Breaking the Shuffling Countermeasure in Lattice-Based Signature Schemes

Alexis Bagia

29.07.2022
Version: Final

Technische Universität Berlin

Professur Security in Telecommunications
Institut für Softwaretechnik und Theoretische Informatik

Bachelor's Thesis

Breaking the Shuffling Countermeasure in Lattice-Based Signature Schemes

Alexis Bagia

1. Reviewer	Prof. Dr. Jean-Pierre Seifert Professur Security in Telecommunications Technische Universität Berlin
2. Reviewer	Prof. Dr. Thorsten Koch Fachgebiet Software und Algorithmen für die diskrete Optimierung Technische Universität Berlin
Supervisor	M.Sc. Vincent Quentin Ulitzsch

29.07.2022

Alexis Bagia
Breaking the Shuffling Countermeasure in Lattice-Based Signature Schemes
Bachelor's Thesis, 29.07.2022
Reviewers: Prof. Dr. Jean-Pierre Seifert and Prof. Dr. Thorsten Koch
Supervisor: M.Sc. Vincent Quentin Ulitzsch

Technische Universität Berlin

Institut für Softwaretechnik und Theoretische Informatik
Professur Security in Telecommunications
Ernst-Reuter-Platz 7
10587 Berlin

Affidavit

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

Berlin, 29.07.2022

Alexis Bagia

Eidesstaatliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, 29.07.2022

Alexis Bagia

Abstract

The current asymmetric cryptographic primitives which we use in our daily life are based on the hardness of the factoring and (elliptic curve) discrete logarithm problem. Quantum computers may in the near future become powerful enough to solve these problems and break all of our currently used asymmetric cryptographic primitives. Because of this growing threat the the National Institute of Standards and Technology (NIST) called for proposals of post-quantum secure signature schemes and key encapsulation mechanism (KEM) already back in 2016. This resulted in extensive research on the proposed signatures schemes and KEMs, which provides reasonable assurance that they are very secure in theory. In practice all algorithms are run on real hardware like microcontrollers and smart cards which are susceptible to physical attacks such as fault attacks. Research on possible fault attacks showed various critical parts of signature algorithms which need to be protected. One particular part in Fiat-Shamir based signature schemes is the sampling of the so-called masking polynomial, which is used to blind the linear linear relationship between the secret key and a publicly known value. If the sampling of the coefficients of this polynomial is aborted, due to fault, some of the polynomial coefficients will be uninitialized. This will cause information about the secret polynomial to leak which can be used to recover the secret polynomial. Different countermeasures have been proposed to prevent this attack. One among them is the shuffling countermeasure. The idea behind this countermeasure is to shuffle the order of the coefficients after they have been sampled.

Our results show that this countermeasure is not as effective as believed. Despite this shuffling countermeasure implemented we were still able to recover the secret key. Our attack is based on Integer Linear Programs (ILPs) to recover the secret polynomial. We show the effectiveness of our attack by extensive simulations for the signature schemes Dilithium, qTESLA as well as BLISS. Our results highlights that the shuffling countermeasure alone is not sufficient to prevent fault attacks on the masking polynomial. Additional countermeasure are needed to properly secure upcoming microcontrollers and smart cards running post-quantum secure signature schemes against sophisticated attackers.

Zusammenfassung

Die derzeitigen asymmetrischen kryptographischen Primitive, die wir in unserem täglichen Leben verwenden, basieren auf der Schwere des Faktorisierungsproblems und des diskreten Logarithmus (auf elliptischen Kurven). Quantencomputer könnten in naher Zukunft leistungsstark genug sein, um diese Probleme zu lösen und somit alle derzeit verwendeten asymmetrischen kryptografischen Primitive zu brechen. Aufgrund dieser wachsenden Gefahr hat das National Institute of Standards and Technology (NIST) bereits im Jahr 2016 zur Einreichung von Vorschlägen für post-quantum-sichere Signaturverfahren und Schlüsselkapselungsmechanismen (KEM) aufgerufen. Daraufhin wurden die vorgeschlagenen Signaturverfahren und KEMs ausgiebig erforscht und in der Theorie als sehr sicher eingestuft. In der Praxis werden alle Algorithmen auf echter Hardware wie Mikrocontrollern und Chipkarten ausgeführt, die anfällig für physische Angriffe wie Fehlerangriffe sind. Die Forschung zu möglichen Fehlerangriffen zeigte auf, dass verschiedene kritische Bereiche von Signaturalgorithmen geschützt werden müssen. Ein besonderer Teil der auf Fiat-Shamir mit Abbruch basierenden Signaturverfahren ist die Generierung des so genannten Maskierungspolynoms, das verwendet wird, um die lineare Beziehung zwischen dem geheimen Schlüssel und einem öffentlich bekannten Wert zu verbergen. Wenn die Generierung der Koeffizienten dieses Polynoms aufgrund eines Fehlers abgebrochen wird, werden einige der Polynomkoeffizienten nicht initialisiert. Dies führt zum Preisgabe von Informationen über das geheime Polynom, welche zur vollständigen Wiederherstellung des geheimen Polynoms verwendet werden können. Es wurden verschiedene Gegenmaßnahmen vorgeschlagen, um diesen Angriff zu verhindern. Eine davon ist die Mischen-Gegenmaßnahme. Die Idee hinter dieser Gegenmaßnahme besteht darin, die Reihenfolge der Koeffizienten nach der Abtastung zu mischen.

Unsere Ergebnisse zeigen, dass diese Gegenmaßnahme nicht so effektiv ist wie angenommen. Trotz dieser Gegenmaßnahme war es uns möglich, den geheimen Schlüssel wiederherzustellen. Unser Angriff basiert auf ganzzahliger linearen Optimieung (Integer Linear Program, ILP), um das geheime Polynom wiederherzustellen. Wir zeigen die Wirksamkeit unseres Angriffs durch umfangreiche Simulationen für die Signaturverfahren Dilithium, qTESLA und BLISS. Unsere Resultate zeigen, dass die Mischen-Gegenmaßnahme allein nicht ausreicht, um Fehlerangriffe auf das Maskierungspolynom zu verhindern. Es sind zusätzliche Gegenmaßnahmen erforderlich, um zukünftige Mikrocontroller und Chipkarten, auf denen post-quantum-sichere Signaturverfahren eingesetzt werden, gegen raffinierte Angreifer zu schützen.

Contents

1 Introduction

The potential advent of large-scale quantum computers threatens to undermine the security of currently used public-key cryptography. Virtually all public-key cryptography that is used right now is based on either the hardness of factoring or discrete logarithm problems. Using Shor's algorithm [Sho99], large enough general-purpose quantum computers will be able to break these problems, rendering the respective public-key cryptography insecure. Because of this the National Institute of Standard and Technology (NIST), in an effort to standardize post-quantum cryptography, called for proposals for post-quantum secure cryptographic schemes [ST16], i.e., schemes that withstand quantum adversaries. The call encompasses signature, encryption and key encapsulations algorithms that can be used in a potential quantum computer era.

The standardization process is currently in the fourth round, with various proposed candidates being eliminated because they were found to be insecure or impractical and some candidates already selected for standardization [Ala+22]. The remaining and selected candidates have thus been subject to thorough theoretical analysis. However, the corresponding implementations require scrutiny on the implementation security, since existing cryptoanalytic efforts focused mainly on the schemes' designs. The foreseeable execution platforms for post-quantum cryptography algorithms include microcontrollers and smart cards, to, e.g., sign banking transactions. Smart cards and microcontrollers are subject to physical attacks, such as fault injection attacks. Especially for public-key cryptography like signature schemes, fault attacks can break the security of a device running a poorly implemented algorithm easily. Fault attacks cause the device signing a message to perform a miscalculation and therefore leak secret information. Past research demonstrated fault attacks that revealed the entire secret with only single or a few faulty signatures [Aum+03]. Designing efficient and effective countermeasures against fault-injection attacks is non-trivial, given the powerful attack primitive that they need to defend against.

In this thesis, we will investigate the effectiveness of one specific fault-injection countermeasure against so-called loop-abort fault attacks. We will focus in lattice based signature schemes because two out of the three signature schemes the NIST

will standardize are based in lattices. Additionally the currently recommended PQC signature scheme is Dilithium, a lattice based signature scheme.

Prior work by Espitau et al. [Esp+17] has shown that lattice-based signature schemes based on Fiat-Shamir with aborts (over ideal lattices) leak secret information if a masking polynomial is not properly sampled. Fault injections can cause such an improper sampling and thus can be used to reveal the secret key. To mitigate this attack, a proposed countermeasure is to randomize the order the sampling of the masking polynomial, i.e. shuffling the order of its coefficient [Esp+17, p. 155]. A second countermeasure was to check whether the upper few coefficients are zero and abort if too many are zero [Esp+17, p. 155].

1.1 Our Contribution

This thesis shows that the shuffling countermeasure is not as effective as believed. We formulate Integer Linear Programs (ILPs) that can recover the secret key effectively from faulted signatures, even if the shuffling countermeasure is present. The described attack affects the implementation security of signature schemes like BLISS, qTESLA as well as the signature scheme Dilithium [Duc+13b; Alk+20; Duc+18]. Dilithium is the winner of the NIST Post-Quantum-Cryptography standardization process [Ala+22]. It can thus be assumed that it will be implemented a lot in the near future. We will validate the effectiveness of our attack by extensive simulations for all the aforementioned PQC signature schemes.

Furthermore our results will show that we can even prevent the “checking for zero” countermeasure for Dilithium and qTESLA as we only require 1 or 2 zeroed coefficients per signature. For a single signature this can happen naturally with a high probability and thus can not be detected.

Our results will highlight the need for further research into securing the implementations of lattice-based signature schemes against fault attacks.

2 Background

This section will provide the reader with the required knowledge to understand the attack presented in this paper. First we will introduce the basic concepts of identification and signature schemes, then we will discuss the mathematical concepts of lattices and the computational problems lattice-based cryptography is based upon. Finally we will provide brief information about fault attacks in general.

2.1 Identification and signature schemes

In the public key setting signature schemes are constructions which allow to digitally sign messages. Messages can be can be anything that can be represented with zeroes and ones, e.g. PDF-files, software, videos and photos.

Just like analog signatures digital signatures have the property that they can be efficiently created, that they are valid just for a specific message, that everyone else can verify if a message-signature pair is authentic and that the signer is not able to deny a signature which is valid.

Formally Katz and Lindell [KL20, p. 442] describe a signature scheme as follows:

Definition 1 (Signature Scheme)**.** *A* (digital) signature scheme *consists of three probabilistic polynomial-time algorithms* (Gen, Sign, Vrfy) *such that:*

1. *The* key generation algorithm Gen *takes as input a security parameter* 1^n *and outputs a pair of keys (pk, sk). These are called the* public key *and* private key*, respectively. We assume that pk and sk each has length at least* n*. and that* n *can be determined from pk or sk.*

2. *The* Sign algorithm *takes as input a private key sk and a message* m *from some message space (that may depend on pk). It outputs a signature* σ*, and we write this as* $\sigma \leftarrow \text{Sign}_{sk}(m)$.

3. *The determininistic* verification algorithm Vrfy *takes as input a public key pk, a message* m*, and a signature* σ*. It outputs a bit* b*, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid*. We write this as* $b := \text{Vrfy}_{pk(m,\sigma)}$.

It is required that except with negligible probability over (pk, sk) output by $Gen(1^n)$*, it holds that* $Vrfy_{pk}(m, Sign_{sk}(m)) = 1$ *for every (legal) message* m*.*

If there is a function l *such that for every (pk, sk) output by* $Gen(1^n)$ *the message space is* $\{0,1\}^{l(n)}$*, then we say that* (Gen, Sign, Vrfy) *is a* signature scheme for messages of length $l(n)$.

2.2 Lattices

The general definition of a lattice is as follows:

Definition 2 (Lattice)**.** *A* n*-dimensional lattice* Λ *is a discrete subgroup of* $\mathbb{R}^n$.

