial on the elements of Ω. The image of a polynomial $\mathbf{f}$ by $\mathcal{F}_\Omega$ will be simply denoted by $\mathbf{f}|_\Omega$. The function FormatC maps the set of bit strings output by the Hash function H into a set of sparse polynomials. Once again, since its details are not mandatory when mounting the attack, we let the interested reader to refer to the original paper for an in-depth description. It full description is given in Figure 6.

Description of the TESLA signature scheme. The TESLA scheme is a variation of the BG scheme presented in [2], initially modified by Dagdelen et al. in [3] at LATINCRYPT 14, to get rid of the forking lemma in the security analysis.

On the contrary of the two previous presented schemes, the TESLA signatures works directly on vectors — and no more on the additional algebraic structure provided by the use of polynomials —. The matrix $\mathbf{A}$ used in the scheme is publicly known and can be seen as a global constant shared by arbitrary many users. The CheckE function is fully described in the original paper from Dagdelen et al. [3] and ensures mandatory properties to preserves that the signature remains short. Once again, we do not fully describe it here since its details are irrelevant for our attacks. We conclude this presentation by noting that the security proof uses the hardness of the LWE problem. Its specificity is to avoid the use of the *Forking Lemma* proposed by Pointcheval and Stern in [18].

More precisely, we are interested in its variant Ring-TESLA, presented in [1], which offers provably secure instantiation. Its full description is given in Figure 7.

Extension of the first attack to other members of the Fiat–Shamir family. In this section we precise a bit more why the attack described on BLISS apply almost straightly to the other members of the Fiat–Shamir family described above: GLP, PASSSign and Ring-TESLA.

On Lyubashevsky Scheme. The difference with BLISS lies in the rejection sampling used and in the generation of the $\mathbf{y}_1, \mathbf{y}_2$ commitment coefficients. Thus there is no difference in the way of mounting the attack: here again, only a single fault is only needed to early-abort the generation loop of the element $\mathbf{y}_1$ and force its degree to be low.

On Ring-TESLA. In Ring-TESLA, the situation is slightly different since only one element $\mathbf{y}$ is generated, whose coefficients are drawn uniformly in $[-B; B]$. Yet, the same early-abort in its generation can be performed to force its degree to be low. Let us suppose that its degree is $m-1$; that is, the generation loop has been stopped after m iteration. Then, once again with high probability — namely $(1 - \frac{1}{q})^n$ — the element $\mathbf{c}$ outputted by the signature is invertible and the following equality holds:

$$\mathbf{c}^{-1}\mathbf{z} - \mathbf{s} \equiv \mathbf{c}^{-1}\mathbf{y} \equiv \sum_{i=0}^{m-1} y_i \mathbf{c}^{-1}\mathbf{x}^i \pmod{q} \qquad (7)$$

where $\mathbf{y} = \sum_{i=0}^{m-1} y_i \mathbf{x}^i$. We can now perform the same trick as in Section 3.

The analysis yields this time that:

$$\ell + 1 \gtrsim \frac{m + 1 + \frac{\log \sigma}{\log q}}{1 - \frac{\log(\sigma\sqrt{2\pi e})}{\log q}}.$$

Then, as in Section 3, to complete the attack, it suffices to apply the above to a family of subsets I of $\{0, \ldots, n-1\}$ covering the whole set of indices, which reveals the entire vector $\mathbf{s}$. Recovering the remaining components of the secret

function SIGN($\mu, \mathbf{a}, \mathbf{s}_1, \mathbf{s}_2$)
  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow \mathcal{R}_{q,k}$
  $\mathbf{c} = H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$
  $\mathbf{z}_1 = \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$, $\mathbf{z}_2 = \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
  If $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{q,k-32}$, goto 1
  return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$
end function

function VERIFY($\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t}$)
  Accept iff $\mathbf{z}_1$ and $\mathbf{z}_2 \in \mathcal{R}_{q,k-32}$ and $\mathbf{c} = H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mu)$
end function

Fig. 5. Lyubashevsky or [9] signature scheme based on Ring $\ell_2 - \mathrm{SIS}_{q,n,m,\beta}$. The signing key are $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{R}_{q,1}$ where each coefficient of every $\mathbf{s}_i$ is chosen uniformly and independently from $\{-1, 0, 1\}$ The verification key is $(\mathbf{a}, \mathbf{t})$ where $\mathbf{a} \leftarrow \mathcal{R}_q$ and $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$ The random oracle is modeled by $H : \{0,1\}^* \rightarrow \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^n, \|\mathbf{v}\|_1 \leq \kappa\}$ with $\kappa = 32$ Two sets of parameters for (n, q, k) are given for estimated security of 100 and 256 bits: Set I $(512, 8383489, 2^{14})$ for a 8,950-bit signature, 1620-bit secret key and 11800-bit public key and Set II $(1024, 16760833, 2^{15})$ for a 18800-bit signature, 3250-bit secret key and 25000-bit public key.

function SIGN(μ, f)
  $\mathbf{y} \leftarrow \mathcal{B}_k^\infty$
  $\mathbf{h} = H(\mathbf{y}|_\Omega, \mu)$
  $\mathbf{c} = \mathrm{FormatC}(\mathbf{h})$
  $\mathbf{z} = \mathbf{y} + \mathbf{f} \cdot \mathbf{c}$
  If $\mathbf{z} \notin \mathcal{B}_{k-b}^\infty$, goto 1
  return $(\mathbf{c}, \mathbf{z}, \mu)$
end function

function VERIFY($\mu, \mathbf{c}, \mathbf{z}, \mathbf{c}, \mathbf{f}|_\Omega$)
  Accept iff $\mathbf{z}_2 \in \mathcal{B}_{k-b}^\infty$ and $\mathbf{c} = \mathrm{FormatC}(H(\mathbf{z}|_\Omega - \mathbf{f} \cdot \mathbf{c}|_\Omega, \mu))$
end function

