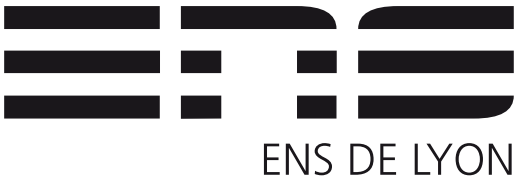

Towards optimal trapdoors for discrete Gaussian sampling over unstructured lattices. Application to hash-and-sign signatures.

SQUIRRELS: An efficient and secure post-quantum signature scheme based on plain lattices

by Guilhem Niot

Master Thesis

Dr. Thomas Espitau, PQShield
Thesis Supervisor

Prof. Serge Vaudenay, EPFL
Prof. Aurélien Garivier, ENS Lyon
Thesis Advisor

Dr. Mehdi Tibouchi, NTT Corporation
External Expert

PQShield, 75012 Paris

August 2023

Acknowledgments

I would first like to thank my thesis supervisor, Thomas Espitau, who guided and worked with me on the realization of this project. He has been a great source of insightful remarks. I warmly thank Mehdi Tibouchi for his fundamental contributions to SQUIRRELS, and for giving me the opportunity to visit his lab in Tokyo.

I would also like to thank the researchers of PQShield Paris, and especially Thomas Prest, who welcomed me to do this internship with them, and integrated me in their team.

I am grateful to Serge Vaudenay and Aurélien Garivier for accepting to be my advisors, and before that for teaching me valuable lectures.

Lastly, I will also take this chance to mention all the researchers I met with along this journey, which often led to interesting discussions and valuable insights on my future work and research interests.

Paris, August 25, 2023 Guilhem Niot

Abstract

The development of quantum computing opened new possibilities in cryptography and cryptanalysis. Notably, the invention of Shor's algorithm [46] – allowing one to factor an integer in polynomial time – showed that quantum computers would break widely deployed primitives based on RSA and elliptic curves, and protocols relying on them such as TLS. The paradigm "Store now, Decrypt later", as well as the possibility that quantum computers may be available in the near future, already make this a critical threat to the security of today's communications. This motivated the scientific community to work on post-quantum replacements and to collaborate with the industry to deploy them as early as possible.

The National Institute of Standards and Technology (NIST) initiated a standardization effort of post-quantum primitives, which led to the upcoming standardization of the stateful hash-based signature schemes XMSS and LMS [13], 3 lattice-based schemes (the Kyber KEM [45] and the signatures Dilithium and Falcon [44, 32]) and 1 stateless hash-based signature (SPHINCS [27]), further demonstrating the increasing significance and practicality of post-quantum cryptographic primitives.

However, given the significant challenges associated with the costly and time-consuming deployment of entirely new cryptographic systems, coupled with the need to ensure the longevity of these systems over decades, there are application domains where adopting a *conservative* approach to selecting a post-quantum candidate scheme is preferable. In light of the unpredictable trajectory of quantum computing technology and quantum cryptanalysis in years to come, practitioners with valuable data requiring long-term confidentiality and authenticity guarantees may want to prioritize security and simplicity over premature optimization. In that perspective, the question of which class of hard problem to rely on is critical.

Contents

Chapter 1

Introduction

1.1 Context

The theory of quantum computing was developed in the '80s, with the conceptualization of the quantum Turing machine. In this theory, bits are replaced by qubits that allow a probabilistic superposition of 0 and 1. In theory, quantum computers can solve some classically hard problems, such as determining whether a blackbox function is constant. It is also possible to exploit quantum intrication to intuitively make several parallel computations exponential in the number of qubits available. In the field of cryptography, Shor's algorithm [46] solves the problem of factoring and the discrete logarithm problem in polynomial time which are believed to be hard to solve classically. Current primitives based on RSA and elliptic curves are hence broken by quantum computing.

It is however not completely clear whether quantum computers are practical. At the time of writing this thesis, quantum computers are limited to several dozen qubits, several orders below what is required by Shor's algorithm. And there are doubts about the scalability of the current methods. Yet, Google claimed quantum supremacy in 2019 [24], even though not everyone agrees this is true, this shows that the field is in active development and there are regular improvements in the technology which may one day make quantum computers scalable.

Quantum computing theory is here to stay, and although practice is lagging, it is important to consider the risks of sticking with cryptography possibly broken in the near future. The paradigm "Store now, decrypt later" makes it paramount to develop and deploy quantum-

resistant cryptographic alternatives.

Quantum computers are not almighty, notably, NP-hard problems have not been solved in polynomial time by quantum computers. For symmetric cryptography, quantum cryptography usually reduces solving time to the square root of the size of the input size, which is a quadratic speedup and not exponential. For asymmetric cryptography, new hardness assumptions are made, that are believed to be hard to solve even with quantum computers, and new primitives based on them are designed.

1.2 Lattice-based cryptography

Some of the main problems believed to be hard to solve on quantum computers are based on lattices. The first lattice-based cryptographic construction was introduced in 1996 by Ajtai [4]. It was followed in 1998 by NTRU [26] based on stronger assumptions allowing more efficient construction, and two decades of intense research in the field, with the construction of various cryptographic primitives, key exchange [45, 37, 41], signatures [44, 32, 20], homomorphic encryption [49, 12], and many others.

The security of several lattice problems is supported by average to worst-case reductions [5, 22], and there was intense cryptanalysis of lattice problems which gives strong confidence in the security of these problems.

1.2.1 Structured vs. unstructured lattices.

The security of public-key cryptosystems relies on the assumption that certain computational problems are difficult to solve. In lattice-based cryptography, two related important problems are the Learning with Errors (LWE) and the Short Integer Solution (SIS) problem. The former involves solving a noisy, random linear system over the ring $\mathbb{Z}/q\mathbb{Z}$, whereas the latter asks to find a short solution to a linear system, again over $\mathbb{Z}/q\mathbb{Z}$. Both problems can also be interpreted as approximate close vector problems (i.e., the problems of decoding errors of a certain size) in random q-ary lattices.

Variants of these problems include algebraically structured versions such as *Ring-LWE* [33] and *Module-LWE* [29], as well as problems associated with the so-called NTRU lattices. These variants correspond to decoding problems in lattices over rings of algebraic integers (endowing the lattices themselves with additional algebraic structure). Cryptosystems based on those

structured assumptions and structured lattices generally offer greater efficiency, since the corresponding lattices admit more compact representations, and typically benefit from the faster arithmetic of the underlying rings. In principle, however, the additional structure could also, introduce vulnerabilities that do not exist in the unstructured setting.