Lattices are often defined by k linearly independent basis vectors $b_1, \ldots, b_k \in \mathbb{R}^n$. Given these basis vectors, the corresponding lattice $\Lambda(b_1, \ldots, b_k)$ is defined as $\Lambda(b_1, \ldots, b_k) = \{\sum_{i=1}^{k} a_i b_i | a_i \in \mathbb{Z}\}$. In cryptography the lattices of interest are mostly discrete subgroups of $\mathbb{Z}^n$ or $\mathbb{Z}_q^n$ and in practice most modern lattice based cryptographic lattice primitives use so called ideal lattices. These kind of lattices were first introduced by Lyubashevsky and Micciancio [LM06, p. 149] and they define them as follows:

Definition 3 (Ideal Lattice)**.** *An ideal lattice is an integer lattice* $\Lambda(b_1, \ldots, b_k) \subseteq \mathbb{Z}_n$ *with* $b_1, \ldots, b_k \in \mathbb{Z}_q$ *being the basis vectors of that lattice, such that* $\Lambda(b_1, \ldots, b_k) = \{g \bmod f | g \in I\}$ *for some monic polynomial* f *of degree* n *and ideal* $I \subseteq \mathbb{Z}[x]/(f)$.

These ideal lattices have the great advantage that they can reduce public and private key sizes and, if proper parameters are chosen, the ring operations (multiplying polynomials) can be performed very efficiently using the number theoretic transform (NTT) [AB74] speeding up the computations significantly. While the general definition of ideal lattice is hard to grasp, in this thesis we will only deal with the ideal lattice of the form $\mathbb{Z}[x]/(x^n + 1)$. These kind of lattices have the additional property that they are nega-cyclic, meaning that if the vector $(a_1, a_2, \ldots, a_{n-1}, a_n)$ is in in the lattice, the rotated and negated vector $(-a_n, a_1, \ldots, a_{n-2}, a_{n-1})$ is also in the lattice [Lyu08, p. 166].

Next we introduce two basic properties of lattices which are often of great interest:

Definition 4 (Shortest Vector)**.** *A shortest vector of a lattice Λ according to a norm $\|\cdot\|$ is a nonzero lattice vector $v \in \Lambda \setminus \{0\}$ such that for all $w \in \Lambda \setminus \{0\}$ it holds that $\|v\| \leq \|w\|$.*

Definition 5 (Closest Vector)**.** *Given a n-dimensional lattice Λ and a point $p \in \mathbb{R}^n$ and a norm $\|\cdot\|$ a closest vector to the point p is a vector $v \in \Lambda$ such that for all $w \in \Lambda$ it holds that $\|p - v\| \leq \|p - w\|$.*

2.2.1 Lattice Problems

Lattice problems are problems which are related to lattices. Such problems are used to build lattice-based cryptographic primitives. The Closest Vector Problem (CVP) and the Shortest Vector Problem (SVP) are the fundamental problems of lattice based cryptography as they have been well studied and they appear to be resistant against quantum computers. They are defined as follows:

Definition 6 (Shortest Vector Problem (SVP))**.** *Given k basis vectors $b_1, \ldots, b_k \in \mathbb{R}^n$ which define the lattice $\Lambda(b_1, \ldots, b_k)$, find a shortest vector in $\Lambda(b_1, \ldots, b_k)$.*

Definition 7 (Closest Vector Problem (CVP))**.** *Given k basis vectors $b_1, \ldots, b_k \in \mathbb{R}^n$ which define the lattice $\Lambda(b_1, \ldots, b_k)$ and a point $p \in \mathbb{R}^n$, find a closest vector $v \in \Lambda(b_1, \ldots, b_k)$ which is closest to p.*

In lattice-based cryptography new problems which are developed to build a basis for new cryptographic schemes often prove their hardness by reduction, i.e. showing that if one can solve the new problem, one can also solve CVP or SVP.

Using such reductions, the entire lattice based cryptography is based on solid foundations.

Short Integer Solutions Problem

The Short Integer Solution (SIS) problem was first introduced Ajtai in 1996 [Ajt96]. It is defined as follows:

Definition 8 (Short Integer Solution (SIS))**.** *Let $A \in \mathbb{Z}_q^{m \times n}$ chosen uniformly at random. According to a norm $\|\cdot\|$, find a vector $x \in \mathbb{Z}_q^n$ such that $Ax = 0$ and $||x||$ is small and non-trivial.*

Ajtai showed in his work [Ajt96] that that if one is able to solve the SIS problem with probability at least $\frac{1}{2}$ one can solve SVP with a probability near 1. Thus the SIS problem is at least as hard as SVP. The SIS problem is used by many latticed-based cryptographic schemes to prove their security.

Ring-LWE and the Module-Ring-LWE

The Learning with Errors over Rings problem, also referred as search-RLWE and the generalized variant which is known as Module-LWE or Ring-Module-LWE form the foundations for many lattice based signature and key encapsulation schemes.

Here we will give a more concrete definitions than the original ones [LPR10; BGV14] which will be just as general enough to cover all signature schemes we will discuss here.

First we will describe the ring we will be working with: Let n be a power of two. Let q be an odd prime. Let $\mathbb{Z}_q$ be the ring of integers modulo q. Then we define the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$.

Next both problems require some sort of error distribution χ. We will either use the discrete gaussian distribution centered at zero with standard derivation σ or a uniform distribution over a integer range, e.g. $[-(2^{21} - 1), 2^{21} - 1] \cap \mathbb{Z}$.

The search variant of the Ring-LWE problem is defined as follows:

Definition 9 (Ring-LWE)**.** *For a given $s \in \mathcal{R}_q$, given pairs of the form $(a_i, b_i = a_i \cdot s + e_i) \in \mathcal{R}_q \times \mathcal{R}_q$, where $a_i \in \mathcal{R}_q$ with coefficients uniformly at random and e_i chosen according to χ, find s.*

The Module-Ring-LWE problem can be seen as an generalization of the Ring-LWE problem:

Definition 10 (Module-Ring-LWE)**.** *For a given $\boldsymbol{s} \in \mathcal{R}_q^k$, given pairs of the form $(\boldsymbol{a}_i, b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i) \in \mathcal{R}_q^k \times \mathcal{R}_q^k$, where $\boldsymbol{a}_i \in \mathcal{R}_q^k$ with polynomials with coefficients uniformly at random and e_i chosen according to χ, find $\boldsymbol{s}$.*

We can see that for $k = 1$ the Module-Ring-LWE problem is the same as the Ring-LWE problem.

2.3 Fiat-Shamir with aborts

Fiat-Shamir with aborts describes a lattice based identification scheme as well as a lattice-based signature scheme, first introduced by Lyubashevsky [Lyu09]. This identification scheme uses a provable collision resistant lattice-based hash function. Based on this identification scheme Lyubashevsky used the Fiat-Shamir heuristic [FS87] to construct a lattice-based signature scheme. The general design is used by many more recent lattice-based signature schemes such ass BLISS, Dilithium as well as qTESLA.

The Fiat-Shamir with aborts identification scheme has the general structure of a sigma-protocol: First the prover commits to a value, the commitment value, then the challenger sends a value, the so called challenge, to the prover. The prover then performs an operation using the commitment value, challenge and secret key which proves the challenger that he indeed possesses the secret key. The novelty the Fiat-Shamir with aborts scheme is that the prover may abort the identification process. In this case nothing is proven to the challenger.

The aborts allow the scheme to use smaller masking values which reduces signature size while still keeping the scheme secure.

Both, the identification scheme as well as the signature scheme, can work over the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ and require a linear collision resistant hash function $h : \mathcal{R}_q^m \mapsto \mathcal{R}_q$ from a set $\mathcal{H}$ of collision resistant hash functions.

2.3.1 The identification scheme

The private key of the prover is a vector $\boldsymbol{s} \in D_s^m$ from a secret key domain D_s. The public key is the hash function $h \in H$ as well as $S = h(\boldsymbol{s})$.

The commitment value is calculated by choosing a random $\boldsymbol{y} \in D_y^m$ from a commitment domain and calculating $Y = h(\boldsymbol{y})$. Next the challenger chooses an $c \in D_c$ from a challenge domain D_c. The prover then computes $\boldsymbol{z} = \boldsymbol{s}c + \boldsymbol{y}$, aborts if $\boldsymbol{z}$ is not in a good range, i.e. $\boldsymbol{z} \notin G^m$, and otherwise sends $\boldsymbol{z}$ to the challenger.

The verifier accepts if $\boldsymbol{z} \in G^m$ and $h(\boldsymbol{z}) = Sc + Y$. The scheme is correct because $h(\boldsymbol{z}) = h(\boldsymbol{s}c) + h(\boldsymbol{y}) = h(\boldsymbol{s})c + h(y) = Sc + Y$.

This identification scheme is considered secure as it is proven that if an attacker can break the scheme he can also break an approximated version of SVP.

2.3.2 The signature scheme

The signature scheme is similar to the identification scheme. It differs from the identification scheme mainly in that it additionally requires a random oracle H and that the challenge c is replaced by $e = c = H(h(\boldsymbol{y}, \mu))$, where μ is the message. Replacing the challenge with the random oracle removes the needed interaction between the two parties. In detail the signature scheme is described in Figure 2.1. The scheme is correct because $H(h(\boldsymbol{z}) - Sc, \mu) = H(h(\boldsymbol{sc} + \boldsymbol{y}) - Sc, \mu) = H(Sc + Y - Sc, \mu) = H(Y, \mu)$.

Signing Key: $\widehat{\mathbf{s}} \xleftarrow{\$} D_s^m$
Verification Key: $h \xleftarrow{\$} \mathcal{H}(R, D, m), \mathbf{S} \leftarrow h(\widehat{\mathbf{s}})$
Random Oracle: $\mathrm{H} : \{0,1\}^* \to D_c$

$\mathrm{Sign}(\mu, h, \widehat{\mathbf{s}})$
1: $\widehat{\mathbf{y}} \xleftarrow{\$} D_y^m$
2: $\mathbf{e} \leftarrow \mathrm{H}(h(\widehat{\mathbf{y}}), \mu)$
3: $\widehat{\mathbf{z}} \leftarrow \widehat{\mathbf{s}}\mathbf{e} + \widehat{\mathbf{y}}$
4: if $\widehat{\mathbf{z}} \notin \mathrm{G}^m$, then goto step 1
5: output $(\widehat{\mathbf{z}}, \mathbf{e})$

$\mathrm{Verify}(\mu, \widehat{\mathbf{z}}, \mathbf{e}, h, \mathbf{S})$
1: Accept iff
$\widehat{\mathbf{z}} \in \mathrm{G}^m$ and $\mathbf{e} = \mathrm{H}(h(\widehat{\mathbf{z}}) - \mathbf{Se}, \mu)$

Fig. 2.1: The Fiat-Shamir with aborts signature scheme. Figure copied from [Lyu09, p. 611].

2.4 Lattice based signature schemes

In this section we will briefly explain the how the signature schemes we will attack work. For each signature we will explain the key generation, the signature generation and the signature verification.

2.4.1 Dilithium

Dilithium is currently the only PQC signature scheme that is both recommended and to be standardized by the NIST. [Ala+22] The authors proposed parameter sets for the NIST security levels 2, 3 and 5 [Bai+20, pp. 16–17].

The signature works over the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ where q is an odd prime. Simplified works as follows:

- **Key generation** First a $k \times l$ matrix $\boldsymbol{A}$ of polynomials with uniform coefficients in $\mathbb{Z}_q$ is sampled as well as two secret vectors $(\boldsymbol{s}_1, \boldsymbol{s}_2) \in S_\eta^l \times S_\eta^l$. S_η^k and S_η^l

denote the set of vectors of polynomials with coefficients with absolute value no more than η with k or l entries respectively. Then the vector $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2$ is calculated. Finally the public key is the pair $(\boldsymbol{A}, \boldsymbol{t})$ and the private key is the tuple $(\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{s}_2, \boldsymbol{s}_2)$.

- **Signature generation** First the vector $\boldsymbol{y} \in \tilde{S}^l_{\gamma_1}$ is sampled. The only difference between $\tilde{S}^l_{\gamma_1}$ and $S^l_{\gamma_1}$ is that $\tilde{S}^l_{\gamma_1}$ does not contain any polynomial with coefficient $-\gamma_1$. Then the high bits of $\boldsymbol{w}_1 = \boldsymbol{A}\boldsymbol{y}$ together with the message M are hashed to the ball B_τ. The Ball B_τ is the set of polynomials with exactly τ coefficients being -1 or 1 and the rest zero. The output of the hash function is the commitment value (commitment polynomial) c. Finally the vector $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s_1}$ is calculated. If $\boldsymbol{z}$ passes a tests which ensures that no information about the secret polynomials is leaked, the signature defined as the pair $\sigma = (\boldsymbol{z}, c)$ is outputted. Otherwise the process will be repeated until a generated signature passes the tests and can be outputted.

