Loop-Abort Faults on Lattice-Based Signature Schemes 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 interested both in Fiat–Shamir type constructions (particularly BLISS, but also GLP, PASSSign, 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 NewHope, Frodo, and Kyber. These schemes form a representative sample of modern, practical lattice-based signatures and key exchange protocols, and achieve a high level of efficiency 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), show that those attacks can be mounted in practice based on concrete experiments in hardware, and discuss possible countermeasures against them.

Index Terms—Fault attacks, digital signatures, postquantum cryptography, lattices

1 Introduction

1.1 Lattice-Based Cryptography

Recent progress in quantum computation [15], the NSA advisory memorandum recommending the transition away from Suite B and to postquantum cryptography [45], as well as the announcement of the NIST standardization process for postquantum cryptography [13] 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 flavors of postquantum cryptography, lattice-based schemes occupy a position of particular interest, as they rely on well-studied problems and come with uniquely strong security guarantees, such as worst-case to average-case reductions [52]. A number of works have also focused on improving the performance of lattice-based cryptography, and actual implementation results suggest that properly optimized schemes may be competitive with, or even outperform, classical factoring and discrete logarithm-based constructions.[1]

1. Usually, those efficient instantiations do not have parameters achieving high security under known reductions to worst-case problems, but the existence of those reductions is usually seen as a good sign that the security assumptions themselves are sound.

- *T. Espitau is with the Sorbonne Universités, UPMC, LIP6, Paris 75005, France. E-mail: thomas.espitau@lip6.fr.*
- *P.-A. Fouque is with the Institut Universitaire de France & IRISA, University of Rennes, Rennes 35000, France. E-mail: pierre-alain.fouque@univ-rennes1.fr.*
- *B. Gérard is with the DGA.MI, Bruz 35131, France, and is with the IRISA, Rennes 35000, France. E-mail: benoit.gerard@irisa.fr.*
- *M. Tibouchi is with the NTT Secure Platform Laboratories, Musashino-shi, Tokyo 180-8585, Japan. E-mail: tibouchi.mehdi@lab.ntt.co.jp.*

Manuscript received 15 May 2017; revised 26 Feb. 2018; accepted 18 Mar. 2018. Date of publication 3 May 2018; date of current version 16 Oct. 2018. (Corresponding author: Thomas Espitau.)
Recommended for acceptance by Ç. K. Koç, Z. Liu, and P. Longa.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2018.2833119

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). 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! [58]). 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 that a few bits of leakage or a few fault injections can reveal an entire secret key (the well-known attacks from [6], [8] are typical examples). 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 and key exchange protocols, as those primitives are probably the most likely to be deployed in a setting where fault attacks are relevant, and 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 have been 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 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 [37].

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 [39], which uses rejection sampling to ensure that

the underlying identification scheme achieves honest-verifier zero-knowledge. Among lattice-based schemes, the exemplar in that family is Lyubashevsky's scheme from EUROCRYPT 2012 [40]. 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 [34], and then to a larger extent by Ducas et al. in their scheme BLISS [18], 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. [54]. Other schemes in that family include Hoffstein et al.'s PASSSign [36], which incorporates ideas from NTRU, and Akleylek et al.'s Ring-TESLA [1], 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 [32] and NTRUSign [35], but despite several attempts to patch them[2] they turned out to be insecure. A principled, provable approach to designing lattice-based hash-and-sign signatures was first described by Gentry et al. in [30], based on discrete Gaussian sampling over lattices. The resulting scheme, GPV, is rather inefficient, even when using faster techniques for lattice Gaussian sampling [43]. However, Ducas, Lyubashevsky and Prest [21] 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 [3], [9], [10], [51] 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 percent 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 [51] 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 [3], and constructions using standard lattices, like Frodo [9]. The recent scheme Kyber [11], relying on so-called module lattices, is in some sense a middle ground between those two approaches.

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 mentioned above.

As noted previously, early lattice-based signature schemes with heuristic security have been broken using standard attacks [29], [31], [46] but recent constructions including [18], [21], [30], [39], [40] 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 [44] and pairing computations [49], 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 [18], [54], and show how it extends to GLP [34], PASSSign [36] and Ring-TESLA [1]. 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.

Our second attack targets the GPV-based hash-and-sign signature scheme of Ducas et al. [21]. 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 three protocols that represent the state of the art for lattice-based key exchange, namely NewHope [3], Frodo [9] and Kyber [11]. 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 mathematical simulations in Sage [57]. We also take a close look at the

2. There is a provably secure scheme due to Aguilar et al. [42] 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.

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 (taking NewHope as an example) in two ways against two types of platforms:

- first, we successfully carry out a simulation of our fault attack 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;
- second, we concretely carry out those faults with clock glitches against an 8-bit AVR XMEGA microcontroller, using the power analysis and fault attack testing board ChipWhisperer-Lite [47].

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 [58], is the fault analysis work of Kamal and Youssef on NTRUSign [38]. It is, however, of limited interest since NTRUSign is known to be broken [22], [46]; it also suffers from a very low probability of success.

In 2016, concurrently with our work, Bindel, Buchmann and Krämer [7] 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 [7, Section 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. [33]. 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. Further side-channel attacks on BLISS, including the works of Pessl et al. [53] and Espitau et al. [24], have similar limitations.

A preliminary version of this paper has been published at SAC 2016 in [23]. We extend this version by attacking a larger number of signature schemes (GLP, PASSSign, Ring-TESLA), and generalizing our original attacks to the key exchange settings (against NewHope, Frodo and Kyber). 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, as is the practical attack on the AVR microcontroller.

1.6 Applicability to NIST-Submitted Variants of the Target Schemes

Many of the schemes we consider in this paper have been submitted to the NIST postquantum standardization effort [13] in a slightly modified form.

In particular, the NIST-submitted versions of NewHope, Frodo and Kyber are presented as key encapsulation mechanisms (KEMs) instead of key-exchange protocols in the usual sense. Key exchange then corresponds to key generation for one party and key encapsulation for the other. That modification of the overall structure of the schemes does not significantly affect the actual operations carried out as part of those schemes, and in particular our attacks apply to the KEM versions with essentially no change.