The current state of the art indicates that the recommended parameterizations for algebraically structured lattice problems do not appear to exhibit specific weaknesses when compared to plain lattice versions. However, the (quantum) complexity of certain related problems on specific types of algebraic lattices is lower than their counterparts on general lattices (see, e.g., [14, 15]). Whether new cryptanalytic techniques may in the future improve and extend those stronger attacks that exist in the structured setting to the point of meaningfully reducing the security of cryptosystems based on NTRU or structured variants of LWE and SIS remain to be seen. Uncertainty in this matter does however make unstructured lattices appear as a clearly more conservative choice.

1.3 SQUIRRELS: a signature scheme based on unstructured lattices

Given the uncertain long-term prospects of algebraically structured lattices and the need for post-quantum standards to remain secure, this thesis introduces SQUIRRELS – a new digital signature scheme based on plain lattice problems, i.e., without additional algebraic structure. SQUIRRELS is an instantiation of the GPV framework [23] on a large class of lattices – called co-cyclic lattices, and with a high density among plain lattices. It leverages novel optimizations of the key generation and verification procedures to achieve small signature sizes, and fast signature generation and verification. We choose to employ conservative parameterizations for enhanced security. While this choice incurs some efficiency trade-offs compared to algebraic variants, we believe it ensures robustness against potential weaknesses in the future. Furthermore, the use of plain lattices presents minimal restrictions on parameter choices, making it possible to reach specific security levels in a fine-grained manner.

SQUIRRELS was submitted to NIST 2023 Call for Additional Digital Signature Schemes [1], with a submission package including an extended specification and analysis, a reference code and test vectors. Notably, at NIST security level I, SQUIRRELS produces signatures of 1019 bytes, which is in between the size of the signatures produced by Falcon and Dilithium, the two lattice-based signature NIST finalists. It has however larger public keys of about 600kB against 1kB for Falcon. Our signature scheme offers an interesting trade-off between security and communication size. It becomes particularly viable for long-term use, when the public

key is rarely transmitted on the network.

Chapter 2

Background

Notations. This paragraph states the convention that will be used in this document.

- Vectors are denoted with bold lower-case letters (e.g. $\mathbf{a}, \mathbf{v}$), matrices are denoted in bold upper-case letters (e.g. $\mathbf{A}, \mathbf{M}$). For a set D, the set of n-dimensional vectors with coordinates in D is denoted D^n, the set of matrices of n rows, m columns with coordinates in D is denoted $D^{n\times m}$.
- Matrices and vectors are denoted in row notation.
- Given an n-dimension vector $\mathbf{v}$, its i-th coordinate is written v_i for $1 \leqslant i \leqslant n$. Given an n-by-m matrix $\mathbf{A}$, its (i,j)-th coordinate (the coordinate in its i-th row, j-th column) is written $\mathrm{A}_{i,j}$ for $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m$.
- The matrix multiplication is denoted $\mathbf{A}\cdot\mathbf{B}$. It is extended to n-dimensional vectors by interpreting them as a 1-by-n matrix.
- The transpose of the matrix $\mathbf{A}$ is denoted $\mathbf{A}^T$. Its orthogonal space is denoted $\mathbf{A}^\perp = \{\mathbf{x} \mid \forall \mathbf{y}, \langle \mathbf{x}, \mathbf{y}\cdot\mathbf{A}\rangle = 0\}$
- The ring of integers is denoted $\mathbb{Z}$. For a positive integer q, we denote the quotient ring of integers modulo q as $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$.
- For a finite set S, we denote the uniform distribution over S as $\mathcal{U}(S)$.
- The floor of a real number a, i.e., the largest integer less than or equal to a, is denoted by $\lfloor a \rfloor$.

- For a real vector $\mathbf{v} \in \mathbb{R}^n$, its euclidean norm (i.e. ℓ_2) is denoted $||\mathbf{v}||$.
- For an integer a, and a positive integer p, we denote the reduction of a modulo p by $a \bmod p \in [0, p-1]$.
- For two n-dimensional vectors $\mathbf{a}, \mathbf{b}$ over a common ring, their inner product is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i$.
- For an n-by-m matrix $\mathbf{B}$, we denote $\tilde{\mathbf{B}}$ its Gram-Schmidt orthogonalization i.e. the matrix $\tilde{\mathbf{B}}$ verifying:

$$\forall i \in [\![1, n]\!], \tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{1 \leqslant i < j} \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \tilde{\mathbf{b}}_j$$

  We note the Gram-Schmidt norm of a matrix $||\mathbf{B}||_{\text{GS}} = \max_i ||\tilde{\mathbf{b}}_i||$.

2.1 Introduction to Lattices

Lattices are a common object used in post-quantum cryptography. They are very versatile and allow designing a broad range of primitives while remaining very efficient. This section gives a brief introduction to lattices and some of the main tools used to study them.

Definition 2.1.1 (Lattice)**.** *Given $H = \mathbb{R}^m$, a lattice is a discrete subgroup of H. For a basis $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} \in \mathbb{R}^{n \times m}$, we note $\mathscr{L}(\mathbf{B})$ and call lattice generated by $\mathbf{B}$ the set of vectors*

$$\left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$$

A lattice is commonly noted Λ or $\mathscr{L}(\mathbf{B})$. The rank of a lattice – commonly noted n – is the cardinal of a basis of the lattice.

Definition 2.1.2 (Successive Minima of a Lattice)**.** *Let $\Lambda \in \mathbb{R}^m$ a lattice of rank n. For $i \in [\![1, n]\!]$, we note $\lambda_i(\Lambda)$ and call i-th successive minimum of Λ the value:*

$$\lambda_i(\Lambda) = \inf_{\substack{\mathbf{B} \text{ s.t. } \Lambda = \mathscr{L}(\mathbf{B}) \\ ||\mathbf{b}_1|| \leqslant ||\mathbf{b}_2|| \leqslant \ldots \leqslant ||\mathbf{b}_n||}} ||\mathbf{b}_i||$$

Definition 2.1.3 (Determinant of a Lattice)**.** *Let* $\Lambda = \mathscr{L}(\mathbf{B})$ *be a lattice of rank* n*. We call determinant of* Λ *the value* $\det(\Lambda) := \sqrt{\det(\mathbf{B}\mathbf{B}^t)}$*. This value is independent of the basis* $\mathbf{B}$ *chosen.*

The basis of a lattice can be transformed to respect extra properties. Two interesting such forms are the Hermite Normal Form, which is unique and allows us to simply verify whether two bases describe the same lattice, and the δ-LLL reduced form which intuitively reduces the basis and sorts Gram-Schmidt norms of the basis. Both of these forms can be computed in polynomial time.