- **Verification** To verify a signature σ of a message M according to a public key *pk* the high bits of $\boldsymbol{w}_1 = \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$ are calculated, we name them $\boldsymbol{w}'_1$. Then the signature is accepted iff $\|\boldsymbol{z}\|_\infty \leq \gamma_1 - \beta$ and $c = H(M\|\boldsymbol{w}'_1)$.

This simplified version of the Dilithium signature scheme is also depicted in Algorithm 1.

To see why the scheme is correct the interesting part is the second condition which is checked. The question is whether following holds:

$$c = H(M\|\boldsymbol{w}'_1) = H(M\|\text{HighBits}(\boldsymbol{A}\boldsymbol{y})) \stackrel{?}{=} H(M\|\text{HighBits}(\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t})) \tag{2.1}$$

We know that

$$\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \boldsymbol{A}\boldsymbol{y} + c\boldsymbol{A}\boldsymbol{s}_1 - c\boldsymbol{A}\boldsymbol{s}_1 - c\boldsymbol{s}_2 = \boldsymbol{A}\boldsymbol{y} - c\boldsymbol{s}_2 \tag{2.2}$$

Per definition of the c and $\boldsymbol{s}_2$ the coefficients of $c\boldsymbol{s}_2$ can be at most β. Because β is comparatively small this addition / subtraction does not affect the high bits of $\boldsymbol{A}\boldsymbol{y}$. Thus the equation (2.1) holds and the signature scheme is correct.

The security of the scheme is based on the hardness of the Module-LWE problem as well as the SIS problem.

The actual scheme contains many improvements to decrease the memory footprint, decrease execution time and the amount of entropy required but these details are not required for understanding our fault attack nor do they affect our attack.

Function `Gen()` **is**
  $\boldsymbol{A} \leftarrow R_q^{k \times l}$;
  $(\boldsymbol{s}_1, \boldsymbol{s}_2) \leftarrow S_\eta^l \times S_\eta^k$;
  $\boldsymbol{t} \leftarrow \boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2$;
  return $(pk = (\boldsymbol{A}, \boldsymbol{t}), sk = (\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{s}_1, \boldsymbol{s}_2))$;
end
Function `Sign`$(sk = (\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{s}_1, \boldsymbol{s}_2), M)$ **is**
  $\boldsymbol{z} \leftarrow \perp$;
  while $\boldsymbol{z} = \perp$ **do**
    $\boldsymbol{y} \leftarrow \widetilde{S}_{\gamma_1}^l$;
    $\boldsymbol{w}_1 \leftarrow$ `HighBits`$(\boldsymbol{A}\boldsymbol{y}, 2\gamma_2)$;
    $c \in B_\tau \leftarrow$ `H` $(M \| \boldsymbol{w}_1)$;
    $\boldsymbol{z} \leftarrow \boldsymbol{y} + c\boldsymbol{s}_1$;
    if $\|\boldsymbol{z}\|_\infty \geq \gamma_1 - \beta$ *or* `LowBits`$(\boldsymbol{A}\boldsymbol{y} - c\boldsymbol{s}_2) \geq \gamma_2 - \beta$ **then**
      $\boldsymbol{z} \leftarrow \perp$;
    end
  end
  return $\sigma = (\boldsymbol{z}, c)$
end
Function `Verify`$(pk = (\boldsymbol{A}, \boldsymbol{t}), M, \sigma = (\boldsymbol{z}, c))$ **is**
  $\boldsymbol{w}_1' \leftarrow$ `HighBits`$(\boldsymbol{A}\boldsymbol{z}, 2\gamma_2)$;
  if $\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta$ *and* $c =$ `H`$(M \| \boldsymbol{w}_1')$ **then**
    Accept
  else
    Reject
  end
end

Algorithm 1: Simplified Dilithium signature scheme.

2.4.2 qTESLA

The qTESLA signature scheme [Alk+20] was selected for the second round of the NIST post-quantum cryptography standardization project but was not selected for round three. When describing the scheme we will leave out various checks which are performed during key generation, signature generation and verification and also some implementation details. The checks include the rejection sampling but also various checks which aim to counteract fault attacks. None of these checks detect our fault attack because our attack produces perfectly valid signatures. For the reader interested in all details of this signature scheme we refer to [Alk+20].

Notation

qTESLA uses the ring, $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ with $n \in \{1024, 2048\}$ and q an odd prime. The set $\mathcal{R}_{[B]} \subseteq \mathcal{R}_q$ is defined as $\mathcal{R}_{[B]} = \{\sum_{i=0}^{n-1} f_i x^i \in \mathcal{R} | f_i \in [-B, B]\}$ and the set $\mathbb{H} \subseteq \mathcal{R}_q$ is defined as $\mathbb{H} = \{\sum_{i=0}^{n-1} f_i x^i \in \mathcal{R} | f_i \in \{-1, 0, 1\}, \sum_{i=0}^{n-1} f_i = h\}$, $h \in 25, 40$. The $[\cdot]_M$ operator rounds aways the last few bits of its argument, roughly speaking. G and H are hash functions. H applies the rounding operation $[\cdot]_M$ to any polynomials from $\mathcal{R}_q$ it receives as input.

Key generation

From an initial so called "pre-seed", a κ bit long uniform random bitstring, multiple different seeds like $\mathbf{seed}_y$ and $\mathbf{seed}_a$ are generated to be used to sample the following polynomials. First the public key polynomials $a_1, \ldots, a_k$ are sampled uniformly from $\mathcal{R}_q$ using $\mathbf{seed}_a$. Next the secret polynomial $s \in \mathcal{R}_q$ is sampled with coefficients distributed following the discrete Gaussian distribution with standard deviation σ. The error polynomials $e_1, \ldots, e_k \in \mathcal{R}_q$ are sampled just like s was sampled previously. The public key polynomials t_i are then calculated as $t_i = a_i s + e_i$ for $i \in 1, \ldots, k$. Finally the value g is crafted by hashing all public key polynomials t_i using H. The secret key is then the tuple $(s, e_i, \ldots, e_k, \mathbf{seed}_a, \mathbf{seed}_y, g)$ and the public key is $(t_1, \ldots, t_k, \mathbf{seed}_a)$.

Signature generation

When signing a message m, first a randomness r is collected. Based on this randomness the masking polynomial y is sampled uniformly from $\mathcal{R}[B]$. Next the

polynomials $v_i = a_i y$ are calculated for $i \in \{1, \ldots, k\}$. These polynomials are then hashed together with $G(m)$ and g. This hash is then used to construct the commitment polynomial $c \in \mathbb{H}$. Finally the $z = y + sc$ is calculated and the signature is defined as the pair (z, c).

Signature verification

Given the public key, private key, a signature (z, c) and the message m, to verify a signature we first calculate $w_i \leftarrow a_i z - t_i c$. The signature is valid iff $z \in \mathcal{R}_{[B-S]}$ and $c = H(w_1, \ldots, w_k, G(m), G(t_1, \ldots, t_k))$. The second condition holds for valid signatures because [Alk+20, p. 445]:

$$[w_i]_M = [a_i z - t_i c]_M = [a_i(y + sc) - (a_i s + e_i)c]_M = [a_i y - e_i c]_M = [a_i y]_M \quad (2.3)$$

Note that the last equations holds because $e_i c$ only has small coefficients, thus only affecting the lower bits of $a_i y$, and $[\cdot]_M$ rounds away these lower bits.

2.4.3 BLISS

BLISS [Duc+13b] is an acronym for "Bimodal Lattice Signature Scheme". It introduced a new more efficient way of rejection sampling by using a bimodal Gaussian distribution instead of a Gaussian one. We will skip some technical details parts of the signature scheme they are not relevant for our attack.

Key generation

First the two polynomials f and g are sampled with $\lfloor \delta_1 n \rfloor$ coefficients uniformly from $\{-1, 1\}$ and $\lfloor \delta_2 n \rfloor$ coefficients uniformly from $\{-2, 2\}$, rest zero. The private key $\boldsymbol{S}$ is now defined as $\boldsymbol{S} = (s_1, s_2)^t = (f, 2g + 1)^t$. For the public key we first calculate $a_q = (2g + 1)/f \bmod q$ and set $\boldsymbol{A} = (2a_q, q - 2) \bmod 2q$.

If a check (which we did not mention here) on S does not pass or f is not invertible the key generation restarts until it succeeds. Once it succeeded, due to the way they keys were constructed, it holds that $\boldsymbol{AS} = q \mod 2q$.

Signature generation

The signature generation begins by sampling two masking polynomials y_1 and y_2 with coefficients from the the discrete Gaussian distribution D_σ. Then we calculate $u = \zeta a_1 y_1 + y_2 \bmod 2q$ and the commitment polynomial $c = H(\lfloor u \rceil_d \bmod p, \mu)$. Here H is a cryptographically secure hash function mapping to polynomials with exactly κ coefficients in $\{-1, 1\}$and the rest zero. Finally z_1 and z_2 are calculated by first sampling $b \in \{0, 1\}$ and then calculating $z_1 = y_1 + (-1)^b s_1 c$ and z_2 analogously: $z_2 \leftarrow y_2 + (-1)^b s_2 c$. Now the rejection step is performed. Finally let $z_2^\dagger = (\lfloor u \rceil_d - \lfloor u - z_2 \rceil_d) \bmod p$. The signature is the triple $(z_1, z_2^\dagger, c)$.

Signature verification

To verify a signature $(z_1, z_2^\dagger, c)$ given a public key $\boldsymbol{A} = (a_1, q - 2)$ the verifier first checks whether $\|(z_1 | 2^d \cdot z_2{}^\dagger)\|_2 > B_2$ or $\|(z_1 | 2^d \cdot z_2{}^\dagger)\|_\infty > B_\infty$ holds. If so he rejects the signature. Finally he accepts the signature iff $c = H(\lfloor \zeta a_1 z_1 + \zeta q c \rceil_d + z_2^\dagger \bmod p, \mu)$.

Understanding why the signature verification is correct is not straight-forward for BLISS. Because this aspect is irrelevant for the understanding of our attack and for the sake of brevity we will refer the interested reader to the original publication [Duc+13a] for details.

2.5 Fault attacks

Fault attacks describe an attack model where the attacker is able to choose the physical environment of a device under attack (DUA). The attacker tries to cause the device to malfunction and thus to output the result of a faulty calculation. When the device is running a cryptographic algorithm like an encryption algorithm or signature algorithm a faulty output like a faulty ciphertext or a faulty signature may reveal information about the internal state of the device which may include information of the secret key.

The idea of a fault attack was first introduced in [BDL97] which is now commonly known as the bellcore attack. The bellcore attack is able to break RSA-CRT with a single faulty signature. DES and AES can be broken with two faulty ciphertexts [KQ07, p. 544]. This shows how powerful the fault attack model is.

Faults can be achieved in different ways. In a voltage-glitch attack the supply voltage of the DUA is increased or decreased for a short amount of time to a range the device is not rated for, thus causing unexpected, faulty behaviour. A clock-glitch describes an attack, where the clock signal is disturbed. An attacker could for a single clock cycle significantly reduce the time the clock signal is high. This may result in the processor to skip an instruction. If the skipped instruction would have prevented a loop from aborting, this could be called loop-abort attack. This type of fault attack has already been shown to work on lattice-based signature schemes by Espitau et al. [Esp+18, pp. 1545–1546]. Shining light on the DUA can also cause a fault. The light can be a flash or laser. Because the light needs to be applied directly to the silicone this attack is semi-invasive as the plastic layer has to be removed [GT04, pp. 161–162]. Fault attacks are also possible by creating an electromagnetic field close to the circuit. Such attacks work even without opening the package of the IC [Deh+12]. Further types of attacks exists, see [KQ07] for a survey.

For many of the attacks countermeasures do exists and are used in practice. Countermeasures can either be implemented in hardware (e.g. detect high / low voltage) or in software (e.g. check if a previous calculation seems plausible). Because this attack model gives the attacker seemingly endless possibilities to attack it is very hard to defend against.

3 Related Work

Since many lattice based signature schemes are based on worst-case to average-case reductions [Ajt96] they are almost provably secure. Due to this and because the NIST explicitly encouraged to focus more research on implementation security [ST16; Moo+20], the implementation security of lattice-based signature schemes received more scrutiny recently.