Regarding signatures, none of the schemes discussed in this paper (BLISS, Lyubashevsky's scheme, Ring-TESLA, PASSSign and the DLP scheme of Ducas, Lyubashevsky and Prest) have been submitted to the NIST effort. However, NIST candidate qTESLA [2] is a slight variant of Ring-TESLA, and our attack extends in a straightforward way. One should simply pay attention to the fact that the randomness $\mathbf{y}$ is generated as a deterministic function of the message; since that deterministic function involves sampling each coefficient of $\mathbf{y}$ one by one, loop-abort faults apply similarly. On a related note, NIST candidate Dilithium [20] can be seen as a modified version of BLISS, using module lattices instead of ideal lattices (and, again, a deterministic generation for $\mathbf{y}$). As a result, just as what happens for Frodo and Kyber, our attack may extend or not depending on the direction in which the matrix of coefficients is filled as part of the generation of $\mathbf{y}$.

PASSSign does not appear to have a counterpart submitted to NIST, and Lyubashevsky's scheme is conceptually related to most submissions, but the low-level behavior is usual different. On the hash-and-sign side, the closest NIST candidate to DLP is Falcon [27]. However, although the main idea of constructing a hash-and-sign signature using Gaussian sampling on an NTRU lattice is the same, Falcon uses a tree-shaped recursive algorithm for lattice sampling instead of a linear loop. Due to this modified structure, our attack does not apply directly to that scheme.

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

2.1 Notation

For any integer q, we identify the elements of the ring $\mathbb{Z}_q$ with the integers in $[-q/2, q/2)$. Vectors are considered as column vectors and are written in bold lower case letters; matrices are denoted by upper case letters. We use both the ℓ_2 Euclidean norm of vectors, $\|\mathbf{v}\|_2 = (\sum_i v_i^2)^{1/2}$, and their ℓ_∞-norm, $\|\mathbf{v}\|_\infty = \max_i |v_i|$. In ring-based schemes, ring elements (which can be seen as polynomials) are identified with their vectors of coefficients, and hence also denoted by bold lower case letters.

The Gaussian distribution with standard deviation $\sigma \in \mathbb{R}$ and center $c \in \mathbb{R}$ at $x \in \mathbb{R}$, is defined by $\rho_{c,\sigma}(x) = \exp(\frac{-(x-c)^2}{2\sigma^2})$ and more generally in higher dimension 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

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($\mu, pk = \mathbf{a}_1, sk = \mathbf{S}$)
  $\underline{\mathbf{y}_1 \leftarrow D^n_{\mathbb{Z},\sigma}}$, $\mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$
  $\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$
  $\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$
  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 $1/(M \exp(-\|\mathbf{Sc}\|/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle / \sigma^2)$
  $\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
  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 [18] 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 underlined instruction (Step 2 in SIGN) is where we introduce our faults.

$D_\sigma(x) = \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ (or $D_{\mathbb{Z},\sigma}$) and more generally over $\mathbb{Z}^m$ by $D^m_\sigma(\mathbf{x}) = \rho_\sigma(\mathbf{x})/\rho_\sigma(\mathbb{Z}^m)$, 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 [18] is possibly the most efficient lattice-based signature scheme so far. It has been implemented in both software [19] and hardware [54], 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 [40], sharing the same "Fiat–Shamir with aborts" structure [39]. 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 Fig. 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 [21]

The second signature scheme we consider is the one proposed by Ducas, Lyubashevsky and Prest at ASIACRYPT 2014 [21]. It is an optimization using NTRU lattices of the GPV hash-and-sign signature scheme of Gentry et al. [30], and has been implemented in software by Prest [55]. As in GPV, the signing key is a "good" basis of a certain lattice L (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 L, and uses the good, secret basis to sample $\mathbf{v} \in L$ according to a discrete Gaussian distribution of small variance supported on L 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 leaks no information about the secret basis is what makes it possible to prove security). The actual scheme of Ducas–Lyubashevsky–Prest, described in Fig. 2, uses a lattice of the same form as NTRU: $L = \{(\mathbf{y}, \mathbf{z}) \in \mathcal{R}^2 \mid \mathbf{y} + \mathbf{z} \cdot \mathbf{h} = 0\}$, 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.

2.4 Description of NewHope

The NewHope [3] key exchange protocol is one of the highest-profile key exchange protocols from lattices, and has been awarded the 2016 Internet Defense Prize at the USENIX Security conference. It is based on the so-called Peikert tweak and can be seen as a variant of the scheme of Ding et al. [16]. NewHope improves upon earlier schemes in various ways. On the one hand the authors proposed a refined analysis of the failure probability of the key 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 matching pseudorandom ring element. On Alice's side, this element

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 $D_{L,\sigma,\mathbf{c}}$
end function

function SIGN($\mu, sk = \mathbf{B}$)
  $\mathbf{c} \leftarrow H(\mu) \in \mathbb{Z}_q^n$
  $(\mathbf{y}, \mathbf{z}) \leftarrow (\mathbf{c}, \mathbf{0}) -$ 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 skew-circulant of dimension n. For 128 bits of security, the authors of [21] recommend the parameters $n = 256$ and $q \approx 2^{10}$. The constant 1.17 is an approximation of $\sqrt{e/2}$. The underlined step (the main loop in GAUSSIANSAMPLER) is where we introduce our faults.

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: $\mathbf{v} = \mathbf{b}\cdot\mathbf{s}' + \mathbf{e}'' = \mathbf{a}\cdot\mathbf{s}\cdot\mathbf{s}' + \mathbf{e}\cdot\mathbf{s}' + \mathbf{e}''$, 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 NewHope protocol is given in Fig. 3.