Definition 2.1.4 (δ-LLL reduced basis)**.** *For* $\mathbf{B} = [\mathbf{b}_1|\cdots|\mathbf{b}_n]$ *a matrix, the definition of a* δ*-LLL-reduced basis is as follows: define* $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j \cdot \tilde{\mathbf{b}}_j \rangle}$ *for any* $1 \leq j < i \leq n$*. We say that* $\mathbf{B}$ *is* δ *LLL-reduced for a parameter* $\delta \in (\frac{1}{4}, 1)$ *if the following holds:*

- *Size-reduced: for* $1 \leq j < i \leq n$*:* $|\mu_{i,j}| \leq \frac{1}{2}$.
- *Lovász condition: for* $k = 2, \cdots, n$*,* $\delta ||\tilde{\mathbf{b}}_{k-1}||^2 \leq |\tilde{\mathbf{b}}_k||^2 + \mu_{k,k-1}^2 |\tilde{\mathbf{b}}_{k-1}||^2$.

The LLL [30] algorithm outputs an LLL-reduced basis.

Definition 2.1.5 (Hermite Normal Form (HNF))**.** *Any* m*-by-*n *matrix* $\mathbf{B}$ *can be factored as* $\mathbf{U} \cdot \mathbf{H}$ *with* $\mathbf{U}$ *being an* m*-by-*m *unimodular matrix, and* $\mathbf{H}$ *is an* m*-by-*n *matrix verifying:*

- $\mathbf{H}$ *is upper triangular with positive coefficients, and rows of zeros are below any non-zero row.*
- *The first non-zero entry in row* i*, called the pivot and noted* $e_i = \mathrm{H}_{i,j_i}$ *is strictly to the right of the non-zero entry of the previous row:* $j_i > j_{i-1}$.
- *The pivot of each row is strictly larger than all entries above it in the same column:* $\forall 1 \leqslant i' < i, \mathrm{H}_{i',j_i} < \mathrm{H}_{i,j_i}$.

2.1.1 On co-cyclic lattices

The structure of the quotient group $\mathbb{Z}^n/\Lambda$, where Λ is an integer lattice, that holds significant importance in the study of lattices. It provides insights into the average complexity of lattice problems, as highlighted in works such as [3] and its generalization [22]. In particular, when this quotient group is cyclic, meaning it is spanned by a single element, we refer to the lattice

as *co-cyclic*. Co-cyclic lattices exhibit a natural density of approximately 85% [38]. Moreover, Paz and Schnorr demonstrated in [39] that the worst-case hardness problems, such as SVP (Shortest Vector Problem) and CVP (Closest Vector Problem), on co-cyclic lattices are as challenging as those on unconstrained lattices.

This class of lattices offers both strong security guarantees and practical efficiency as discussed in section 3.1.

2.1.2 Problems over lattices

We recall several problems over lattices of interest for this thesis.

Definition 2.1.6 (**SVP** – Shortest Vector Problem)**.** *Given a n-dimensional lattice Λ, find a lattice vector $\mathbf{v}$ such that $||v|| = \lambda_1(\Lambda)$.*

Definition 2.1.7 (**ASVP**$_\gamma$ – Approximate Shortest Vector Problem)**.** *Given a n-dimensional lattice Λ and $\gamma \geqslant 1$ a function of n, find a lattice vector $\mathbf{v}$ such that $||v|| \leqslant \gamma \cdot \lambda_1(\Lambda)$.*

Definition 2.1.8 (**CVP** – Closest Vector Problem)**.** *Given a n-dimensional lattice Λ and a point $c \in H$, find a lattice vector $\mathbf{v}$ minimizing $||c - v||$.*

These problems are hard on classical computers, and quantum computers are believed to not significantly decrease their complexity, as opposed to problems such as the factoring of integers, and discrete logarithm.

Some other problems arise from cryptographic constructions themselves.

Definition 2.1.9 (**SIS**$_{n,m,q,\beta}$ – Short Integer Solution)**.** *Let n and $m, q = Poly(n)$ some integers. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly random, find a non-zero vector $\mathbf{z}$ such that $\mathbf{A}\mathbf{z}^t = \mathbf{0} \bmod q$ and $||\mathbf{z}|| \leqslant \beta$.*

Definition 2.1.10 (**ISIS**$_{c,n,m,q,\beta}$ – Inhomogeneous Short Integer Solution)**.** *Let n and $m, q = Poly(n)$ some integers. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly random and $\mathbf{c} \in \mathbb{Z}_q^n$, find a non-zero vector $\mathbf{z}$ such that $\mathbf{A}\mathbf{z}^t = \mathbf{c}^t \bmod q$ and $||\mathbf{z}|| \leqslant \beta$.*

ISIS can be reduced to **SIS**, and **SIS** was proven to be as hard as worst-case **ASVP**$_\gamma$ [4]. This means that cryptographic constructions based on (**I**)**SIS** can ultimately rely on standard lattice hardness assumptions.

The problems mentioned below are formally defined as the generalization of the (**I**)**SIS** problems when the structure of the quotient $\mathbb{Z}^n/\Lambda$ is prescribed to a group G.

Definition 2.1.11 (**GSIS**$_{G,m,\beta}$ – Group Short Integer Solution)**.** *Let G be a finite abelian group. Given $(g_1,...,g_m) \in G^m$, find $x \in \mathbb{Z}^m$ such that $\sum_{i=1}^{m} x_i \cdot g_i = 0_G$ and $||x|| \leqslant \beta$.*

Definition 2.1.12 (**GISIS**$_{G,c,m,\beta}$ – Group Inhomogeneous Short Integer Solution)**.** *Let G be a finite abelian group. Given $(g_1,...,g_m) \in G^m$ and $c \in G$, find $x \in \mathbb{Z}^m$ such that $\sum_{i=1}^{m} x_i \cdot g_i = c$ and $||x|| \leqslant \beta$.*

We have **SIS**$_{n,m,q,\beta}$ = **GSIS**$_{\mathbb{Z}_q^n,m,\beta}$ and **ISIS**$_{c,n,m,q,\beta}$ = **GSIS**$_{\mathbb{Z}_q^n,c,m,\beta}$.

G(I)SIS was proven to be as hard as **ASVP**$_\gamma$ in [22] under the assumption that G is sufficiently large. This reduction applies to co-cyclic lattices with a large determinant Δ for which $\mathbb{Z}^n/\Lambda = \mathbb{Z}_\Delta$. The high density of co-cyclic lattices among the moduli space of all integer lattices (without the determinant constraint) also supports the hardness of these problems for co-cyclic lattices by allowing to reduce average **G(I)SIS** on co-cyclic lattices to **G(I)SIS** on all integer lattices. Thus, strong reduction results support the use of co-cyclic lattices in a lattice-based cryptographic scheme.