Using the same loop-abort attack scenario we will use in our thesis Espitau et al. presented attacks on Fiat-Shamir type lattice-based signatures [Esp+17], where the masking polynomial is faulted to have abnormal low degree. Their attack is presented in detail in section 4.1. Furthermore they presented a similar attack approach on Hash-and-Sign type lattice based signatures. Their attacks and proposed countermeasures inspired our work.

Additionally they presented fault attacks against two PQC key-exchange protocols: NewHope as well as Frodo [Esp+18]. In both schemes they aborted the loop which sampled a commitment polynomial. While it would be interesting to see if our attack can also break the shuffling countermeasure for these two schemes, this is out of scope for this thesis.

Bruinderink and Pessel did a thorough analysis on differential fault attacks [GP18]. This work included a partial nonce-reuse attack where similar to our attack scenario some of the coefficients in the masking vector y are being faulted. Despite this similarity their attack scenario was more broad as that the faulted entries in y are assumed to be random in contrast to our more constraint assumption that the faulted values are constant. Additionally they always require a faulted and non-faulted signature pair for a fixed message whereas we simply require faulted signature of arbitrary messages.

Bindel et al. discussed a zeroing attack in which some entries of the masking vector are zeroed [BBK16, pp. 72–73]. In contrast to Espitau et al. [Esp+17] they did not assume the zeroed coefficients to be the upper ones of the polynomial, but they still assumed that the zeroed coefficients are located right next to each other and that the coefficients are not shuffled after sampling. While not discussed by the authors in detail, this allows for a better statistical analysis with lower false-positive rate to

classify zeroed vs. non-zeroed coefficients. Thus the authors did just assume that the classification of zero coefficients is perfect. Our work gives a possible solution for an attacker which does not classify perfectly.

4 Attacking Fiat-Shamir with aborts over rings based signature schemes

In this chapter we will describe our attack. First we introduce a previous attack from Espitau et al. [Esp+17] in section 4.1. Then we will first describe the attack in general for lattice-based Fiat-Shamir with aborts over rings signature schemes in section 4.2. Finally we will describe the technical details to deploy the attack on Dilithium, qTESLA and BLISS in sections 4.3, 4.4 and 4.5 respectively.

4.1 Previous attack by Espitau et al.

Our attack is inspired by the fault attack and proposed countermeasures of Espitau et al. [Esp+17]. They described their attack, among other schemes, against BLISS. They showed that a single fault during the signing process can reveal the secret key. In the following section we will describe their attack against BLISS.

4.1.1 Attack idea

The fault model the authors used was a loop-abort fault. A loop-abort fault is a fault intentionally induced by an attacker to prematurely end a loop, i.e. prevent a certain amount of iterations to be executed. The loop they targeted was the one sampling the coefficients of the masking polynomial y_1. Such a fault yields a masking polynomial with abnormal low degree. The degree is denoted by m. A BLISS signature is the triple $(c, z_1 = s_1 c + y_1, z_2^{\dagger})$. A faulty signature with a faulty y_1 is all the information needed for performing this attack.

The important observation is that if we assume that c is invertible, which is true with a high probability, then the vector $z_1 c^{-1}$ is close to a sublattice of $\mathbb{Z}^n$. To be more precise we can see that

$$z_1 c^{-1} - s_1 = c^{-1} y_1 = \sum_{i=0}^{m-1} y_{1,i} c^{-1} x^i \mod q$$

the sublattice Λ is spanned by the vectors $w_i = y_{1,i} c^{-1} x^i$ for $i = 0, \ldots m-1$ as well as the vectors $q\mathbb{Z}^n$. The difference between $z_1 c^{-1}$ and that lattice is exactly s_1.

Because the dimension of that sublattice is still n, the secret cannot be recovered directly. But if we project the point $z_1 c^{-1}$ as well as the basis vectors of the sublattice Λ to a subset of its rows, it still holds that the difference is the projected s_1. If the projection is chosen such that the degree is low enough, we can recover a part of s_1. By choosing multiple such projections we can eventually recover the entire secret polynomial and thus that the entire secret key.

Additionally the authors mention that the fault can also be of other nature. The attacker could for example target the memory where (part of) the masking vector is stored and fault it to zero [Esp+17, pp. 153–154].

4.1.2 Countermeasures

One of the proposed countermeasures by Espitau et al. is to generate the coefficients of y_1 in a random order [Esp+17, p. 13]. This prevents the attack as they do not know which of the $m < n$ vectors span the sublattice of interest, because they do not know which coefficients of y_1 are zero or not. This countermeasure can easily be added to an implementation because it is simple and fast. Using the Fisher-Yates technique [FY48] the shuffling can be done in n iterations, i.e. linear complexity.

Furthermore they propose to add a check after sampling the coefficients of y_1. The idea is to check whether y_1 has too low degree. This would prevent their attack as they require m to be at most around 100 [Esp+17, p. 151]. The authors claim that if the signature process is aborted if more than $1/16$'th of the upper coefficients of y_1 are zero, the distribution of y_1 is skewed so little that it is statistically indistinguishable from the original one and thus the security proof of BLISS still holds [Esp+17, p. 155].

4.2 General attack scheme

In this section we will describe our attack mostly signature scheme agnostic- First we will discuss our assumptions and important properties of the signatures in subsection 4.2.1, then we will describe our ILP in subsection 4.2.2 and finally we will describe an optimization in subsecton 4.2.3.

4.2.1 Preliminaries

This subsection will define our assumptions and notation and highlight two important properties of the signatures. These properties apply to all the three here discussed signatures.

Assumptions and notation

The Fiat-Shamir with aborts signature schemes produce signatures in the form of $z = cs + y$ where z is (part of) the signature (public), s is (part of) the secret key (secret), c is the commitment value (public) and y is the masking vector (secret, (deterministically) random).

Here we will assume that the ring is $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ with n a power of two and q an odd prime. Thus z, s and y are all elements in $\mathcal{R}_q$.

We assume an attacker who is able produce σ signatures and in every signature he induces a loop-abort fault in the loop which samples the masking polynomial. We assume that uninitialized y-coefficients are zero. Thus the faulted signature will have at least $1 \leq f \leq n$ zero coefficients in y resulting in at most $m = n - f$ non-zero coefficients. As afterwards the coefficients are shuffled, he does not know where the zero coefficients are located.

While we here assume that a loop-abort fault will cause coefficients of y to be zero, any kind of fault is theoretically possible, e.g. a fault in memory as already mentioned in section 4.1.1. Additionally, the faulted coefficients do not necessarily need to be zero, any other constant can works just as efficiently, though small modifications to the attack are needed.

No modular reductions

The commitment polynomials are sparse polynomials with coefficients which have small absolute values. While not sparse, the secret polynomials s too have only coefficients with very small absolute value. Thus their product is below the modulus q even when adding the masking polynomial y. Thus during the computation of z no modular reductions are performed or in other words, the operations can be seen as if they are performed in the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$

Linearity

To understand why the signature is linear we need to understand how we can describe the multiplication of the commitment polynomial c and the secret polynomial s in the Ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ as a linear operation.

We can describe the multiplication as a matrix vector multiplication: The matrix is a quadratic $n \times n$ matrix C, which contains all n negacyclic rotations (rotate, then negate) of the coefficient vector c' of the commitment polynomial. The vector multiplied with C is the coefficient vector s' of the secret polynomial s [PBY17, p. 1845]. Thus Cs' is the coefficient vector of cs and together with the coefficient vector y' of the masking polynomial y we can write the entire signature as a linear equation system $z' = Cs' + y'$.

4.2.2 The attack

Let $M = \sigma n$. Once we have gathered the faulted signatures we will use these signatures to construct $\dot{z} \in \mathbb{Z}_q^{\sigma n}$ as the vector containing all σ z'-vectors vertically stacked, $\dot{C} \in \mathbb{Z}_q^{\sigma n \times n}$ as the matrix containing all σ C-matrices vertically stacked as well as $\dot{y} \in \mathbb{Z}_q^{\sigma n}$ as the vector containing all σ y'-vectors vertically stacked.

Now we can look at the bigger equation system $\dot{C}s' + \dot{y} = \dot{z}$. We can not directly solve it because we do not know $\dot{y}$. Though we do know that $\dot{y}$ has a lot of zero entries, at least compared to a non-faulted version of this vector. The way we will look at this problem now is as follows: We will take the equation system $\dot{C}s' = \dot{z}$ and say that it is "noisy", i.e. that it has some rows $1 \leq i \leq M$ which are not "true" by which we mean that $\dot{y}_i \neq 0$.

In the following section we will refer to equations $\dot{C}_i s' = \dot{z}_i$ for which $\dot{y}_i = 0$ holds as "faulted equations" or "true equations" whereas for equations $\dot{C}_i s' = \dot{z}_i$ for which $\dot{y}_i \neq 0$ holds we will refer to as "false equations" or "non-faulted equations".

To solve our the noisy linear equation system we will construct an instance of Marzougui's and Ulitzsch's [Mar+22, pp. 46-47] ILP, which can classify true and false equations from one another, and solve the linear equation system for s'.

Variables

The ILP will use two classes of variables: n variables in the same range as the coefficients of the secret polynomial s. We refer to them as the vector $\hat{s} \in \mathbb{Z}_q^n$, Furthermore we will use M boolean variables, We will refer to these variables as the vector $x \in \{0,1\}^M$.

Constraints and objectve

Let $K = \max_{z'_i, C_i, s'} |z'_i - C_i s'|$. The ILP will aim to maximize $\sum_{i=1}^{M} x_i$ while fulfillng the two constraints

$$\dot{z} - \dot{C}\hat{s} \leq K(1-x) \tag{4.1}$$

$$\dot{z} - \dot{C}\hat{s} \geq -K(1-x) \tag{4.2}$$

in addition to the constraints for the domain of the vectors $\hat{s}$ and x.

The two constraints can also be written in a more compact way:

$$-K(1-x) \leq \dot{z} - \dot{C}\hat{s} \leq K(1-x) \tag{4.3}$$

Let s_{min} and s_{max} be the lower and upper bound for the secret polynomial coefficients respecivley. Then the ILP with all its constraints can be formally defined as follows:

$$\text{maximize} \quad \sum_{l=1}^{M} x_l \tag{4.4}$$

$$\text{subject to} \tag{4.5}$$

$$\dot{z}_l - \dot{C}_l \hat{s} \leq K(1 - x_l) \quad \forall l \in \{1, \ldots, M\} \tag{4.6}$$

$$\dot{z}_l - \dot{C}_l \hat{s} \geq -K(1 - x_l) \quad \forall l \in \{1, \ldots, M\} \tag{4.7}$$

$$x_l \in \{0, 1\} \quad \forall l \in \{1, \ldots, M\} \tag{4.8}$$

$$\hat{s}_i \in \mathbb{Z} \quad \forall i \in \{1, \ldots, n\} \tag{4.9}$$

$$\hat{s}_i \leq s_{\text{max}} \quad \forall i \in \{1, \ldots, n\} \tag{4.10}$$

$$\hat{s}_i \geq s_{\text{min}} \quad \forall i \in \{1, \ldots, n\} \tag{4.11}$$

Reasoning

The idea behind this ILP is that the objective will be maximized iff $\hat{s} = s'$ and the x-vector classifies the $\hat{y}$ coefficients correctly, i.e. for all $i \in \{1, \ldots, M\}$ x_i will be 1 iff $\dot{y}_i = 0$ or in other words iff the $(i \bmod n)$'th y-coefficient of the $\lfloor i/n \rfloor$'th signature is zero.

To see why these two constraints do indeed help us to solve our problem we will look at what happens if the ILP-solver sets $x_i = 0$ or $x_i = 1$ for some $i \in \{1, \ldots, M\}$.

If $x_i = 0$, the compact constraint (4.3) will simplify to following.

$$-K(1 - x_i) \leq \dot{z}_i - \dot{C}_i \hat{s} \leq K(1 - x_i) \tag{4.12}$$

$$\Leftrightarrow -K \leq \dot{z}_i - \dot{C}_i \hat{s} \leq K \tag{4.13}$$

This constraint is always true per our definition of K. In our use-case this means that when the ILP-solver sets $x_i = 0$ it effectively removes equation i from our set of linear equations. Note that the ILP-solver can always find a solution that will fulfill all constraints by setting $x = 0$, but this solution will not optimize the objective.