2.5 Description of Frodo

The second key exchange scheme we consider has been introduced by Bos et al. at CCS 2016 [9]. This scheme is a practical demonstration of an efficient lattice scheme based on the hardness of LWE (in contrast with NewHope, which relies on the hardness of the *Ring*-LWE problem). Its performance has been evaluated in a "real-world" setting by benchmarking its implementation within the TLS protocol, when coupled with ECDSA certificates. As in NewHope key exchange, the reconciliation elements can be written as $\mathbf{B}'\cdot\mathbf{S} = \mathbf{S}'\cdot\mathbf{A}\cdot\mathbf{S} + \mathbf{E}'\cdot\mathbf{S}$ and $\mathbf{V} = \mathbf{S}'\cdot\mathbf{B} + \mathbf{E}'' = \mathbf{S}'\cdot\mathbf{A}\cdot\mathbf{S} + \mathbf{S}'\cdot\mathbf{E} + \mathbf{E}''$ for Alice and Bob, but in contrast with NewHope, the secrets are no longer elements of a ring, but vectors of integers. The full outline of the Frodo protocol is given in Fig. 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

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

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 [3]. Note that computations are taken $\bmod q$.

Alice		Bob
$seed \leftarrow^{\$} \mathcal{U}(\{0,1\}^s)$		
$\mathbf{A} \leftarrow$ Gen$(seed)$		
$\underline{\mathbf{S}, \mathbf{E} \leftarrow \chi(\mathbb{Z}_q^{n\times\bar{n}})}$		$\mathbf{S}', \mathbf{E}', \mathbf{E}'' \leftarrow^{\$} \chi(\mathbb{Z}_q^{\bar{m}\times n})$
$\mathbf{B} \leftarrow \mathbf{A}\cdot\mathbf{S} + \mathbf{E}$	$\xrightarrow{(\mathbf{B}, seed)}$	$\mathbf{A} \leftarrow$ Gen$(seed)$
		$\mathbf{B}' \leftarrow \mathbf{S}'\cdot\mathbf{A} + \mathbf{E}'$
		$\mathbf{V} \leftarrow \mathbf{S}'\cdot\mathbf{B} + \mathbf{E}''$
	$\xleftarrow{(\mathbf{B}', \mathbf{C})}$	$\mathbf{C} \leftarrow \langle \mathbf{V} \rangle_{2^B}$
$K \leftarrow$ 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 $\bar{n}, \bar{m}, B \in \mathbb{Z}$. The matrix $\mathbf{A} \in \mathbb{Z}^{n\times n}$ is generated from seed via a pseudo-random function GEN. Recommended parameters are $n = 752, q = 2^{15}$, χ an approximate discrete Gaussian distribution of variance 1.75. Note that computations are taken $\bmod q$.

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 Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2018.2833119, 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,[3] 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 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 mth 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 in terms of invertibility (which is well verified in practice), $\mathbf{c}$ should be invertible with probability about $(1-1/q)^n$, which is over 95 percent for all proposed BLISS parameters. We thus get an equation of the form

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

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 J be a subset of $\{0, \ldots, n-1\}$ of cardinality ℓ, and $\varphi_J : \mathbb{Z}^n \to \mathbb{Z}^J$ be the projection $(u_j)_{0 \le j < n} \mapsto (u_j)_{j \in J}$. Then we also have that $\varphi_J(\mathbf{z}_1)$ is a close vector to the sublattice L_J of $\mathbb{Z}^J$ generated by $q\mathbb{Z}^J$ and the images under φ_J of the $\mathbf{w}_i$'s; and the difference should be $\varphi_J(\mathbf{s}_1)$.

Equivalently, using Babai's nearest plane approach to the closest vector problem [4], we hope to show that $\big(\varphi_J(\mathbf{s}_1), B\big)$, for a suitably chosen positive constant B, is the shortest vector in the sublattice L'_J of $\mathbb{Z}^J \times \mathbb{Z}$ generated by $\big(\varphi_J(\mathbf{v}), B\big)$ as well as the vectors $\big(\varphi_J(\mathbf{w}_i), 0\big)$ and $q\mathbb{Z}^J \times \{0\}$.

The volume of L'_J is given by

$$\mathrm{vol}(L'_J) = B \cdot \mathrm{vol}(L_J) = B \cdot \frac{\mathrm{vol}(q\mathbb{Z}^J)}{[L_J : q\mathbb{Z}^J]} = Bq^{\ell - r},$$

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

$$\begin{aligned}\lambda_J &\approx \sqrt{\frac{\ell+1}{2\pi e}} \cdot \mathrm{vol}(L'_J)^{1/(\ell+1)} \\ &\gtrsim \sqrt{\frac{\ell+1}{2\pi e}} \cdot B^{1/(\ell+1)} q^{1-(m+1)/(\ell+1)}.\end{aligned}$$

Thus, we expect that $\big(\varphi_J(\mathbf{s}_1), B\big)$ will actually be the shortest vector of L'_J provided that its norm is significantly smaller than this bound λ_J. Now $\varphi_J(\mathbf{s}_1)$ 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 $\big(\varphi_J(\mathbf{s}_1), B\big)$ 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_J 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

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

The denominator of the right-hand side of (2) 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 $\varphi_J(\mathbf{s}_1)$ 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 J 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.

3. In the other Fiat–Shamir schemes such as [34], the distribution of each coefficient is uniform in some interval rather than Gaussian, but this doesn't affect our attack strategy at all.

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

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

We attack the BLISS–II parameter set $(n, q, \sigma, \delta_1, \delta_2, \kappa) = (512, 12289, 10, 0.3, 0, 23)$ from [18]. 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.

Remark 1. A variant of that attack which is possibly slightly simpler consists in observing that $\varphi_J(\mathbf{s}_1)$ should be the shortest vector in the lattice generated by L_J and $\varphi_J(\mathbf{v})$. The bound on the lattice dimension becomes essentially the same as (2). The drawback of that approach, however, is that we obtain each $\varphi_J(\mathbf{s}_1)$ up to sign, and so one needs to use overlapping subsets J 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 [18], and even smaller with the improved analysis of BLISS–B [17].

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 et al. [21], 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.

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 Section 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. [35] 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 Fig. 2), but applying [35, Theorem 1], the technique is easily seen to work in both ways.

Moreover, if we start from a polynomial of the form $\boldsymbol{\zeta}\mathbf{F}$ where $\boldsymbol{\zeta}$ is of the form $\pm\mathbf{x}^\alpha$, then applying the previous steps yields the quadruple $(\boldsymbol{\zeta}\mathbf{f}, \boldsymbol{\zeta}\mathbf{g}, \boldsymbol{\zeta}\mathbf{F}, \boldsymbol{\zeta}\mathbf{G})$, 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 do not 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.