2.2 Digital Signature Schemes

A digital signature is a cryptographic scheme that allows one to verify the authenticity of a digital document. The signer typically sends a signature alongside a document, that gives confidence on its origin to anyone verifying it. Signatures have a broad set of use cases, such as software distribution, or TLS handshakes.

A signature typically consists of three routines:

- a *key generation* procedure, which randomly generates a keypair, with a secret key used for signature generation, and a public key that can be shared and allows signature verification.
- a *signing* procedure, given a document and the secret key, generates a signature for that document.
- a *verification* procedure, given a document, a signature, and the public key, verifies that the signature is valid for the document.

We require the scheme to be *correct*, i.e. any signature generated by the signing procedure must pass the verification. And it has to be *secure* or *existentially unforgeable*, i.e. it is hard

to generate a valid signature for a message without knowledge of the private key. We say that a scheme is *strongly* unforgeable if additionally no new signature can be generated for a previously signed message.

2.2.1 Lattice-based signature scheme with the GPV framework

In 2008, Gentry, Peikert, and Vaikuntanathan [23] introduced a provably secure – and even strongly unforgeable – lattice-based signature scheme under the **SIS** assumption. The framework relies on the existence of a trapdoor basis, which serves as the secret key and exhibits excellent properties for Gaussian sampling to generate a lattice point close to the hash of the message. The verification process involves checking whether the signature belongs to the lattice using linear algebra techniques and ensuring that the distance between the lattice point and the message hash is sufficiently small. Over time, this robust framework has undergone enhancements to offer highly compact and efficient signature schemes, most notably resulting in the NIST-standardized Falcon [44] and variants such as Mitaka [20].

The ancestor of this framework is the GGH signature scheme [25], with the main difference being the deterministic procedure to find a point close to the hash of the message, which was leaking information about the secret basis. The GPV signature represents an improvement over this construction by randomizing the point close to the hash and making it independent from the secret basis.

The GPV signature can be described as follows:

- The public key is a full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n} (m < n)$.
- The secret key is a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times n}$ with short entries such that $\mathbf{B} \cdot \mathbf{A}^T = 0$.
- Given a message m', we first hash it to $H(m')$ where H is a hash function defined as $\{0,1\}^* \rightarrow \mathbb{Z}_q^n$. A signature is a short $\mathbf{s} \in \mathbb{Z}_q^n$ such that $\mathbf{s} \cdot \mathbf{A}^T = H(m')$. It is straightforward to check the validity of $\mathbf{v}$ by verifying that $\mathbf{s}$ is indeed short and $\mathbf{s} \cdot \mathbf{A}^T = H(m')$.
- To generate a signature, first a preimage $\mathbf{c}_0 \in \mathbb{Z}_q^n$ is computed such that $\mathbf{c}_0 \cdot \mathbf{A}^T = H(m')$, then $\mathbf{B}$ is used to find a vector $\mathbf{c}$ in the lattice spanned by $\mathbf{B}$ that is close to $\mathbf{c}_0$. The difference $\mathbf{s} = \mathbf{c_0} - \mathbf{c}$ is a valid signature, because $\mathbf{s} \cdot \mathbf{A}^T = \mathbf{c_0} \cdot \mathbf{A}^T - \mathbf{c} \cdot \mathbf{A}^T = H(m')$.

This signature is strongly unforgeable under the **SIS** assumption – i.e. an adversary cannot produce a new signature, even for messages on which it queried the signing oracle. It also

assumes that we do not rerun the signature generation twice for the same message. In practice, we enforce the second condition by sampling a salt in the signature generation and appending it to the message before hashing.

2.3 Babai's Algorithms

Given a lattice $\mathscr{L}(\mathbf{B})$ and a point $\mathbf{c}$, Babai's algorithms find a point $\mathbf{v} \in \mathscr{L}(\mathbf{B})$ close to $\mathbf{c}$. Babai formalized two algorithms in [9]: RoundOff and NearestPlane. We present the NearestPlane algorithm only here, which was used in the GPV signature, and whose core idea was reused to design the Klein Gaussian Sampler introduced in section 2.4. The Klein Sampler is used in SQUIRRELS to sample lattice vectors close to the message hash.

Algorithm 1 NearestPlane$(\mathbf{B}, \mathbf{c})$

Require: A basis $\mathbf{B}$ and a vector $\mathbf{c} \in \mathbb{R}^m$.
Ensure: A vector $\mathbf{v}$ close to $\mathbf{c}$.

$\mathbf{c}_n \leftarrow \mathbf{c}$
$\mathbf{v}_n \leftarrow 0$
for $i = n, ..., 1$ **do**
  $d_i \leftarrow \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle / ||\tilde{\mathbf{b}}_i||^2$
  $z_i \leftarrow \lfloor d_i \rceil$
  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \cdot \mathbf{b}_i$
  $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i \cdot \mathbf{b}_i$
end for
return $\mathbf{v}_0$

Proposition 2.3.1. *The vector* $\mathbf{v}$ *returned by* NearestPlane$(\mathbf{B}, \mathbf{c})$ *verifies [42, prop. 2.26]:*

$$||\mathbf{v} - \mathbf{c}|| \leqslant \frac{\sqrt{n}}{2} ||\mathbf{B}||_{GS}$$

This is a result of the fact that $\mathbf{v}$ *ends in the parallelepiped formed by the Gram-Schmidt vectors of* $\mathbf{B}$.

2.4 Klein Gaussian Sampler

The Klein Gaussian Sampler is a randomized version of NearestPlane sampling vectors according to a discrete Gaussian distribution over the lattice, centered on the target vector c.

Algorithm 2 KleinSampler$(\mathbf{B}, \mathbf{c}, \sigma)$

Require: A basis $\mathbf{B}$ and a vector $\mathbf{c} \in \mathbb{R}^m$.
Ensure: A vector $\mathbf{v}$ following distribution $D_{\mathscr{L}(\mathbf{B}),\sigma,\mathbf{c}}$

```
c_n ← c
v_n ← 0
for i = n, ..., 1 do
    d_i ← ⟨c_i, b̃_i⟩ / ||b̃_i||^2
    σ_i ← σ / ||b̃_i||
    z_i ← ⌊d_i⌉_{σ_i}          ▷ Gaussian sampling in ℤ with mean d_i, std σ_i
    c_{i-1} ← c_i − z_i · b_i
    v_{i-1} ← v_i + z_i · b_i
end for
return v_0
```

The discrete Gaussian distribution is defined as follows:

- The Gaussian function $\rho_\sigma : \mathbb{R}^m \to \mathbb{R}$ is defined as $\rho_\sigma(\mathbf{x}) := \exp(-\sigma^2 ||\mathbf{x}||^2/2)$.
- Then, the discrete Gaussian distribution over Λ centered in $\mathbf{c}$ is defined as:

$$\forall x \in \Lambda, D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_\sigma(\mathbf{x}-\mathbf{c})}{\rho_\sigma(\Lambda-\mathbf{c})}$$

Then, we can obtain a theorem over the quality of the Klein sampler:

Theorem 2.4.1. *Let $\lambda \in \mathbb{N}$ and $\varepsilon = 2^{-\lambda}$. For any basis $\mathbf{B} \in \mathbb{Z}^{n\times m}$, and a target vector $c \in \mathbb{R}^m$, the statistical distance between the output distribution of KleinSampler$(\mathbf{B}, \sigma, \mathbf{c})$ and $D_{\mathscr{L}(\mathbf{B}),\sigma,\mathbf{c}}$ is upper bounded by $2^{-\lambda}$, provided [42, theorem 2.33]:*

$$\sigma \geqslant ||\mathbf{B}||_{GS} \cdot \frac{1}{\pi} \cdot \sqrt{\frac{1}{2}\log\left(2n(1+\frac{1}{\varepsilon})\right)}$$

There is a larger theory to obtain this theorem, related to lattice duals and the notion of "smoothing parameter" as defined in [35], but we stick to a simpler version here.

Hence, for a σ large enough, the output of KleinSampler is independent of the secret basis, which ensures sound security in the GPV signature. In practice, we take ε smaller than in the above theorem, using a result from [43] bounding the Rényi divergence between these two distributions.

Chapter 3

Design

This section describes and justifies some of the main choices made in the design of SQUIRRELS.

3.1 The choice of unstructured lattices

As seen in subsection 1.2.1 and subsection 2.1.1, unstructured lattices and co-cyclic lattices offer strong security guarantees and worst-case reductions which make them great candidates for the design of conservative cryptographic primitives. Co-cyclic lattices also have a practical interest in the context of a GPV signature, with the possibility to efficiently implement lattice membership as required in the verification procedure.

Indeed, an equivalent characterization of co-cyclic lattices, as described in [39], involves the existence of a vector $\mathbf{w}$ such that $\Lambda = \{\mathbf{x} \mid \langle \mathbf{x}, \mathbf{w} \rangle \bmod d = 0\}$. In other words, the lattice Λ can be defined as the set of all vectors $\mathbf{x}$ for which the inner product of $\mathbf{x}$ and $\mathbf{w}$ modulo d yields a specific value. When the lattice determinant is furthermore square-free with large factors, this characterization also relates to the row Hermite Normal Form (HNF) of the lattice, which takes the following form with high probability ($\gtrapprox 1 - 2^{-23}$ in our case):

$$\begin{bmatrix} \mathbf{I}_{n-1} & \mathbf{v}_{\text{check}}^T \\ \mathbf{0} & \Delta \end{bmatrix} \tag{3.1}$$

where $\Delta = \det(\Lambda)$.

This characterization allows us to perform an efficient *lattice membership check*. Computing the row HNF allows us to find such a vector $\mathbf{w}$: $\mathbf{w} = (\mathbf{v}_{\text{check}}, -1)$. Indeed, given a point $\mathbf{c} \in \mathbb{Z}^n$ and the HNF of a co-cyclic lattice Λ:

$$\begin{aligned}
\mathbf{c} = (c_1, ..., c_n) \in \Lambda &\iff \exists \mathbf{Y} = (y_1, ..., y_n) \mid \mathbf{Y} \cdot \text{HNF}(\mathscr{L}) = \mathbf{c} \\
&\iff \exists \mathbf{Y} \mid c_1 = y_1, c_2 = y_2, ..., c_{n-1} = y_{n-1}, \\
&\qquad\qquad c_n = \sum_{1 \leqslant i \leqslant n-1} y_i \cdot v_{\text{check},i} + y_n \cdot \Delta \\
&\iff c_n = \sum_{1 \leqslant i \leqslant n-1} c_i \cdot v_{\text{check},i} \bmod \Delta \qquad (3.2)
\end{aligned}$$

Hence, in the context of GPV-type signatures, we can achieve efficient verification when the underlying lattice is co-cyclic. To fully capitalize on this observation and create a practical signature scheme, our focus lies in the construction of reliable trapdoors specifically designed for co-cyclic lattices. By successfully addressing this aspect, we can develop an effective and secure signature scheme that harnesses the advantages of co-cyclic lattices.

3.2 The SQUIRRELS family

This thesis presents the SQUIRRELS family, a collection of lattice-based digital signature schemes for each of the five NIST security level requirements. These schemes adopt a hash-and-sign structure construction and rely on unstructured co-cyclic lattices as their foundation. At the core of these schemes is an integral matrix of dimensions $n \times n$, serving as a trapdoor sampling basis for the lattice, which we will require to be co-cyclic. This matrix consists of a set of n short and relatively orthogonal vectors, with constraints on their so-called Gram-Schmidt norm to ensure good sampling properties. On the other hand, a public basis, expressed in the Hermite Normal Form (HNF), is made available to enable the membership test for this lattice. Unlike NTRU lattices, the secret basis employed in SQUIRRELS cannot be easily compressed due to the absence of strong geometric properties. It is worth noting that this is a drawback inherent to any plain lattice scheme. Indeed, we are sampling our keys from a distribution that is computationally indistinguishable from the distribution of maximal entropy for the dimension and size involved.

The sampler used to produce signatures is the KleinSampler, already used in the original GPV proposal.

3.2.1 SQUIRRELS secret keys

Even though signatures generation still relies on the Klein sampler, the generation of the trapdoor differs significantly from earlier work, including the original GPV, or even from the more recent Falcon and Mitaka signatures. We start with the observation that the quality of signatures generated by the Klein sampler [28, 23] depends on the maximal norm of the Gram-Schmidt vectors of the trapdoor basis. It follows that we would like to generate trapdoors with such Gram-Schmidt norm as small as possible. However, as we fix the determinant both to address the fact that the problem is essentially scale-invariant and for efficiency purposes, we rather want trapdoor basis vectors to have their Gram-Schmidt norms to differ as little as possible from the geometric mean imposed by the determinant. To construct such a basis, we depart from the usual approach of sampling a trapdoor with random Gaussian vectors and testing its quality afterward, as in [23, 44]. Instead, our *key pair generation* sequentially samples secret vectors from regions of the space that are carefully crafted to provide a well-controlled Gram-Schmidt norm. These vectors should ideally be short and close to orthogonal for the best possible sampling quality, but not *too* short and orthogonal so as not to jeopardize security concerning key recovery attacks. Hence, at each step, the algorithm samples a vector of controlled norm close to the orthogonal subspace of the vector subspace spanned by the previous vectors. It finally selects the last vector in such a way that the resulting matrix matches the prescribed determinant.