If $x_i = 1$ and $\hat{s} = s'$, the compact constraint (4.3) will simplify to following.

$$
\begin{aligned}
& -K(1-x_i) \leq \dot{z}_i - \dot{C}_i\hat{s} && \leq K(1-x_i) && (4.14)\\
& -K(1-1) \leq \dot{z}_i - \dot{C}_i\hat{s} && \leq K(1-1) && (4.15)\\
\Leftrightarrow & 0 \leq \dot{z}_i - \dot{C}_i\hat{s} && \leq 0 && (4.16)\\
\Leftrightarrow & 0 \leq (\dot{C}_i s' + y'_i) - \dot{C}_i\hat{s} && \leq 0 && (4.17)\\
\Leftrightarrow & 0 \leq \dot{y}_i && \leq 0 && (4.18)\\
\Leftrightarrow & 0 = \dot{y}_i && = 0 && (4.19)
\end{aligned}
$$

Thus, if the y'_i is zero, most likely due to a fault, this constraint will be fulfilled.

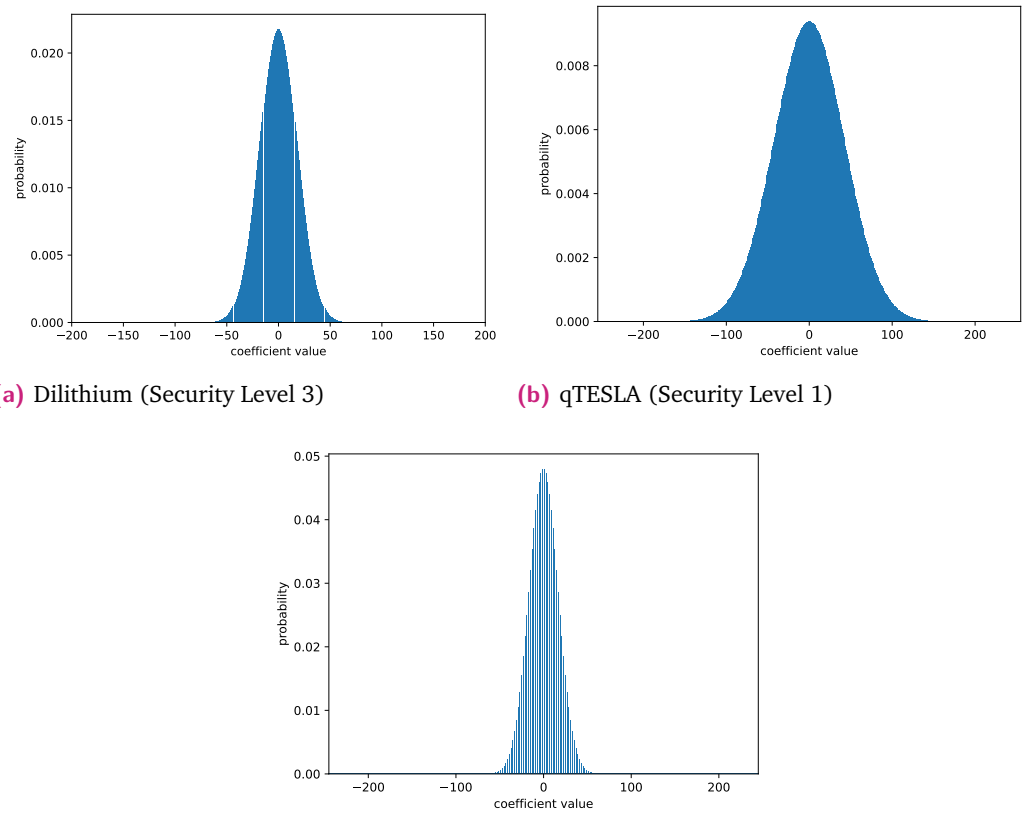
If $x_1 = 1$ but $\hat{s} \neq s'$ this constraint could also be fulfilled. We heuristically assume, supported by our successful simulations, that such events are very unlikely to maximize the objective and will this not be chosen by the ILP-solver.

4.2.3 Pre-filtering equations without faults

The heavy lifting of classifying whether a coefficient of y is zero or not is done by the ILP-solver. Nevertheless for a certain set of equations we can say with ease that the corresponding y-coefficient can not be zero. When a coefficient y_i is zero we know that $z_i = (cs + 0)_i$. For all signature schemes we can give a bound b to the maximum possible absolute value a coefficient of cs. Thus if $|z_i| > b$, we know that $y_i \neq 0$ with a probability of 1.

For improved performance we will thus remove any equations where we can tell, by using the aforementioned approach, that their corresponding y-coefficient is not zero. We do this before passing these equations to the ILP-solver. In practice this means removing the corresponding entry in the vector z' and the corresponding row in the matrix C.

To go even further we can use a lower threshold $t < b$ to pre-filter our equations. Because for all three signature schemes we know that the distribution of cs has most of its probability mass close to zero, is symmetrical and thus has very little probability mass near the bounds $-b$ and b, this will filter out even more equations which are not affected by a fault. On the contrary we will also filter out some equations which were indeed affected by the fault increasing the amount of signatures we need. For a better visualization of the distributions see Figure 4.1



(a) Dilithium (Security Level 3)

(b) qTESLA (Security Level 1)

(c) BLISS (BLISS-III)

Fig. 4.1: The distribution of a cs-coefficient (same as that of a faulted z-coefficient) for the different signature schemes. The plots for Dilithium and BLISS show the entire distributions. The plot for qTESLA shows the 6-σ-interval of the cs distribution.

The metric we control by pre-filtering the equations is the false-positive rate of our equation set. Here we classify an equation as being true, i.e. $y_i = 0$ or not. A true-positive classification would be if we classify an equation is faulted and it indeed is. A false-positive classification is when we classify an equation as faulted, although it actually is not. Thus the false-positive rate is ratio of the number false-positive classifications of our equations to the total number of equations. This number is an interesting metric for the efficiency of our attack as we will discuss later. We are able to control this number by two parameters:

1. the iteration number m, after which we induce a fault.
2. the cutoff value t we use for filtering

4.3 Attacking Dilithium

In this section we will discuss the Dilithium signature scheme specific details for our attack.

4.3.1 Implementation assumptions and attacker model

First we assume that a big loop with nl iterations will sample the coefficients for all l polynomials with n coefficients each in $\boldsymbol{y}$. Secondly we assume that the shuffling occurs throughout all the polynomials of the vector $\boldsymbol{y}$. This means that coefficients which where faulted to be zero located in the last polynomial may be shuffled to another polynomial, e.g. the first one.

We thus adjust our attacker model in the way that the attacker will induce a fault after the $1 \leq m \leq nl$ iteration in the loop which samples the coefficients of the masking polynomials.

4.3.2 One ILP per polynomial

The Dilithium signature scheme calculates the signature like follows:

$$\boldsymbol{z} = \boldsymbol{s}_1 c + \boldsymbol{y} \tag{4.20}$$

Where $\boldsymbol{z}, \boldsymbol{s}_1, \boldsymbol{y} \in \mathcal{R}_q^l$ and $c \in \mathcal{R}_q$. We thus can not directly apply the attack mentioned before. To still use our attack against Dilithium, let $\boldsymbol{z}_i, \boldsymbol{s}_{1i}, \boldsymbol{y}_i$ be the i'th polynomial

in the respective vectors for $1 \leq i \leq l$. According to definition of $\boldsymbol{z}$ it holds for all $1 \leq i \leq l$ that $\boldsymbol{z}_i = c\boldsymbol{s}_{1i} + \boldsymbol{y}_i$. Now, because $\boldsymbol{z}_i, c, \boldsymbol{s}_{1i}, \boldsymbol{y}_i \in \mathcal{R}_q$ we can apply our attack for every i to recover once secret polynomial of $\boldsymbol{s}_1$ at a time, eventually recovering the entire vector $\boldsymbol{s}_1$.

4.3.3 Bound for the difference $z'_i - C_i s'$ and threshold t

The coefficients of the commitment polynomial c have at most an absolute value of 1 which can occur at most τ times. The coefficients of the $\boldsymbol{s}_1$ polynomials have at most an absolute value of η. Thus we know that $|C_i s'| \leq \tau\eta = \beta$. Finally the coefficients of the $\boldsymbol{y}$ masking polynomials have an absolute value of at most γ_1, thus $|z'_i|$ is at most $\tau\eta + \gamma_1 = \beta + \gamma_1$. The absolute value of the difference is consequently $K = 2\beta + \gamma_1$.

To filter equations we will use a threshold of $t = \beta$ as this is the upper bound for coefficients in $\boldsymbol{s}_1 c$.

4.3.4 Creating signatures using only $\boldsymbol{s}_1$

In the Dilithium signature scheme $\boldsymbol{s}_1$ is only one part of the secret key. Our attack can not recover the second part of the secret key, $\boldsymbol{s}_2$, still $\boldsymbol{s}_1$ is enough information for an attacker to sign arbitrary messages. To sign arbitrary messages two different methods were previously presented. One by Bruinderink and Pessel [GP18, pp. 33–34] and one by Ravi et al. [Rav+18, pp. 12–13]. These signatures are indistinguishable from real ones when only given the public key. Only given the secret key we can distinguish the signatures created using only $\boldsymbol{s}_1$ from the ones created by someone with the entire secret key.

4.4 Attacking qTESLA

Attacking qTESLA is straight forward because the qTESLA signature scheme strictly follows our general attack approach we presented in section 4.2.2. Thus we only have to consider the maximum difference $z'_i - C_i s'$ and the appropriate threshold t.

4.4.1 Bound for the difference $z'_i - C_i s'$ and threshold t

The polynomial s has coefficients which follow the discrete gaussian distribution with standard deviation σ. This distribution is a long-tailed one and it can have theoretically infinitely large samples (coefficients). In practice discrete gaussian samplers have a parameter τ which defines a so called cutoff-value $\tau\sigma$, which limits the samples to be in the domain $[-\tau\sigma, \tau\sigma] \cap \mathbb{Z}$. For our evaluation we choose $\tau = 6$ because this is the default one used in SageMath [@Sag22]. Thus we can limit the domain of s to be $[-\tau\sigma, \tau\sigma] \cap \mathbb{Z}$.

The commitment value c has exactly h non-zero coefficients which are -1 or 1 and the rest is zero. Thus we know that the polynomial cs has coefficients in the range $[-h\tau\sigma, h\tau\sigma] \cap \mathbb{Z}$.

The masking polynomial y has per definition coefficients in the range $[-B, B] \cap \mathbb{Z}$. Thus we can say that the coefficients of $z = y + cs$ are in the interval $[-Bh\tau\sigma, Bh\tau\sigma] \cap \mathbb{Z}$. Now looking at the difference $z'_i - C_i s'$ we know that that it's maximum absolute value is bounded by $2h\tau\sigma + B$. Thus we choose $K = 2h\tau\sigma + B$ and $t = h\tau\sigma$.

4.5 Attacking BLISS

The BLISS [Duc+13b] signature scheme was first introduced by Léo Ducas et al. in 2013. With their novel rejection sampling algorithm and other modifications they were able to significantly reduce the standard deviation of their signatures and thus the signature size compared to other post-quantum secure schemes at that time. Their new rejection sampling algorithm is based on the bimodal gaussian distribution. This distribution is obtained in their scheme by choosing a random bit $b \in \{0, 1\}$ and computing a signature vector $z_1 = y_1 + (-1)^b s_1 c$. The entire signature algorithm can be found in section 2.4.3.

Note that the previous two signature schemes do not use such a bit in their signature generation algorithms. To adapt our attack to BLISS we will have to include this bit of information in the ILP.

4.5.1 A modified ILP for BLISS

Due to the random bit we now have three types of unknowns: The coefficients of the secret key polynomial s_1, the coefficients of the masking polynomial y_1 and the bit b.

Thus the ILP will use three classes of variables: In addition to x and $\hat{s}$ we already described in the general ILP, we will have the binary variables $b_1, \ldots, b_\sigma \in \{0, 1\}$ to represent the random bit b.

Next will will define for all $1 \leq i \leq \sigma$ the vector $\vec{b_i} \in \{-1, 1\}^n$ as follows:

$$\vec{b_i} = (\overbrace{2(1-b_i)-1, 2(1-b_i)-1, \ldots, 2(1-b_i)-1}^{n \text{ times}})^t \tag{4.21}$$

This allows us to create the vector $\dot{b}$ which we define as all $\vec{b_i}$ vectors vertically stacked on one another, just like the definition of e.g. $\dot{z}$.