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

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

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 [21] for signatures. The simulation was carried out using our Sage implementation (see Appendix, available in the online supplemental material) on a single core of an Intel Xeon E5-2697v3 workstation, using 100 trial runs for each pair (ℓ, m).

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 et al. [41] (see also [25]), 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: see the Appendix, available in the online supplemental material, for an extended discussion). Moreover, the attack is quite fast even for relatively large values of m: only a couple of minutes for full key recovery for $m = 100$.

5 Attack on NewHope 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 done *mutatis mutandis*.

In this scheme, the commitment element consists of a polynomial $\mathbf{e}$, which intuitively plays the role of 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 with the distribution of $\mathbf{e}$ should cause some information leakage when sending the element $\mathbf{b}$ to Bob.

More formally, $\mathbf{e} \in \mathcal{R}_q$ is a ring element drawn from the centered binomial distribution ψ_{16}. 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: this is quite similar to the attack on BLISS.

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 mth 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 percent for the proposed NewHope parameters. We thus get an equation of the form

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

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 yield 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 (4) is roughly about 0.70 for the parameters sets. We thus expect to recover $\mathbf{s}$ if we can solve the shortest vector problem in a lattice of dimension slightly larger than $m/0.7 \approx 1.4 \cdot m$. This is quite feasible with the LLL algorithm for m up to about 40, and with BKZ for m up to 80 or so.

6 Attack on Frodo Key Exchange

Our second attack targets the Frodo key exchange protocol. See also the Appendix, available in the online supplemental material, for a discussion of an attack on Kyber [11], which has some similarity to the one below. Like in the previous attack we tamper the generation of the "commitment" matrices $\mathbf{E}, \mathbf{E}'$. By the inherent symmetry of the protocol, we describe the attack on Bob's side. The adaptation to Alice's side is straightforward, once one takes into account the simple observation the multiplication by the secret is performed on the other side of the public matrix A (left-multiplication for Bob and right-multiplication for Alice).

In this scheme, the commitment matrix consists on a vector $\mathbf{E}'$, which acts as an additive mask to the secret key matrix $\mathbf{S}'$ 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 matrix $\mathbf{B}'$ to Bob.

More formally, $\mathbf{E}' \in \mathbb{Z}_q^{n \times \bar{n}}$ is a matrix of integers, whose coefficients are independently drawn under a discrete Gaussian distribution. Hence its generation is then once again typically carried out coefficient by coefficient, allowing us to use faults to cause an early abort of this generation process. However, since we are dealing with matrices instead of vectors as in the previous attacks, there is a subtlety in the way the generation can be carried out. The entries of the matrix can be sampled column-by-column or row-by-row, and these two generation algorithms are not equivalent from the standpoint of our attack. In the former case an early abort on the outermost loop of the generation process leads to a matrix with a certain number of column correctly set and then a family of zero column vectors

$$\mathbf{E}' = [E'_1 \mid \mathbf{0}]. \tag{5}$$

In the latter case an early abort makes arise a matrix with full first rows, followed with a certain number of zero rows vectors

$$\mathbf{E}' = \left[\frac{E'_1}{\mathbf{0}}\right]. \tag{6}$$

6.1 Column-Wise Generation

We first focus on the case where the generation is carried out *column-by-column*, so that the fault attack yields a matrix $\mathbf{E}'$ of the form (5).

Precisely, let us consider a matrix $\mathbf{B}'$ obtained by forcing a termination of the loop for sampling $\mathbf{E}'$ after the mth iteration, with $m \ll n$ (that is, the resulting matrix $\mathbf{E}'$ has at most $m-1$ non-zero columns, collected in a plain matrix denoted by E'_1). Without loss of generality, we can assume the non-zero columns are 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, we have access to the pair $(\mathbf{A}, \mathbf{B}')$ of integer matrices, satisfying the following relation taken modulo q (notice that the block decomposition is taken to be consistent with the block decomposition of E in $[E'_1|\mathbf{0}]$)

$$\begin{aligned}[B'_1|B'_2] = \mathbf{B}' &= \mathbf{S}' \cdot \mathbf{A} + \mathbf{E}' \\ &= [S'_1|S'_2] \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right] + [E'_1|\mathbf{0}],\end{aligned}$$

yielding the system

$$\begin{cases} S'_1 \cdot A_1 + S'_2 \cdot A_3 &= B'_1 - E'_1 \\ S'_1 \cdot A_2 + S'_2 \cdot A_4 &= B'_2 \end{cases} \mod q.$$

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

$$\begin{cases} S'_1 \cdot A_1 + S'_2 \cdot A_3 &= B'_1 - E'_1 \\ S'_2 &= (B'_2 - S'_1 \cdot A_2) \cdot A_4^{-1} \end{cases} \mod q, \tag{7}$$

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 of 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

$$\tilde{B} \cdot \tilde{A}^{-1} = S'_1 + E'_1 \cdot \tilde{A}^{-1} \qquad \bmod q.$$

Considering separately each of the n rows yields n equations on vectors

$$\forall i, \left[\tilde{B}\tilde{A}^{-1}\right]_i = \left[S'_1\right]_i + \underbrace{\left[E'_1 \cdot \tilde{A}^{-1}\right]_i}_{\in \mathrm{Span}_{\mathbb{Z}}((\mathbf{e}_{i,j}\tilde{A}^{-1})_{0 \le j \le m-1})} \mod q,$$

where $\mathbf{e}_{i,j}$ is the (i,j)th elementary matrix. The attack is then mounted to recover each rows, exactly in the same fashion as before, this time considering the n lattices of rank m, $(\mathcal{L}_i)_{0 \le i < m}$, generated by the $\mathbf{w}_j^{(i)} = \tilde{A}^{-1}(\mathbf{e}_{i,j})_j \bmod q$ for $j = 0, \ldots, m-1$ and $q\mathbb{Z}^m$. Notice that here the recovery can be performed directly by solving the SVP problem in these lattices since their common rank, m is supposed to be small. Once S'_1 is recovered, Equation (7), allows to recover directly S'_2 by linear algebra.