3.2.2 Public key derivation

To derive an efficient public key, we require additionally that the sampled lattice has to be co-cyclic, i.e., we can find a vector $\mathbf{w} = (\mathbf{v}_{\text{check}}, -1)$ such that $\mathbf{c} \in \mathscr{L} \iff \langle \mathbf{c}, \mathbf{w} \rangle = 0 \bmod \Delta$ as seen in section 3.1. We choose a square-free determinant for efficiency and to enforce a practical form of the HNF with high probability.

The vector $\mathbf{v}_{\text{check}}$ can be computed from the row HNF of the secret basis. The HNF can be computed in polynomial time from any lattice and thus gives a basis of the lattice that is "as bad as can be". The public key of SQUIRRELS is chosen to be the vector $\mathbf{v}_{\text{check}}$.

3.2.3 Signature sampling

As explained previously, *signature generation* consists of first hashing the message to sign, along with a random nonce, into a vector $\mathbf{h}$, whose coefficients are uniformly mapped to

integers in the 0 to $q-1$ range. Then, the signer uses his knowledge of the secret lattice basis to produce a vector $\mathbf{c}$ belonging to the lattice that is close to $\mathbf{h}$. The signature $\mathbf{s}$ properly is $\mathbf{c}-\mathbf{h}$. Sampling a vector in the lattice (very) close to an arbitrary point is in general a hard problem, but here we rely on the fact that the secret basis is a basis of the lattice composed of short vectors. Klein's Gaussian sampler [28, 23] is used to efficiently sample this Gaussian vector.

3.2.4 Fast verification

The *verification* procedure first recomputes the hash of the message $\mathbf{h}$, and the lattice point $\mathbf{c}=\mathbf{s}+\mathbf{h}$. It verifies that $\mathbf{s}$ is a short vector and that $\mathbf{c}$ belongs to the lattice. Efficient membership checks are possible using the properties of co-cyclic lattices from section 2.1.1. We simply need to verify equation 3.2 using $\mathbf{v}_{\text{check}}$. Fixing the determinant of the lattice to a fixed product of primes allows to work modulo small primes, thanks to the Chinese remainder theorem, in the verification procedure, instead of modulo Δ, and makes the verification procedure more efficient.

3.3 Security considerations

To assess the concrete security of the SQUIRRELS scheme, we first prove that the GPV framework securely translates to co-cyclic lattices. Then, we proceed using the usual cryptanalytic methodology of estimating the complexity of the best attacks against *key recovery attacks* on the one hand, and *signature forgery* on the other. Evaluating concrete security from the best known attacks is the usual methodology as security reductions are not tight and give unpractical parameters.

We first give a quick discussion on the modelization of practical lattice reduction algorithms.

3.3.1 The GPV framework with co-cyclic lattices

As previously mentioned, the GPV framework is a versatile framework that can be instantiated to different classes of lattices. The underlying hardness assumptions however vary with this choice. In our case, SQUIRRELS relies on co-cyclic lattices instead of the uniform SIS-like lattices presented in [23] or NTRU lattices [44, 20].

The GPV framework constructs a signature scheme from a preimage samplable function f that is supposed to be collision resistant – we refer to [23] for a formal definition of this class of functions. The prototype of such a function is the SIS hash $\mathbf{e} \mapsto \mathbf{A} \cdot \mathbf{e} \bmod q$, where $\mathbf{A}$ is the public matrix of the scheme and $\mathbf{e}$ follows a Gaussian distribution of standard deviation s.

This can be adapted seamlessly to SQUIRRELS by taking:

$$\begin{aligned} f\colon D_n &\to \mathbb{Z}_\Delta \\ \mathbf{x} &\mapsto \mathbf{x} \cdot \mathbf{A}^T \bmod \Delta \end{aligned}$$

with $D_n = \{\mathbf{e} \in \mathbb{Z}^n \mid ||\mathbf{e}|| \leqslant \beta\}$, the input distribution of $\mathbf{e}$ is $D_{\mathbb{Z}^n, \beta/\sqrt{n}}$ and $\mathbf{A} = (\mathbf{v}_{\text{check}}, -1)$ the public key of SQUIRRELS. Lemma 5.2 of [23] adapts directly to prove that when $\mathbf{e}$ follows $D_{\mathbb{Z}^n, \beta/\sqrt{n}}$ with $\beta/\sqrt{n}$ large enough, $f(\mathbf{e})$ is statistically close to $\mathcal{U}(\mathbb{Z}_\Delta)$.

Then, we define the (samplable) inversion function $f^{-1}(u)$ for $u \in \mathbb{Z}_\Delta$ as follows: we chose via linear algebra a $\mathbf{t} \in \mathbb{Z}_\Delta^n$ such that $\mathbf{t} \cdot \mathbf{A}^T = u$, then sample $\mathbf{v}$ from $D_{\mathcal{L}, u, -\mathbf{t}}$ using the Klein sampler with the secret basis, and output $\mathbf{e} = \mathbf{t} + \mathbf{v}$.

Now the proof of theorem 5.9 of [23] translates in asserting that f forms a preimage samplable function under a variant of ISIS, the Group-SIS $\mathbf{GISIS}_{\mathbb{Z}_\Delta, \mathbf{c}, n, \beta}$ with cyclic group $\mathbb{Z}_\Delta$, for a uniform syndrome $u \in \mathbb{Z}_\Delta$. It is collision-resistant under the hardness of the corresponding $\mathbf{GSIS}_{\mathbb{Z}_\Delta, n, 2\cdot\beta}$ problem.

Regularity of the keygen output.

To fully follow the security proof of GLP, we hence only need to assume that the lattices sampled by our key generation algorithm "behaves as if" they were sampled randomly from the family above. More precisely, we make the following assumption:

- **SQR-PR$_\lambda$:** The public matrix $\mathbf{A} = (\mathbf{v}_{\text{check}}, -1)$ output by KeyGen(1^λ) is computationally indistinguishable from a uniformly random element of $\mathbb{Z}_\Delta^{n-1} \times \{-1\}$.

This assumption is very natural from the construction and is the exact non-structured analog of the NTRU assumption (the NTRU assumption over the ring $\mathbb{Z}$ would coincide exactly with our assumption as the normal form of the NTRU lattice would be exactly the Hermite form of the basis).