Finally we can redefine the two constraints (4.1) and (4.2) of the ILP as follows,

$$\dot{z} - \dot{b} \cdot C\hat{s} \leq K(1-x) \tag{4.22}$$
$$\dot{z} - \dot{b} \cdot C\hat{s} \geq -K(1-x) \tag{4.23}$$

where the $\cdot$ (dot) between $\dot{b}$ and $C\hat{s}$ denotes the entry-wise multiplication of the two vectors.

4.5.2 Bound for the difference $z'_i - C_i s'$ and threshold t

Let $d_1 = \lfloor n\delta_1 \rfloor$ be the amount of ± 1 coefficients and $d_2 = \lfloor n\delta_2 \rfloor$ the amount of ± 2 coefficients in s_1. Note that c has exactly κ coefficients with at most an absolute value of 1 and the rest zero. Thus we maximize $C_i s'$ if all ± 2-coefficient of s_1 match with non-zero coefficients of C_i and the remaining $\kappa - d_2$ non-zero coefficients of C_i match with the ± 1-coefficients in s_1. As for all parameter sets $\kappa > d_2$ holds, the bound for the absolute value of $C_i s'$ is $2d_2 + \kappa - d_2 = d_2 + k$, which we will use as the threshold.

The coefficients of y_1 follow the discrete gaussian distribution with a standard deviation σ. As already discussed in detail in section 4.4.1 we can limit the absolute values of the coefficients of y_1 to $\tau\sigma$ with $\tau = 6$. Thus we know that $|z'_i| \leq d_2 + \kappa + \tau\sigma$. Thus we choose $K = 2(d_2 + \kappa) + \tau\sigma$.

4.5.3 Recovering the entire secret key

This attack will only recover the secret polynomial s_1 but not the second secret polynomial s_2. This is not a problem as s_1 is sufficient to recover the entire secret key: $s_2 = a_1 s_1/2 \bmod q$. This holds because:

$$a_1 s_1/2 = ((2g+1)/f)s_1 = ((2g+1)/f)f = (2g+1)f/f = 2g+1 = s_2 \mod q \tag{4.24}$$

5 Attack evaluation

In this chapter we will quantify the effectiveness of our attack. We will begin by explaining our assumptions and evaluation strategy in section 5.1. In section 5.2 we will then present our results for Dilithium, qTESLA and BLISS respectively.

5.1 Preliminaries

In this section we will discuss on what assumptions we base our simulations and what kind of parameters we will evaluate.

5.1.1 Assumptions

We assume an attacker as described in section 4.2.1. Further will always assume that we as the attacker will have at least as much information as would be sufficient to break the signature scheme if the shuffling countermeasure would not be used. This translates to the fact that we will always have n or more faulted coefficients which correspond to at least n linear independent equations. This is because without the shuffling countermeasure an attacker would know which coefficients were faulted (the last few), take the corresponding linear equations and solve them using gaussian elimination.

5.1.2 Notion of success

For a certain m we will say that we were successful in breaking the signature scheme, if we manage to succeed recovering the entire secret in at least two out of three tries, i.e. we have a success rate of at least around 66%. Each try fails after a timeout of 5 minutes wall time.

5.1.3 Parameter evaluation

When we use only the minimal mount of signatures required to fulfill our aforementioned requirements the attack will not show optimal results. Thus, for Dilithium, we will evaluate two parameters of our attack which we hope to improve our results:

1. surplus of equations p
2. threshold t

For qTESLA and BLISS we will only evaluate the parameter p.

The surplus of equations p is the minimal amount of additional faulted equations, relative to the minimum described in 5.1.2. In our simulations we control this value by generating signatures until we can guarantee that we have has at least $n + p$ faulted equations out of which where exist n linear independent ones. Note that using this approach the resulting amount of faulted equations may be higher than the surplus of equations parameter, because a single signature might produce more than one faulted signature.

The threshold parameter is used as to steer the false-positive rate as described in detail in section 4.2.3. The lower the threshold, the lower the false-positive rate and vice versa. The false-positive rate is dependent on both, m and t. Thus for a fixed m the false-positive rate only depends on t.

We choose the two parameters based on the following two arguments:

We argue that a greater surplus of equations opens more choices to the ILP-solver to classify enough equations properly and thus the searchspace will more likely contain an easy-to-find path to a solution. The arguing with the false-positive rate is that a lower false-positive rate makes (random) positive classification guesses of the ILP-solver more likely to be correct and thus the ILP-solver should reach its objective quicker, eventually recovering the secret polynomial. Additionally a lower false-positive rate reduces the total amount of equations required to have at least n true equations. This reduces the amount of total variables and thus the difficulty of the generated ILP instance.

We tested our hypothesis statistically by running attack simulations for Dilithium (NIST security level II) for $f = 1, \dots 10$. Recall that f denotes the amount of faulted coefficients. We do not choose higher f because we empirically know our attack almost always succeeds in that case.

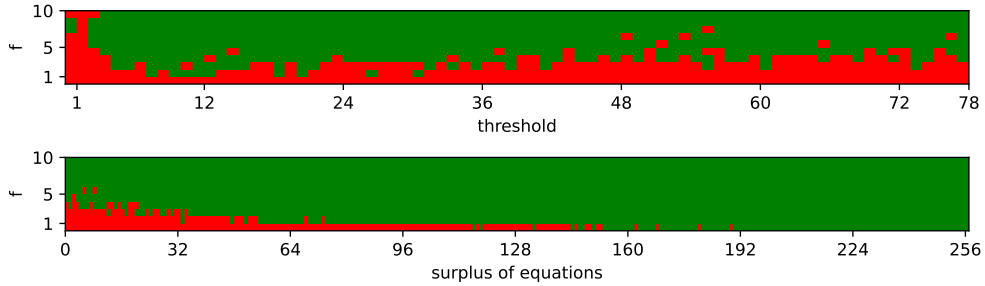
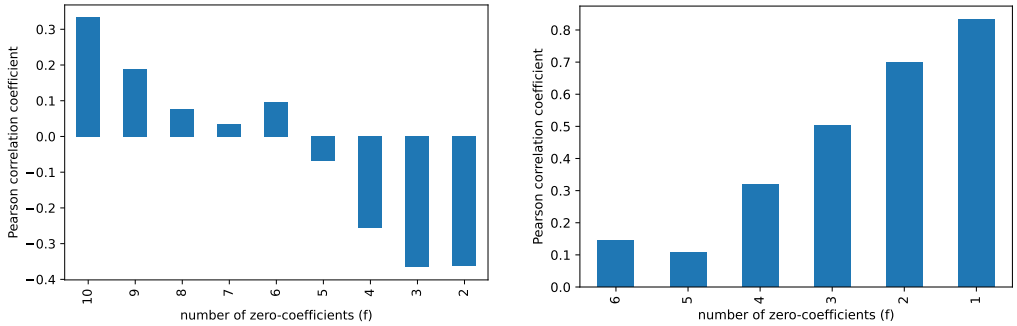


Fig. 5.1: Heatmap showing the correlation between threshold and success as well as the correlation between surplus and success for $f = 1, \ldots, 10$. A red rectangle indicated a failed attack, a green rectangle indicated a successful attack.



(a) threshold t

(b) surplus of equations s.

Fig. 5.2: The minimal required surplus of equations and the corresponding signature count per f. If a bar is not present it means that no surplus was required for that security level and f.

The heatmap 5.1 clearly shows that the surplus parameter has an effect on the attack performance which is also confirmed by the Pearson correlation coefficients in Figure 5.2. For the threshold parameter on the other hand this is not directly clear. Our data shows that the attack performance is poor when extremely low thresholds are used. We think that an extremely low threshold affects a parameter other than the false-positive rate and this to us unknown parameter has a negative impact on the attack performance. This phenomena can also be seen in Figure 5.2a: For $f \geq 6$, we see a positive correlation is caused by the poor attack performance with for extremely low thresholds. For smaller f we see a negative correlation, due to the low false-positive rate. While the absolute value of the correlation coefficient for $f \leq 5$ is not very big, taking into consideration that it also has to compensate for the poor performance when the threshold is extremely low, the positive effect of the threshold to the attack performance is not neglectable.

When we perform an attack with default parameters, we mean we will use a threshold which does not filter any true-positive equations, we fulfill the assumptions

defined in section 5.1.2 and all that with the minimal amount of signatures required.

5.1.4 Parameter evaluation strategy

In general the simulations aim to find out how lax we can be in the timing of the fault, which translate to how high we can go with the iteration number m, after which we will induce a fault. While in the simulations we will fault exactly f coefficients per signature, the simlation results can be translated to an attacker who faults on average f coefficients per signature.

```
1  t ← β // threshold
2  f ← nl // number of zero-coefficients; m = nl − f
3  while f ≥ 1 and t ≥ 1 do
4  |  if at least 2 out of 3 attacks successful then
5  |  |  print(for f zero-coefficients we need a threshold of no more than t);
6  |  |  f ← f − 1;
7  |  else
8  |  |  t ← t − 1;
9  |  end
10 end
```

Algorithm 2: Evaluation strategy for the s parameter.

```
1  s ← n // surplus
2  f ← nl // number of zero-coefficients; m = nl − f
3  while f ≤ 1 and s ≤ 2n do
4  |  if at least 2 out of 3 attacks successful then
5  |  |  print(for f zero-coefficients we need a surplus of at least s);
6  |  |  f ← f − 1;
7  |  else
8  |  |  s ← s + 1;
9  |  end
10 end
```

Algorithm 3: Evaluation strategy for the t parameter.

The evaluation strategy for the parameter “surplus of equations” and “threshold” are depicted in Algorithm 2 and 3 respectively. Using these strategies we will be able to determine for every m if our attack succeeds, and if yes, what parameter value is necessary for success.

5.1.5 Simulation hard- and software

The ILP-solver used for our simulations is Gurobi [1]. Our simulation is written in Python using the Numpy library [2] and the gurobipy library [3] for the Gurobi Python bindings. All simulations were run on a computer with a Intel® Xeon® Processor E7-4870 (4 sockets, 10 cores, using 40 out of 80 available threads @ 2.4GHz) with 500GB of RAM.

5.2 Simulation results

In this section we will discuss the results of our simulations. First we will start with a thorough discussion on the results for Dilithium, as we will evaluate and compare both the parameters s and p. Next we will follow with qTESLA as well as BLISS. In the end we will discuss how the repetition rate affects the required amount of signatures.

5.2.1 Dilithium

For Dilithium we will first discuss the results when our default parameters are used. Then we will discuss the results for both, the parameter s and the parameter t. Finally we will compare the results of both parameters.

Default parameters

When using the default threshold described in section 4.2.3 and a surplus of 0 our attack succeeds for f as low as 5, 6 and 4 requiring 224, 236.5 and 491 signatures on average for the NIST security levels II, III and V respectively. Further we observe that the higher the NIST security level the more signatures are required. This is due to the fact that minimum amount of equations increases as the security level increases because the parameter l increases for every security level. Recall that the minimum amount of required equations is nl for Dilithium. The amount of signatures required per m is depicted in Figure 5.3.