6.2 Row-Wise Generation

We now turn to the case where the generation is conducted *row-by-row*, so that $\mathbf{E}'$ is of the form (6).

More precisely, consider a matrix $\mathbf{B}'$ obtained by forcing a loop abort during the sampling $\mathbf{E}'$ after the m-h iteration, with $m \ll n$ (that is, the resulting matrix $\mathbf{E}'$ has at most $m-1$ non-zero rows, collected in a full matrix denoted by E'_1). Without loss of generality, we can assume the non-zero columns are the first ones. Once again, we get access to the pair $(\mathbf{A}, \mathbf{B}')$ of integer matrices, satisfying the following relation taken modulo q

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

$$\left[\frac{B'_1}{B'_2}\right] = \mathbf{B}' = \mathbf{S}' \cdot \mathbf{A} + \mathbf{E}' = \left[\frac{S'_1 \mid S'_2}{S'_3 \mid S'_4}\right]\left[\frac{A_1}{A_2}\right] + \left[\frac{E'_1}{\mathbf{0}}\right].$$

In this case, no factorization similar to the one appearing in the previous section allows to reduce the problem to a smaller instance. As a result, in this setting, it does not seem possible to mount the loop-abort fault attack on Bob's side!

However, note that in this case, the scheme is vulnerable on *Alice's side*. Indeed, taking the same notations one have, by tampering the generation of $\mathbf{E}$ to get the following equation once $(\mathbf{A}, \mathbf{B})$ are obtained

$$\left[\frac{B_1}{B_2}\right] = \mathbf{B} = \mathbf{A} \cdot \mathbf{S} + \mathbf{E} = \left[\frac{A_1 \mid A_2}{A_3 \mid A_4}\right]\left[\frac{S_1}{S_2}\right] + \left[\frac{E_1}{\mathbf{0}}\right],$$

yielding the system

$$\begin{cases} A_1 \cdot S_1 + A_2 \cdot S_2 &= B_1 - E_1 \\ A_3 \cdot S_1 + A_4 \cdot S_2 &= B_2 \end{cases} \bmod q,$$

which is of the same shape of Equation (7). The discussion of Paragraph 6.1 can then be directly adapted to recover S_1 and S_2.

7 Feasibility of the Faults

In this section, we investigate how an attacker may obtain helpful faulty values for the proposed attacks. We base our discussion on two available implementations of BLISS signature, namely the software implementation of Ducas and Lepoint [19] and the FPGA implementation by Pöppelmann et al. [54], and on Prest's software implementation of the GPV-based scheme of Ducas et al. [55]. The discussion extends to the other signature schemes and to the key exchange protocols directly, since the type of faulty behavior we want to cause (loop aborts) is the same in all cases.

We emphasize 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 for 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 to 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 overly ambitious are actually achievable during lab experiments with good reproducibility. That is true for single-bit faults both using EM [48] or laser injections [12]. There are also many reported instruction-skipping faults in the literature. One example of such faults is the work of Rivière et al. on ARMv7 instruction cache [56].

In the next sections, we discuss which fault models[5] may lead to faulty signatures that are relevant with respect to the attacks presented in this paper.

7.2 Fault Attacks on Software Implementations

In BLISS, the polynomial $\mathbf{y}_1$ can be generated using a loop over the n coefficients. This is indeed how the implementation in [19] does it: a loop constructs the polynomials $\mathbf{y}_1$ and $\mathbf{y}_2$ coefficient by coefficient using a Gaussian sampler (function `Sign::signMessage`). The condition to perform the attack successfully is not very demanding since we only try to ensure that three quarters or so of the coefficients of $\mathbf{y}_1$ are fixed to zero. Such a result can easily be obtained by exiting the loop after a few iterations. A random fault on the loop counter or skipping the jump operation will lead to such a 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 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 dozens of loop iterations. Moreover, we may use side-channel analysis to detect the loop iteration pattern to trigger the fault injection. That pattern is likely to be detected after much fewer than 50 iterations, and thus it seems that the synchronization here will be quite easy.

Similarly, the short random polynomials $\mathbf{R}$ and $\mathbf{r}$ used in the GPV scheme are generated in a single loop [55] 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 is large since, according to the results described in Table 2, if one collects say 50 faulty signatures, one can tolerate close to 50 executed iterations each time.

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

7.3 Fault Attacks on Hardware Implementations

In BLISS, the generation of the polynomial $\mathbf{y}_1$ requires sampling n random coefficients, where $n = 512$ in the most common parameter settings. Even in hardware, it is highly unlikely that all these coefficients are obtained at the same time (n is simply too large); thus, the generation of $\mathbf{y}_1$ is most likely to be sequential. This is the case in the implementation

5. We only focus on single fault attacks here.

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 (in this case, the BLISS coprocessor) 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 supported by the attack helps the adversary 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 the previous/initial value of memory cells.

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—as well as the localization of the `rstram` signal.[6]

8 Implementation of the Faults

In this section, we further validate the proposed attack model in software against two different platforms. We consider the specific case of NewHope, and compile a lightweight implementation of it for the SPARC-based 32-bit LEON3 processor, as well as for the 8-bit Atmel XMEGA128 microcontroller.

In the case of the LEON3 code, we carry out a simulation of our loop abort fault using a tool [50] based on the `tsim` LEON3 emulator. And in the case of the XMEGA code, we actually mount the attack in practice using the ChipWhisperer-Lite side-channel and fault analysis evaluation board [47]; we use clock glitches to inject the fault, and successfully manage to cause early loop aborts.

6. Since $\mathbf{y}_1$ is not output directly, checking if the attack actually worked can be a bit 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 than for a $\mathbf{z}_1$ corresponding to a dense $\mathbf{y}_1$.

8.1 Simulated Faults on LEON3

To confirm that an instruction-skipping fault actually leads to a full secret recovery, we have first simulated fault attacks on NewHope (both on $\mathbf{s}$ and $\mathbf{e}$) using the tool made available by authors of [50]. 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) with freely available source code. 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 $\mathbf{e}$ and $\mathbf{s}$, the generation from the "small" (centered binomial) distribution in NewHope is performed by first generating some pseudorandom bits using Chacha20 then processing each resulting 32-bit word to convert it into a binomially distributed value (function `poly_get_noise`). Our fault injection simulation targets this loop, in particular, attempting to cause an early abort in its execution.