Fig. 6. Description of the PASSSign signature. The public parameters are: g a primitive $N - th$ root of unity in $\mathbb{Z}_q$, Ω a subset of $\{g^i | 1 \leq i \leq N-1\}$, t its cardinal, k the infinity norm of noise polynomials, and b the 1-norm of challenge polynomials. The signing key is the secret $\mathbf{f} \in \mathbb{Z}_q[X]/(X^n - 1)$ of small norm, that is of L_∞ norm equal to 1. Authors recommend the simple strategy of choosing each coefficient independently and uniformly from $1, 0, 1$. The vector $\mathbf{t}$ is defined as $\mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$ The random oracle is modeled by $H : \mathbb{Z}_q^t \times \{0,1\}^* \rightarrow \{0,1\}^l$. Two sets of parameters for (n, q, k) are given for estimated security of 100 and 128 bits: Set I $(769, 1047379, 2^{15} - 1)$ for a 12624-bit signature, 1600-bit secret key and 7720-bit public key and Set II $(1152, 968521, 2^{15} - 1)$ for a 18800-bit signature, 2000-bit secret key and 12000-bit public key.

function SIGN($\mu, \mathbf{a}_1, \mathbf{a}_2, sk = (\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2)$)
  $\mathbf{y} \leftarrow^\$ [-B; B]^n$
  $\mathbf{v}_1 = \mathbf{a}_1\mathbf{y} \bmod q$
  $\mathbf{v}_2 = \mathbf{a}_2\mathbf{y} \bmod q$
  $\mathbf{c} \leftarrow H(\lfloor \mathbf{v}_1 \rceil_d, \lfloor \mathbf{v}_2 \rceil_d, \mu)$
  $\mathbf{c} \leftarrow F(\mathbf{c})$
  $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{s}\mathbf{c}$
  $\mathbf{w}_1 \leftarrow \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \bmod q$
  $\mathbf{w}_2 \leftarrow \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \bmod q$
  If **If** $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - L$ **or** $\|\mathbf{z}\|_\infty > B - U$ **then** Restart.
  return $(\mathbf{z}, \mathbf{c})$
end function

function KEYGEN()
  $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow D_\sigma^n$
  If not CheckE($\mathbf{e}_i$) **then Restart**
  return $(pk = (\mathbf{t}_1, \mathbf{t}_2), sk = (\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2))$ where $\mathbf{t}_i = \mathbf{a}_i\mathbf{s} + \mathbf{e}_i \bmod q$
end function

function VERIFY($\mu, \mathbf{a}_1, \mathbf{a}_2, (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c}), pk = (\mathbf{t}_1, \mathbf{t}_2)$)
  $\mathbf{c} \leftarrow F(\mathbf{c})$
  $\mathbf{w'}_1 \leftarrow \mathbf{a}_1\mathbf{z} - \mathbf{t}_1\mathbf{c} \bmod q$
  $\mathbf{w'}_2 \leftarrow \mathbf{a}_2\mathbf{z} - \mathbf{t}_2\mathbf{c} \bmod q$
  $\mathbf{c'} \leftarrow H(\lfloor \mathbf{w'}_1 \rceil_d, \lfloor \mathbf{w'}_2 \rceil_d, \mu)$ **Accept iff** $\mathbf{c'} = \mathbf{c}$ **and** $\|\mathbf{z}\|_\infty \leq B - U$
end function

Fig. 7. Description of the Ring-TESLA Signature Scheme. The public parameters are $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{Z}_q^n, n \in \mathbb{Z}$. The scheme uses an encoding function: $F : \{0,1\}^\kappa \rightarrow \mathcal{B}_{n,\omega}$, the space of vectors length n and weight ω. The random oracle is modeled by $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$. A set of parameters are proposed with security level at least 128 bits. The interesting parameters for us are: $\kappa = 128$, $n = 512$, $q = 39960577$, $\sigma = 52$, $U = 3173$, $d = 23$, $\omega = 19$, $L = 2766$ and $B = 2^{22} - 1$. The resulting signature size around 1,488B, secret key size around 1,920B and public key size of 3,328B. From the point of view of the attack mounted, we are not interested in the CheckE function and we will not detail it here.

key is now a straightforward modular inversion using the public parameters $\mathbf{a}_1, \mathbf{a}_2$.

On PASSSign. Like in the Ring-TESLA scheme only one $\mathbf{y}$ is generated when signing and the same attack can be mounted against the generation of this last vector. With regards to the methodology used, the only difference which appears when following the analysis lies in the norm of the secret key $\mathbf{f}$: in PASSSign, the secret key is a polynomial of coefficients independently drawn from $\{-1, 0, 1\}$. As such, if using the same notations as before, we get a vector $\varphi_I(\mathbf{s})$ of norm roughly equals to $\sqrt{\frac{2\ell}{3} + B^2}$. We then choose $B = 1$, which leads to the following inequality on ℓ:

$$\ell + 1 \gtrsim \frac{m+1}{1 - \frac{\log 2 \cdot \sqrt{(\frac{\pi e}{3})}}{\log q}}.$$

Then, to complete the attack, it suffices to apply the same method to a family of subsets I of $\{0, \ldots, n-1\}$ covering the whole set of indices, which reveals the entire secret $\mathbf{f}$.