[1] `https://www.gurobi.com/`
[2] `https://numpy.org/`
[3] `https://pypi.org/project/gurobipy/`

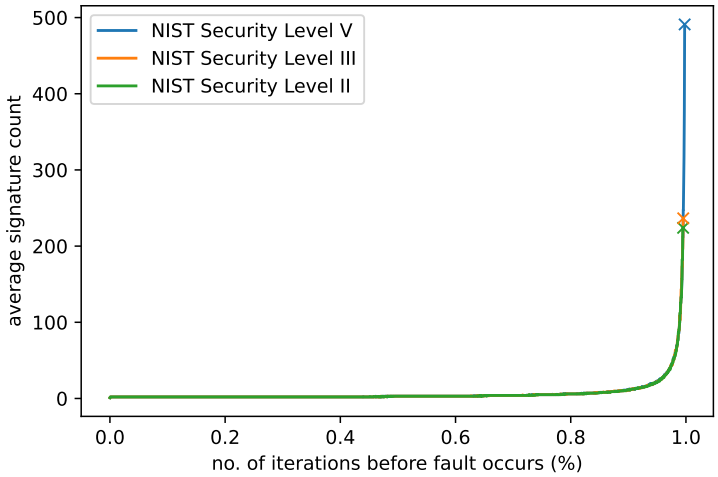
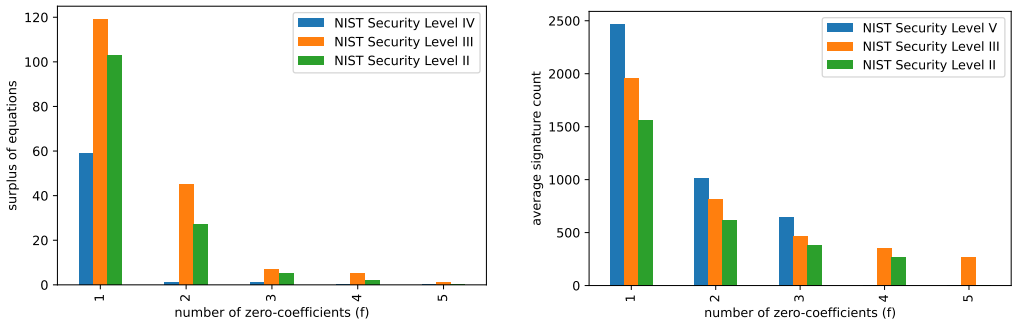


Fig. 5.3: Average amount of signature required for a successful attack per m in percent (relative to nl). The amount of signatures is very similar throughout all security levels for early iteration aborts. Thus the green line covers the other two. Crosses indicate the maximum signature count.

Surplus of equations

Our simulation results show, that for every m, we are able to succeed our attack. This means that an attacker who is able to (on average) fault a single coefficient of $\boldsymbol{y}$ can recover $\boldsymbol{s}_1$, given that he can do it on around 1562, 1959.5, 2465 signatures for NIST security level II, III and V respectively. The surplus of equations required per m as well as the amount of signatures required per m are depicted in Figure 5.4.

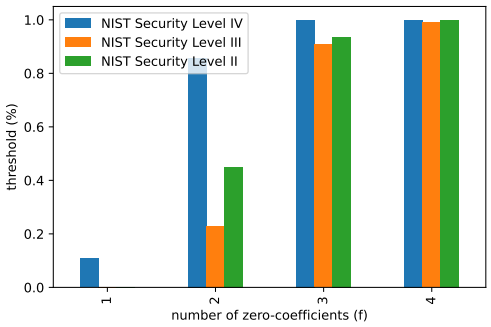
We can observe that the signature count increases as f decreases, as well as when the security level increases. First is due to the fact that we get less zero-coefficients



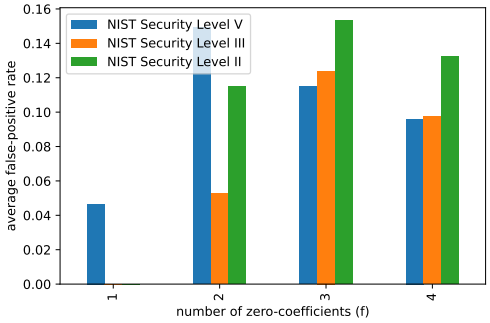
(a) Surplus of equations per f

(b) Corresponding signature count per f.

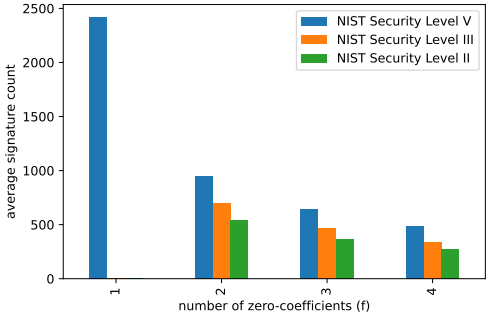
Fig. 5.4: The minimal required surplus of equations and the corresponding signature count per f. If a bar is not present it means that no surplus was required for that security level and f.



(a) Maximum possible threshold (%).



(b) Corresponding false-positive rate.



(c) Corresponding signature count.

Fig. 5.5: The maximum possible threshold per f and the corresponding false-positive rate as well as the corresponding signature count.

per signature and letter is the case because higher security levels require more zero-coefficients in total (nl many) as l increases as the security level does.

Threshold

We evaluated the threshold parameter for $f \leq 4$ as for higher f our attack succeeded without needing a reduced threshold. We were able to succeed our attack for all $f \geq 2$ for all NIST security levels. With a single coefficient faulted our attack succeeded only for security level V.

In Figure 5.5a we observe that as f decreases (m increases) we require a lower threshold to succeed. This can be explained because as f decreases for a fixed threshold we will have an increased false-positive rate. To achieve the same or lower false-positive rate as we had with $f + 1$ we require a lower threshold.

Comparing the threshold of different security levels for a fixed f is difficult because the distributions of $c\boldsymbol{s}_1$ differs for different the security levels. This kind comparison is better performed by looking at the false positive rate.

Inspecting the false-positive rate in Figure 5.5b for $f = 3, 4$ we observe that the higher the security level the lower the false-positive rate. This behavior is expected as l ILPs need to be solved per attack and l increases as the security level increases. As all security levels have the same time limitation, higher security levels need to solve more ILPs per fixed time and thus have less time per ILP. To solve an ILP in a shorter timeframe the attacker requires easier ILP instances. Thus a lower false-positive rate is required.

For $f = 2$ and security level V we are unable to properly explain the suddenly increasing false-positive rate. It might be that by chance, despite the high false-positive rate, the ILP instances were easy.

The signature count (Figure 5.5c) increases as f decreases and also for a fixed f the signature count increases as the security level does. Both of these phenomena have been explain in the previous section 5.2.1.

Comparison

Looking at the aspect of for which f we are able to perform a successful attack we can see that when using the "surplus of equations" parameter we can do so for any f and any security level, whereas using the "threshold" parameter we are only able to do so for for all $f \geq 2$ and $f = 1$ only for security level V. The amount of signatures required is similar for both parameters.

Thus we conclude that the parameter "surplus of equations" has a bigger impact on the attack performance than "threshold". Albeit we note that both parameters have impact on the attack performance. While the optimal solution is probably to use both parameters together with a certain weight, the attack results of the "surplus of equations" parameter are good enough for our use-case so that we will not do further research to increase the performance. Additionally, because the attack strategy is very similar for the other two signature schemes, we believe that the results we obtained for Dilithium can also be applied to the other two signature schemes. Thus we will only evaluate the "notion of success" parameter for qTESLA and BLISS.

5.2.2 qTESLA

Our simulation results show that we are able to break the shuffling countermeasure for the parameter set `qTESLA-p-I` (NIST security level I) but not so for the parameter set `qTESLA-p-III` (NIST security level III) for any reasonable m. When attacking

security level I using the default parameters our attack succeeds for all $m \leq 510$. Using the "surplus if equations" parameter we were able to succeed with only two faulted coefficients for which we require on average of 406 signatures. The exact results for security level I can be found in Figure 5.6.

Compared to Dilithium and BLISS qTesla works on a way higher dimension even in the NIST security level I ($n = 1024$). Thus we assume this is the reason we were unable to break NIST security level III of qTESLA, as it doubles the dimension compared to security level I to $n = 2048$. Higher dimensions result in higher variable-count in the corresponding ILP, exponentially increasing the difficulty to solve it.

5.2.3 BLISS

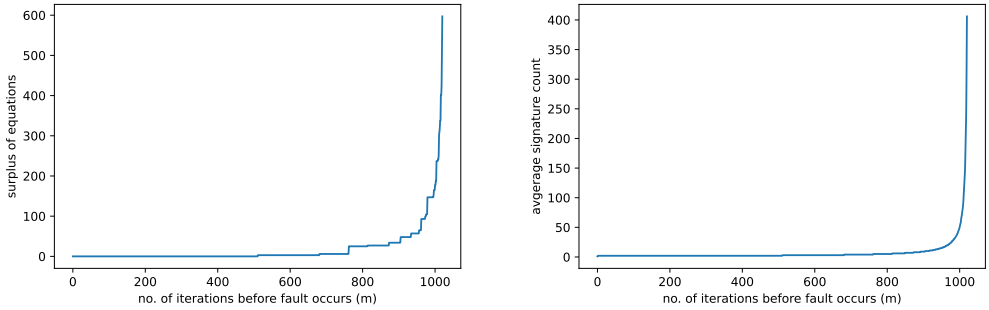
The authors of BLISS propose four different parameter sets, BLISS-I, BLISS-II, BLISS-III and BLISS-IV which aim to offer 128, 128, 160 and 192 bits of security respectively [Duc+13a, pp. 23–24]. We were able to successfully attack all these parameter sets, though in contrast to Dilithium and qTESLA (NIST security level I) we require the loop-abort to be more early. To be precise the loop-abort needs to occur at least prior the 405th loop iteration, which is around 80% of the total iteration count.

The surplus of equations needed per m and the corresponding required amount of signatures are depicted in Figure 5.7 for all BLISS parameter sets.

5.2.4 Repetition rate and required amount of signatures

All the schemes we attack are based on the Fiat-Shamir with aborts design. Thus these schemes may abort and retry. The expected amount of tries before the schemes output a valid signature is called the repetition rate. An attacker can not a priori know the repetition rate, thus he can not perfectly time his fault a priori.

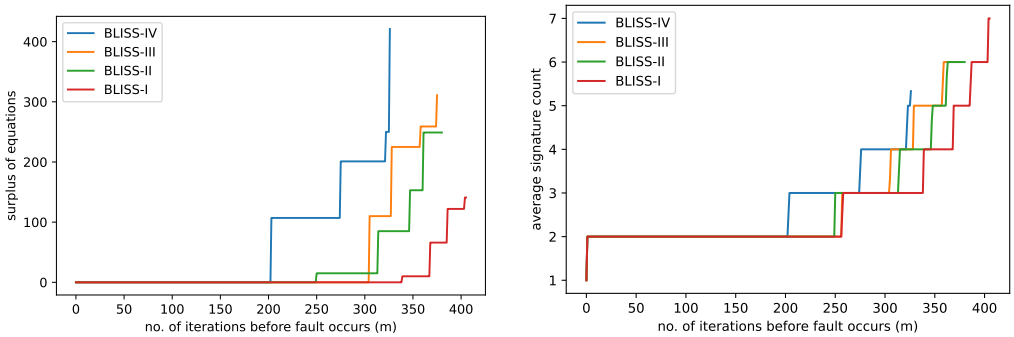
If the attacker model is strong enough such that an attacker can induce multiple loop abort faults during the signature generation, one for every possible repetition, the amount of signature required matches with the numbers we present here. If the attacker model is less strong, such that the fault can only be executed for one specific repetition count, e.g. only the first repetition, then the amount of signature required will increase, depending on the signature scheme and parameter set.



(a) Surplus of equations per m.

(b) Corresponding signature count per m.

Fig. 5.6: qTESLA (NIST security level I) simulation results. The minimal required surplus of equations and the corresponding signature count per m.



(a) Surplus of equations per m.

(b) Corresponding signature count per m.

Fig. 5.7: BLISS simulation results. The minimal required surplus of equations and the corresponding signature count per m.

NIST Security Level	II	III	V
Repetition rate	4.25	5.1	3.85

Tab. 5.1: Repetition rates for the Dilithium signature scheme [Bai+20, p. 16].

Parameter set	BLISS-I	BLISS-II	BLISS-III	BLISS-IV
Repetition rate	1.6	7.4	2.8	5.2

Tab. 5.2: Repetition rates for the BLISS signature scheme [Duc+13a, p. 24].

For deterministic signature schemes like Dilithium an attacker can generate a signature of a message μ without fault and observe the repetition rate by time measurements. Using this repetition rate he can set a proper timing for the fault (loop abort in the last repetition) and sign μ again, but this time with the a fault induced in the last repetition. Such an approach would double the amount of signature generations which need to be performed.

This does not work for probabilistic signature schemes, like BLISS and qTESLA. In this case an attacker has to set up the timing for a fault in the first repetition and repeat the signature generation until a signature is outputted after the first try. In this case the amount of signatures needed has to be scaled by the repetition rate of the particular scheme and parameter set.

qTESLA NIST security level I has an acceptance rate of $.13$ [Alk+20, p. 452], thus a repetition rate of $7,69$. The amount of required signatures need to be scaled by $7,69$. The repetition rates for Dilithium and BLISS can be found in Table 5.1 and 5.2 respectively.