More precisely, sufficiently many random bytes are first 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 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 [50] 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. These faulty outputs have then been used to feed a `sage` attack script recovering the key (when the number of iterations is small enough).

This confirms that skipping an instruction will lead to the expected behavior according to the widely deployed simulator for the LEON3 core. In particular, this shows that there is no side-effect of the fault nor of the targeted code that would render the proposed attack more complicated or inefficient than expected.

8.2 Concrete Faults on XMEGA with ChipWhisperer

Finally, we concretely mount our fault attack against the same software implementation of NewHope, this time compiled for the Atmel XMEGA128D4 8-bit microcontroller coming with the ChipWhisperer-Lite evaluation board [47].

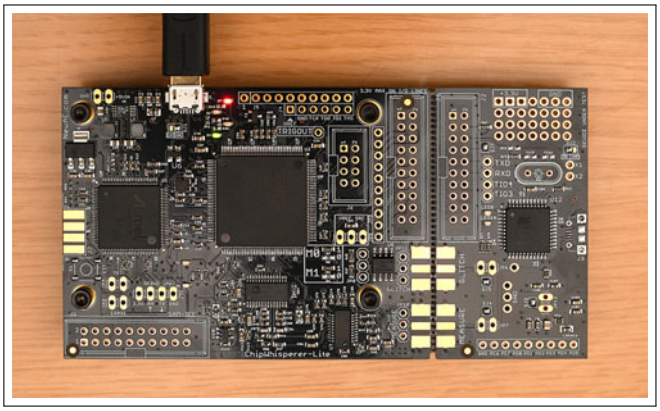
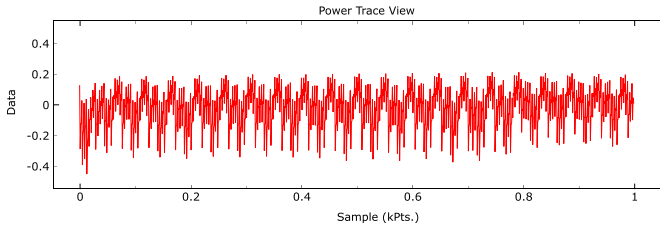
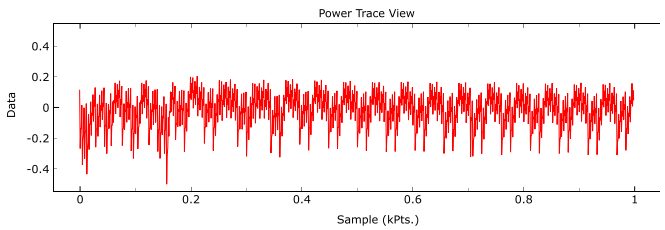


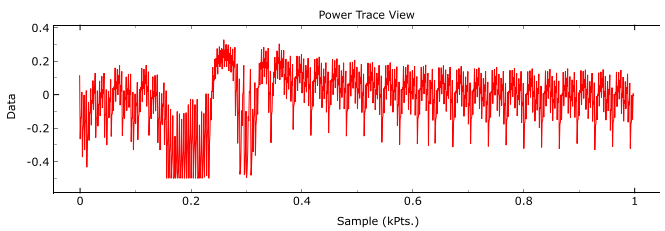
Fig. 5. The ChipWhisperer-Lite evaluation board, with XMEGA microcontroller target attached.



(a) Trace without fault injection



(b) Trace with single glitch at offset 33



(c) Trace with 20 glitched cycles from offset 33

Fig. 6. Power traces of the target loop in NewHope on the XMEGA microcontroller, with and without faults. Sampling rate is $4\times$ the clock frequency.

The ChipWhisperer-Lite (Fig. 5) enables the injection of clock and voltage glitches during the execution of the compiled code on the microcontroller. We successfully use clock glitches to cause an early loop abort in the generation of the polynomial **e**, allowing a full secret key recovery with the attack of Section 5.

While the XMEGA128D4 has a substantial amount of flash memory to store the compiled code, it has only 8 KB of RAM, which makes it impossible to run the entire NewHope code on the microcontroller. As a result, we have to do some offline precomputations on the host computer and store the result in the compiled binary before the experiment. However, the relevant part of the code as far as our attack is concerned, namely the loop generating the polynomial **e** from pseudorandom bits generated in advance (or the copy moving the result to the target area in memory), can be executed in full, and we can carry out our clock glitches against it. In addition, we note that the XMEGA executes the more time-consuming operations in NewHope like the direct and inverse number-theoretic transforms without problem.

Another way in which we "cheat" slightly in our fault attack is by raising a hardware trigger in the code before the execution of the loop of interest. Using more advanced tooling than the ChipWhisperer-Lite, however, it would be easy to replace that manually inserted trigger by a trigger on the waveform of the power trace before entering the loop. Such a triggering mechanism is for example available in the ChipWhisperer-Pro (which can detect patterns in power traces using SAD) and most higher-end oscilloscopes.

Apart from those slight changes to accommodate for our very lightweight target, we are able to mount the glitch attack on an almost unmodified implementation of NewHope. It takes some time to select proper parameters for the clock glitches, so as to obtain reliably reproducible faults, but the "glitch explorer" included in the ChipWhisperer Capture software makes this a relatively painless exercise. In the end, we ended up selecting one-shot clock glitches of width 2.5 percent of a clock period and offset -5.5 percent of a period: in other words, at the clock cycle where the fault is injected, the square signal providing the clock input for the microcontroller is modified by XORing it with a short rectangular pulse of the same amplitude as the original signal, slightly before the normal raising edge. This reliably caused a perturbation in the execution of the code on the microcontroller, which can be observed on power traces: see Fig. 6 (the effect is especially visible when we repeat the glitch over multiple cycles, as in Fig. 6c, but such a long glitch causes the device to reset, and hence is not useful after parameter selection is complete). Note that although it took us some time to choose those parameters, reliable faults can be obtained from a relatively large range of parameters: see the short discussion in the Appendix, available in the online supplemental material, and its accompanying figure.