3.3.2 Heuristic modelization of lattice reduction, GSA and beyond

On the core-SVP model

To accurately assess the hardness of the underlying problems and ensure security in terms of bits, it is necessary to model the behavior of a practical oracle that approximates the Shortest Vector Problem (SVP). Our problems involve finding relatively short vectors in various lattices. For this purpose, we will employ the well-known (self-dual) Block Korkine-Zolotarev (BKZ) algorithm. The BKZ algorithm, with a block size denoted as B, may require a polynomial number of calls to an SVP oracle in dimension B, with a heuristic estimation of the number of calls being essentially linear. To account for potential future improvements in this reduction technique, we will only consider the cost of a single call to the SVP oracle. This conservative estimation is referred to as "core-SVP hardness." This cautious approach is motivated by the fact that there are methods to amortize the cost of SVP calls within BKZ, particularly when sieving is employed as the SVP oracle. Sieving is getting the de facto standard for larger cryptographic block sizes (we for instance refer to [6] for more details on the practical challenges raised by the use of sieving within lattice reduction).

Modelization of the output of reduced bases.

In all of the following and to ease the presentation, we follow the so-called *Geometric series assumption* (GSA), asserting that a reduced basis sees its Gram-Schmidt vectors' norm decrease with geometric decay. More formally, it can be instantiated as follows for self-dual BKZ (DBKZ) reduction algorithm of Micciancio and Walter [36]: an output basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ yielded by DBKZ algorithm with block size B on a lattice Λ of rank n satisfies

$$\|\tilde{\mathbf{b}}_i\| = \delta_B^{d-2(i-1)} \det(\Lambda)^{\frac{1}{n}}, \quad \text{where} \quad \delta_B = \left(\frac{(\pi B)^{\frac{1}{B}} \cdot B}{2\pi e} \right)^{\frac{1}{2(B-1)}},$$

for $\tilde{\mathbf{b}}_i$ being the i-th Gram Schmidt vector of the basis.

To obtain a more accurate estimation when computing actual figures, it is beneficial to enhance this analysis by employing the probabilistic simulation proposed in [16]. This simulation provides a more precise determination of the Block Korkine-Zolotarev (BKZ) block size B required for a successful attack, surpassing the coarse estimation based on the Geometric series assumption (GSA). By incorporating this probabilistic simulation, we can consider the widely recognized "quadratic tail" phenomenon of reduced bases [50], thereby improving the

precision of our calculations.

From lattice reduction blocksize to bitsec estimates.

This analysis translates into concrete bit-security estimates following the methodology of NEWHOPE [8] (so-called "core-SVP methodology"). In this model, the bit complexity of lattice sieving (which is asymptotically the best SVP oracle) is taken as $\lfloor 0.292B \rfloor$ in the classical [10] setting and $\lfloor 0.257B \rfloor$ in the quantum setting [11] in dimension B.

As the whole methodology is restated, we now turn to the fine-grained security of key recovery and then forgery.

3.3.3 Key Recovery attack

The key recovery attack aims at finding (at least) one of the short vectors of the secret basis, from the knowledge of the public key. A direct approach to key recovery is to do lattice reduction on a public basis, aiming at finding a relatively short vector in the spanned lattice: such attacks are addressed in Section 3.3.3.

Basic projection attack

This technique, initially described in the Falcon specification [44] and subsequently utilized in Mitaka [20] , operates by examining the lattice formed by the public basis, which is encoded in the public key as the Hermite Normal Form of the lattice in our scheme. It then finds vectors of the secret basis by listing all possible lattice vectors of norm less than $g_{\min}$. The attack avoids listing all lattice vectors in that sphere by restricting the search space to a projection.

More precisely, we fix B the block size of the DBKZ algorithm [36], and we first reduce the public basis using DBKZ to obtain a reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$. Then, we consider the lattice projected on $P = \text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-B-1})^{\perp}$. If we can find a projection of a secret vector in P, we can efficiently lift it to a vector of the target norm using *Babai Nearest Plane* algorithm [9]. Running a classical sieve (see [19] for instance) on P will list all vectors of norm smaller than $\sqrt{\frac{4}{3}}\ell$, where ℓ if the norm of the $n-B$-th Gram-Schmidt vector of the reduced basis. Under

the GSA assumption, we have:

$$\ell = \det(\Lambda)^{\frac{1}{n}} \cdot \delta_{2B+2-n} \approx \det(\Lambda)^{\frac{1}{n}} \cdot \left(\frac{B}{2\pi e}\right)^{1-\frac{n}{2B}}$$

Assuming that secret vectors behave as random vectors of norm $g_{\min}$, their projection on P is roughly:

$$\|\pi_P(\mathbf{b}_i)\| = \sqrt{\frac{B}{n}} \cdot g_{\min}$$

Thus, we will retrieve the projection among the sieved vectors if $\sqrt{\frac{B}{n}} \cdot g_{\min} \leqslant \sqrt{\frac{4}{3}}\ell$, that is if the following condition is fulfilled:

$$g_{\min} \leqslant \sqrt{\frac{4n}{3B}} \cdot \det(\Lambda)^{\frac{1}{n}} \left(\frac{B}{2\pi e}\right)^{1-\frac{n}{2B}}. \tag{3.3}$$

Remark.

- *This approach is similar to the one used in the security evaluation of [8] , but we use all the vectors given by the last step of sieving, resulting in a slightly stronger attack and as such more conservative parameters choices.*
- *(On the size of the enumeration window.) In the previous description we only considered the space $\mathscr{P}$, orthogonal to* $\mathrm{span}(b_1,\ldots,b_{2d-B-1})$*. It is natural to want to extend its dimension and choose the optimal one. It appears that for the specific parameters of our work, this optimization would only result in a difference of less than a single bit of security. Besides, on the one hand, by using the exact block size beta we can extract the vectors we need to sieve for free from the preliminary run of DBKZ, avoiding the need for an additional sieving pass. On the other hand, using a larger dimension for the additional sieving pass adds a non-negligible cost. Note that this is a consequence of the Core-SVP methodology which ignores the polynomial overhead cost of (D)BKZ.*

3.3.4 Signature forgery by BDD reduction.

As a Hash-and-Sign paradigm signature, forging a signature stems from feeding a lattice point $\mathbf{v}$ at a bounded distance from a random space point $\mathbf{x}$ (in practice which is actually $(H(r\|m))$. This bounded distance decoding (BDD) problem can be solved using the so-called *Nearest-Cospace* framework developed in [21]. Under the Geometric Series assumption, Theorem 3.3 of [21] states that under the condition: $\|\mathbf{x}-\mathbf{v}\| \leqslant \left(\delta_B^n \det(\Lambda)^{\frac{1}{n}}\right)$, the decoding can be done in time $\mathrm{Poly}(n)$ calls to a CVP oracle in dimension B.