Finally we want to highlight that the repetition rate does not, other than the required signatures, affect our attack efficiency, e.g. difficulty of ILP instances or the false-positive rate. The repetitions performed for a particular signature can be detected by time measurements and thus signatures which did not match the expected repetition rate will be discarded and never appear in the actual attack data.

6 Conclusion, Countermeasures and future work

Quantum computers may be able to break classic cryptography based on factoring and (elliptic curve) discrete logarithm soon. Thus we need post-quantum secure alternatives now. The NIST standardization process already revealed that Kyber is the recommended post-quantum secure key encapsulation mechanism and Dilithium the recommended post-quantum secure signature scheme to be standardized [Ala+22]. Thus in the near future these schemes will be deployed on real devices, including smartcards, microcontrollers and other devices susceptible to fault attacks but essential for our daily life. It is of great importance that before these devices are deployed in the real-world it is well known what attacks do exist and how to properly prevent them. While research on PQC fault attacks has already been conducted [Esp+17; GP18; BBK16; PP21], it was still far unclear whether the proposed countermeasures were sufficient.

In this thesis we showed how break one particular countermeasure, the shuffling countermeasure, which was so far assumed to be a sufficient countermeasure against loop-abort faults. We used multiple faulty signatures to construct a noisy equation system. We were able reduce the noise by filtering certain equations based on basic statistical analysis. Finally we constructed an ILP which was able to remove the last bit of noise and solve the equation system which revealed (part of) the secret key. We demonstrated the attack effectiveness by attacking lattice-based signature schemes based of the Fiat-Shamir with aborts over rings type. We picked three signature schemes, namely the winner of NIST's PQC standardization process, Dilithium, as well as qTESLA and BLISS. Especially for Dilithium and qTesla's parameter set `qTESLA-p-I` we showed that the timing of the loop-abort is irrelevant, as long as at least one or two coefficients of the masking polynomial(s) are zero respectively. As such an event can occur with high probability under normal operation conditions for a single signature, it is hard to detect by software using the "check for zero" countermeasure. Our results highlight that more offensive research is needed on the proposed countermeasures and more additional countermeasures need to be proposed.

6.1 Possible Countermeasures

Bindel et al. [BBK16, p. 74] presented a countermeasure against an attack, which skipped the addition of the entire masking polynomial y: Instead of adding the error polynomial on to the sc vector, we add the sc and y into a new variable. This does not directly apply to our attack scenario as we do not cause y to be zero by skipping its addition but instead we skip only part of its addition by skipping iterations in the sampling loop. Still, we can use this idea as a countermeasure against our attack by combining the sampling of y and the addition of y and sc into a single loop, instead of doing both operations separately. Thus if we try to abort the loop which samples y, we also abort the addition of sc and y. Consequently for all coefficients we fault, we only get uninitialized memory which does not contain any information. While this countermeasure is very efficient it only works for faults which target the execution flow of the program but it does not cover faults which for example target the programs memory.

As a possible countermeasure to protect BLISS, and possible other signature schemes which use the discrete gaussian distribution for their masking vectors, blinded gaussian shuffling can be used. This countermeasure was first introduced by Saarinen [Saa18, p. 82] to counteract side-channel attacks against discrete gaussian sampler. The general idea is that then adding two discrete gaussian distributed random variables X and Y with a standard deviation of σ, centered at zero, $X + Y$ is also follows the discrete gaussian distribution, centered at zero with a standard deviation of $\sqrt{\sigma^2 + \sigma^2}$. We can use this fact to construct a y-vector with standard deviation σ by adding k discrete gaussian distributed vectors with standard deviation $\sigma' = \frac{\sigma}{\sqrt{m}}$. Furthermore we shuffle all vectors before we add them together. This would require an attacker to perform k loop-abort faults and still the expected amount of zero coefficients in the resulting masking vector would be exponentially small in k. Latter because a faulted-coefficient of a sampled polynomial might be added with a non-faulted coefficient the other vector due to the shuffling. Additionally this also protects against side-channel attacks. On the other hand this strategy also decreases the performance of the discrete gaussian sampler by k-fold.

6.2 Future work

While we believe that our simulations already stress the need for more countermeasure research it would be interesting to see this attack to be implemented in practice on real hardware to further proof the point. Additionally it might be possible to

improve the performance of the attack by combining the p- and t-parameter in an optimal fashion. When evaluating "surplus of equations" parameter we choose a very conservative threshold. Especially the qTESLA signature scheme was affected by the very conservative choice of the threshold, as this threshold was also used in the ILP as a constraint for the secret polynomial coefficient variables. A stricter constraint for these variables might improve the ILP performance and thus the attack performance.

Bibliography

[AB74] R. Agarwal and C. Burrus. “Fast Convolution using fermat number transforms with applications to digital filtering”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 22.2 (Apr. 1974), pp. 87–97 (cit. on p. 4).

[Ajt96] Miklós Ajtai. “Generating Hard Instances of Lattice Problems (Extended Abstract)”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC ’96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 99–108 (cit. on pp. 5, 6, 15).

[Ala+22] Gorjan Alagic, David A. Cooper, Quynh Dang, et al. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST Interagency/Internal Report (NISTIR) 879716. Gaithersburg, Maryland, USA: National Institute of Standards and Technology, July 2022 (cit. on pp. 1, 2, 8, 43).

[Alk+20] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, et al. “The Lattice-Based Digital Signature Scheme qTESLA”. In: *Applied Cryptography and Network Security*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12146. Lecture Notes in Computer Science. Rome, Italy: Springer International Publishing, Oct. 2020, pp. 441–460 (cit. on pp. 2, 11, 12, 41).

[Aum+03] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. “Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures”. In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by Burton S. Kaliski, çetin K. Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Redwood Shores, California, USA: Springer Berlin Heidelberg, Aug. 2003, pp. 260–275 (cit. on p. 1).

[Bai+20] Shi Bai, Léo Ducas, Eike Kiltz, et al. *CRYSTALS-Dilithium*. Algorithm Specifications and Supporting Documentation. `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`. Gaithersburg, Maryland, USA: National Institute of Standards and Technology, Oct. 2020 (cit. on pp. 8, 41).

[BBK16] Nina Bindel, Johannes Buchmann, and Juliane Krämer. “Lattice-Based Signature Schemes and Their Sensitivity to Fault Attacks”. In: *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. Ed. by Juan E. Guerrero. Santa Barbara, California, USA: Institute of Electrical and Electronics Engineers, Aug. 2016, pp. 63–77 (cit. on pp. 15, 43, 44).

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. “On the Importance of Checking Cryptographic Protocols for Faults”. In: *Advances in Cryptology — EUROCRYPT ’97*. Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Konstanz, Germany: Springer Berlin Heidelberg, Mar. 1997, pp. 37–51 (cit. on p. 13).

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) Fully Homomorphic Encryption without Bootstrapping”. In: *ACM Transactions on Computation Theory* 6.3 (July 2014) (cit. on p. 6).

[Deh+12] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, et al. *Injection of transient faults using electromagnetic pulses Practical results on a cryptographic system*. Journal of Cryptology ePrint Archive: Report 2012/123. 2012 (cit. on p. 14).

[Duc+13a] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. *Lattice Signatures and Bimodal Gaussians*. Cryptology ePrint Archive, Paper 2013/383. 2013 (cit. on pp. 13, 39, 41).

[Duc+13b] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. “Lattice Signatures and Bimodal Gaussians”. In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Santa Barbara, California, USA: Springer Berlin Heidelberg, Aug. 2013, pp. 40–56 (cit. on pp. 2, 12, 27).

[Duc+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, et al. “CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.1 (Feb. 2018), pp. 238–268 (cit. on p. 2).

[Esp+17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. “Loop-Abort Faults on Lattice-Based Fiat-Shamir and Hash-and-Sign Signatures”. In: *Selected Areas in Cryptography – SAC 2016*. Ed. by Roberto Avanzi and Howard Heys. Vol. 10532. Lecture Notes in Computer Science. St. John’s, Newfoundland and Labrador, Canada: Springer Cham, Aug. 2017, pp. 140–158 (cit. on pp. 2, 15, 17, 18, 43).

[Esp+18] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. “Loop-Abort Faults on Lattice-Based Signature Schemes and Key Exchange Protocols”. In: *IEEE Transactions on Computers* 67.11 (Nov. 2018), pp. 1535–1549 (cit. on pp. 14, 15).

[FS87] Amos Fiat and Adi Shamir. “How To Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *Advances in Cryptology — CRYPTO’ 86*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Santa Barbara, California, USA: Springer Berlin Heidelberg, 1987, pp. 186–194 (cit. on p. 7).

[FY48] Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research*. English. London: Oliver and Boyd, 1948 (cit. on p. 18).

[GT04] Christophe Giraud and Hugues Thiebeauld. “A Survey on Fault Attacks”. In: *Smart Card Research and Advanced Applications VI*. Ed. by Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kalam. Vol. 153. IFIP Advances in Information and Communication Technology. Toulouse, France: Springer New York, NY, Aug. 2004, pp. 159–176 (cit. on p. 14).

[GP18] Leon Groot Bruinderink and Peter Pessl. “Differential Fault Attacks on Deterministic Lattice Signatures”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.3 (Aug. 2018), pp. 21–43 (cit. on pp. 15, 26, 43).

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 2nd ed. Chapman & Hall, 2020 (cit. on p. 3).

[KQ07] Chong Hee Kim and Jean-Jacques Quisquater. “Faults, Injection Methods, and Fault Attacks”. In: *IEEE Design Test of Computers* 24.6 (Dec. 2007), pp. 544–545 (cit. on pp. 13, 14).

[Lyu09] Vadim Lyubashevsky. “Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures”. In: *Advances in Cryptology — ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Tokyo, Japan: Springer Berlin Heidelberg, Dec. 2009, pp. 598–616 (cit. on pp. 7, 8).

[Lyu08] Vadim Lyubashevsky. “Lattice-Based Identification Schemes Secure Under Active Attacks”. In: *Public Key Cryptography — PKC 2008*. Ed. by Ronald Cramer. Vol. 4939. Lecture Notes in Computer Science. Barcelona, Spain: Springer Berlin Heidelberg, Mar. 2008, pp. 162–179 (cit. on p. 4).

[LM06] Vadim Lyubashevsky and Daniele Micciancio. “Generalized Compact Knapsacks Are Collision Resistant”. In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Vol. 4052. Lecture Notes in Computer Science. Venice, Italy: Springer Berlin Heidelberg, July 2006, pp. 144–155 (cit. on p. 4).

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On Ideal Lattices and Learning with Errors over Rings”. In: *Advances in Cryptology — EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera, France: Springer Berlin Heidelberg, 2010, pp. 1–23 (cit. on p. 6).

[Mar+22] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. *Profiling Side-Channel Attacks on Dilithium: A Small Bit-Fiddling Leak Breaks It All*. Cryptology ePrint Archive, Paper 2022/106. 2022 (cit. on p. 21).

[Moo+20] Dustin Moody, Gorjan Alagic, Daniel C. Apon, et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST Interagency/Internal Report (NISTIR) 8309. Gaithersburg, Maryland, USA: National Institute of Standards and Technology, July 2020 (cit. on p. 15).

[PBY17] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. “To BLISS-B or Not to Be: Attacking StrongSwan’s Implementation of Post-Quantum Signatures”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Special Interest Group on Security, Audit and Control. Dallas, Texas, USA: Association for Computing Machinery, Nov. 2017, pp. 1843–1855 (cit. on p. 20).

[PP21] Peter Pessl and Lukas Prokop. “Fault Attacks on CCA-secure Lattice KEMs”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.2 (Feb. 2021), pp. 37–60 (cit. on p. 43).

[Rav+18] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. *Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate*. Cryptology ePrint Archive, Paper 2018/821. Sept. 2018 (cit. on p. 26).

[Saa18] Markku-Juhani O. Saarinen. “Arithmetic coding and blinding countermeasures for lattice signatures”. In: *Journal of Cryptographic Engineering* 8.1 (Apr. 1, 2018), pp. 71–84 (cit. on p. 44).

[Sho99] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Review* 41.2 (1999), pp. 303–332. eprint: `https://doi.org/10.1137/S0036144598347011` (cit. on p. 1).

[ST16] National Institute of Standards and Technology. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization/Call-for-Proposals`. Dec. 2016 (cit. on pp. 1, 15).

Webpages

[@Sag22] SageMath Developers. *Discrete Gaussian Samplers over the Integers*. 2022. (Visited on June 20, 2022) (cit. on p. 27).