When inserting the glitch at a suitable offset from the trigger, we were able to cause an early loop abort after any chosen number of iterations. More precisely, on the compiled code used in our experiments, inserting the glitch at offset $12k + 9$ clock cycles after the trigger is raised,[7] we obtained an exit of the loop after exactly k iterations, and hence a polynomial **e** of degree exactly $k - 1$ allowing to recover the entire secrets for all small values of k. This result is highly reproducible in our setup: among 100 injected faults at various offsets, we obtained 97 successful loop aborts after the expected number of iterations.

9 Possible 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

7. Like the parameters for the glitches, this offset pattern was also found by trial and error using the glitch explorer in ChipWhisperer Capture.

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 has 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 percent 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 percent, and hence be able to attack. The same observation applies to the key exchange protocols.

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. Again, this method is also applicable to NewHope (where the larger size of the coefficient vectors ensures that a smaller $\varepsilon \approx 0.3$ suffices to achieve a statistical distance below 2^{-128}). It can also be adapted relatively easily to Frodo and Kyber as well (in fact, for Frodo, one could even require all the rows and columns of the noise matrices to be non-zero for maximum safety, although the countermeasure then becomes somewhat costly).

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.

10 Conclusion and Outlook

The main takeaway of our results is that physical attacks, and particularly faults, can be a devastating threat against lattice-based schemes. Extensive research has been conducted on fault attacks against RSA and discrete logarithm-based schemes, so that both threats and countermeasures are well-understood. As lattice-based cryptography advances closer to real-world deployment, similar efforts need to be devoted to their fault analysis.

This paper considered one class of attacks, loop-abort faults, that has wide-ranging applicability to lattice-based signature schemes and key exchange protocols. It is also a relatively cheap attack to protect against, so introducing countermeasures in actual implementations is strongly recommended.

Future work could extend our results both by widening the scope of possible schemes vulnerable to that class of attacks (e.g., loop-abort faults have also recently been shown to be a threat to isogeny-based cryptography [28]), and by considering other, possibly more advanced fault attacks on the same schemes. Another possible avenue for further research could be the provable security of certain fault countermeasures on lattice-based (such as the ones discussed above) within a sufficiently restricted fault model. Results of this type have been obtained before on RSA [5], [14], for example, although it is usually not hard to bypass the limits of the corresponding fault model [26].

Acknowledgments

We acknowledge the support of the French Programme d'Investissement d'Avenir under national project RISQ. This work is also partially supported by the European Union projects PROMETHEUS (Horizon 2020 Research and Innovation Program, grant 780701) and HEAT (Horizon 2020 Research and Innovation Program, grant 644209).

References

[1] S. Akleylek, N. Bindel, J. A. Buchmann, J. Krämer, and G. A. Marson, "An efficient lattice-based signature scheme with provably secure instantiation," in *Proc. Int. Conf. Cryptology Africa*, 2016, pp. 44–60.

[2] E. Alkim, N. Bindel, J. A. Buchmann, Ö. Dagdelen, E. Eaton, G. Gutoski, J. Krämer, and F. Pawlega, "Revisiting TESLA in the quantum random oracle model," in *Proc. Int. Workshop Post-Quantum Cryptography*, 2017, pp. 143–162.

[3] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange - A new hope," in *Proc. USENIX Secur. Symp.*, 2016, pp. 327–343.

[4] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.

[5] G. Barthe, F. Dupressoir, P. Fouque, B. Grégoire, M. Tibouchi, and J. Zapalowicz, "Making RSA-PSS provably secure against non-random faults," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2014, pp. 206–222.

[6] I. Biehl, B. Meyer, and V. Müller, "Differential fault attacks on elliptic curve cryptosystems," in *Proc. Annu. Int. Cryptology Conf.*, 2000, pp. 131–146.

[7] N. Bindel, J. A. Buchmann, and J. Krämer, "Lattice-based signature schemes and their sensitivity to fault attacks," in *Proc. Workshop Fault Diagnosis Tolerance Cryptography*, 2016, pp. 63–77.

[8] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *J. Cryptology*, vol. 14, no. 2, pp. 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 *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1006–1018.

[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 *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 553–570.

[11] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé, "CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2018, pp. 634–648.

[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 *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2015, pp. 177–182.

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

[14] J. Coron and A. Mandal, "PSS is secure against random fault attacks," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2009, pp. 653–666.

[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?" Physical Review X, American Physical Society Physics, vol. 6, no. 3, pp. 31015–31029, 2016.

[16] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem," *IACR Cryptology ePrint Archive*, vol. 2012, 2012, Art. no. 688.

[17] L. Ducas, "Accelerating BLISS: The geometry of ternary polynomials," *IACR Cryptology ePrint Archive*, vol. 2014, 2014, Art. no. 874.

[18] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal Gaussians," in *Proc. Advances Cryptology*, 2013, pp. 40–56.

[19] L. Ducas and T. Lepoint, "A proof-of-concept implementation of BLISS." (2014). [Online]. Available: http://bliss.di.ens.fr

[20] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - Dilithium: Digital signatures from module lattices," *IACR Cryptology ePrint Archive*, vol. 2017, 2017, Art. no. 633.

[21] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over NTRU lattices," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2014, pp. 22–41.

[22] L. Ducas and P. Q. Nguyen, "Learning a Zonotope and more: Cryptanalysis of NTRUSign countermeasures," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2012, pp. 433–450.

[23] T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi, "Loop-abort faults on lattice-based Fiat–Shamir and hash-and-sign signatures," in *Proc. Int. Conf. Sel. Areas Cryptography*, 2016, pp. 140–158.

[24] T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi, "Side-channel attacks on BLISS lattice-based signatures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1857–1874.

[25] F. Fontein and P. Wocjan, "On the probability of generating a lattice," *J. Symbolic Comput.*, vol. 64, pp. 3–15, 2014.

[26] P. Fouque, N. Guillermin, D. Leresteux, M. Tibouchi, and J. Zapalowicz, "Attacking RSA-CRT signatures with faults on Montgomery multiplication," *J. Cryptographic Eng.*, vol. 3, no. 1, pp. 59–72, 2013.

[27] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," Specifications v1.0, 2017. [Online]. Available: https://falcon-sign.info/falcon.pdf

[28] A. Gélin and B. Wesolowski, "Loop-abort faults on supersingular isogeny cryptosystems," in *Proc. Int. Workshop Post-Quantum Cryptography*, 2017, pp. 93–106.

[29] C. Gentry, J. Jonsson, J. Stern, and M. Szydlo, "Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001, " in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2001, pp. 1–20.

[30] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. Annu. ACM Symp. Theory Comput.*, 2008, pp. 197–206.

[31] C. Gentry and M. Szydlo, "Cryptanalysis of the revised NTRU signature scheme," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2002, pp. 299–320.

[32] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Proc. Annu. Int. Cryptology Conf.*, 1997, pp. 112–131.

[33] L. Groot Bruinderink, A. Hülsing, T. Lange, and Y. Yarom, "Flush, Gauss, and reload - A cache attack on the BLISS lattice-based signature scheme," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2016, pp. 323–345.

[34] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2012, pp. 530–547.

[35] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSign: Digital signatures using the NTRU lattice," in *Proc. Cryptographers' Track RSA Conf.*, 2003, pp. 122–140.

[36] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, and W. Whyte, "Practical signatures from the partial Fourier recovery problem," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2014, pp. 476–493.

[37] J. Howe, T. Pöppelmann, M. O'Neill, E. O'Sullivan, and T. Güneysu, "Practical lattice-based digital signature schemes," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 3, 2015, Art. no. 41.

[38] A. A. Kamal and A. M. Youssef, "Fault analysis of the NTRUSign digital signature scheme," *Cryptography Commun.*, vol. 4, no. 2, pp. 131–144, 2012.

[39] V. Lyubashevsky, "Fiat–Shamir with aborts: Applications to lattice and factoring-based signatures," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2009, pp. 598–616.

[40] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2012, pp. 738–755.

[41] G. Maze, J. Rosenthal, and U. Wagner, "Natural density of rectangular unimodular integer matrices," *Linear Algebra Appl.*, vol. 434, no. 5, pp. 1319–1324, 2011.

[42] C. A. Melchor, X. Boyen, J. Deneuville, and P. Gaborit, "Sealing the leak on classical NTRU signatures," in *Proc. Int. Workshop Post-Quantum Cryptography*, 2014, pp. 1–21.

[43] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2012, pp. 700–718.

[44] D. Naccache, P. Q. Nguyen, M. Tunstall, and C. Whelan, "Experimenting with faults, lattices and the DSA," in *Proc. Int. Workshop Public Key Cryptography*, 2005, pp. 16–28.

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

[46] P. Q. Nguyen and O. Regev, "Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures," *J. Cryptology*, vol. 22, no. 2, pp. 139–160, 2009.

[47] C. O'Flynn and Z. D. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Des.*, 2014, pp. 243–260.

[48] S. Ordas, L. Guillaume-Sage, K. Tobich, J. Dutertre, and P. Maurine, "Evidence of a larger EM-induced fault model," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, 2014, pp. 245–259.
[49] D. Page and F. Vercauteren, "A fault attack on pairing-based cryptography," *IEEE Trans. Comput.*, vol. 55, no. 9, pp. 1075–1080, Sep. 2006.
[50] C. Patrick, B. Yuce, N. F. Ghalaty, and P. Schaumont, "Lightweight fault attack resistance in software using intra-instruction redundancy," in *Proc. Int. Conf. Sel. Areas Cryptography*, 2016, pp. 231–244.
[51] C. Peikert, "Lattice cryptography for the internet," in *Proc. Int. Workshop Post-Quantum Cryptography*, 2014, pp. 197–219.
[52] C. Peikert, "A decade of lattice cryptography," Cryptology ePrint Archive, Report 2015/939, 2015. [Online]. Available: http://eprint.iacr.org/
[53] P. Pessl, L. Groot Bruinderink, and Y. Yarom, "To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1843–1855.
[54] T. Pöppelmann, L. Ducas, and T. Güneysu, "Enhanced lattice-based signatures on reconfigurable hardware," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2014, pp. 353–370.
[55] T. Prest, "Implementation of the GPV-based scheme of Ducas et al." (2016). [Online]. Available: https://github.com/tprest/Lattice-IBE
[56] 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 *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, 2015, pp. 62–67.
[57] W. Stein, et al., "Sage mathematics software (version 7.0)," 2016. [Online]. Available: http://www.sagemath.org
[58] M. Taha and T. Eisenbarth, "Implementation attacks on post-quantum cryptographic schemes," *IACR Cryptology ePrint Archive*, 2015.

Thomas Espitau is currently working toward the second-year PhD degree in computer science at UPMC under the joint supervision of Antoine Joux and Pierre-Alain Fouque. He is an alumnus of École normale supérieure de Cachan (Cachan, France). One of his main research topics is lattice-based cryptography and the analysis of lattice reduction algorithms.

Pierre-Alain Fouque received the PhD degree in computer science from the University Paris VII under Jacques Stern, in 2001. He then served as assistant professor with the Computer Science Department, École normale supérieure (Paris, France) from 2003 before moving to the University of Rennes I (Rennes, France) as in 2012. He is now full professor of computer science with Rennes I and the Institut Universitaire de France. His research interests cover all aspects of cryptanalysis and security in general.

Benoît Gérard received the PhD degree in computer science from the INRIA Rocquencourt under the supervision of Jean-Pierre Tillich, in 2010. He then joined the UCL CryptoGroup (Louvain, Belgium) as a postdoctoral researcher before taking his current research position at DGA.MI (Rennes, France), part of the French defense ministry. He is an expert of side-channel analysis and physical attacks.

Mehdi Tibouchi received the PhD degree in computer science from the University Paris VII and the University of Luxembourg, in 2011 after doctoral studies at École normale supérieure (Paris, France) under the supervision of David Naccache and Jean-Sébastien Coron. He then joined NTT Corporation (Tokyo, Japan), where he has been carrying out research in cryptology ever since. His research interests cover several mathematical aspects of public-key cryptography and cryptanalysis.