Remark. *On the contrary to Falcon and Mitaka, we manage to reduce the security gap between forgery and key recovery to only a few bits (even less than 1 for* SQUIRRELS*-128), thanks to the flexibility of the choice of parameters.*

3.4 Advantages and limitations

This new signature scheme presents advantages and limitations in comparison to the existing literature. It is important to be aware of them and be transparent so that application developpers can choose the most adapted primitives to their needs. This section tries to objectively list the main characteristics of SQUIRRELS to take into account during such a decision process.

3.4.1 Advantages

Confidence in unstructured lattices. One concern about Falcon is its use of NTRU lattices which might be vulnerable to specialized and more powerful attacks than the generic attacks on lattices. Our scheme samples lattices with no strong geometric property and bases its security on generic lattice problems. These problems have been studied for decades, notably with average-to-worst-case reductions, which give strong confidence in their security.

Compact signatures. Despite leveraging an unstructured problem, our scheme still manages to generate remarkably compact signatures. The byte-size of our signatures falls within the range of Falcon and Dilithium, both of which are renowned for producing concise signatures in the post-quantum setting.

Efficient signature generation and verification. SQUIRRELS is also very competitive in terms of signature generation and verification efficiency. On a personal laptop, it is capable of generating several dozens to hundreds of signatures per second, while also verifying thousands of signatures within a single second.

Simple signature verification. The signature verification process is remarkably straightforward, primarily consisting of a hash computation and the verification of a *single linear equation*. Its simplicity streamlines the verification procedure without compromising the security of the scheme.

3.4.2 Limitations

Slow key generation. One aspect that warrants consideration is the relatively slow key generation procedure employed in our scheme. This process involves computationally expensive operations on high-dimensional matrices, such as determinant calculations and HNF (Hermite Normal Form) computations. Consequently depending on the target hardware and security level desired, generating a single keypair can take anywhere several dozen seconds. The acceptability of this duration depends on the specific application at hand. However, it may prove to be prohibitive if the application necessitates a high frequency of rotation of signature keys. This is a direct consequence of the choice of using unstructured lattices: every non-trivial linear algebra operation is at the very least quadratic in the dimension of the lattice.

Large public keys. Due to the absence of structure in the lattices we sample, it is not possible to compress the public key in a manner similar to Falcon. As a result, our public keys are larger by a factor of $\mathcal{O}(n)$, weighing several hundred kilobytes to a few megabytes. This increase in size should be taken into consideration, particularly when storage or transmission constraints are significant factors in the system's design. Once again, this can not be improved when dealing with unstructured lattices, as the entropy of the matrix representing the keys is essentially maximal for their size and dimensions.

Floating-point arithmetic. It is important to note that our signature scheme utilizes floating-point arithmetic during both key generation and signature generation procedures. While this choice contributes to the scheme's effectiveness, it may pose a significant limitation when implementing the scheme on very constrained devices with limited computational capabilities or limited support for floating-point operations. Careful consideration must be given to the feasibility and the practicality of implementing our scheme in such environments. We stress that verification, on the other hand, does not rely on floating point arithmetic, so the more common use case where signatures only need to be *verified* on constrained devices (e.g., bootloader signing) is supported without issue.

Chapter 4

Implementation

This section covers some of the implementation details of SQUIRRELS. We focus on the main details of the key generation procedure, as the signature generation and verification are less novel and described from a high level in subsections 3.2.3 and 3.2.4. For extended details, we refer to the specification of SQUIRRELS [2] submitted to NIST. It can be found at https://squirrels-pqc.org/squirrels-spec-v1.0.pdf.

4.1 Public parameters

SQUIRRELS uses public parameters in its algorithms:

1. n the dimension of the lattices sampled.
2. Bounds $g_{\min} < g_{\max}$ on the Gram-Schmidt norms accepted during the key generation after rounding the vectors.
3. Bounds $g_{0,\min} < g_{0,\max}$ on the norms of the vectors sampled in key generation, before rounding.
4. Bound e_δ controls the distance to the target determinant of the sampled basis at each step of the key generation.
5. A target determinant $\Delta = \prod_{p \in P_\Delta} p$. With P_Δ a set of primes in $[2^{30}, 2^{31}]$. We also define $l_{\text{det}} = \frac{\log(\Delta)}{n}$, which corresponds to the target Gram-Schmidt norm.

6. A bound $q \in \mathbb{N}^*$ on coefficients of hashed messages during signature generation. Messages are hashed in $[0, q-1]^n$.
7. A real bound $\lfloor \beta^2 \rfloor > 0$ on the square norm of signatures.
8. Standard deviations σ and $\sigma_{\min} < \sigma_{\max}$ used in Klein's sampler.
9. Integers sig_{size} and sig_{rate} used to compress the signatures.

4.2 Key pair generation

Keypair generation splits into three main steps:

1. First, we generate the first $n-1$ vectors of the secret basis.
2. Then, we compute the last secret basis vector so that the basis has the target determinant.
3. Finally we compute the row HNF of the secret basis and derive the public key from it.

During the whole process, we must carefully control the norms of the Gram-Schmidt vectors, and the expected distance to the target determinant so that the norm of the last vector is also bounded.

This procedure is described in Keygen.

4.2.1 Generation of the first vectors

The first step generates $n-1$ vectors, with Gram-Schmidt norms between $g_{\min}$ and $g_{\max}$. Additionally, to bound the norm of the last Gram-Schmidt vector, as we have $||\tilde{\mathbf{b}}_n|| = \frac{\Delta}{\prod_{1 \leqslant i \leqslant n-1} ||\tilde{\mathbf{b}}_i||}$, we control the value $\delta := \sum_{1 \leqslant j < i} \log(||\tilde{\mathbf{b}}_j||) - \frac{\log(\Delta)}{n} \cdot (i-1)$ at each step i so that it remains small in absolute value.

This procedure works sequentially and at each step, it samples a candidate vector $\mathbf{v} = \mathbf{v_B} + \mathbf{v_{B^\perp}}$ verifying:

- $\mathbf{v_{B^\perp}}$ is uniformly distributed among the vectors of $\mathbf{B}^\perp$ with a norm bounded between b_{low} and b_{up}.

Algorithm 3 Keygen()

Ensure: A secret key sk, a public key pk
  while True **do**
    $\mathbf{B} \leftarrow$ GenVectors$(n-1)$ ▷ Sample the $n-1$ first vectors of the basis
    $\mathbf{v}_{\text{last}} \leftarrow$ ComputeLastVector$(\mathbf{B})$ ▷ $\mathbf{v}_{\text{last}}$ such that $\det(\mathbf{B} \cup \{\mathbf{v}_{\text{last}}\}) = \Delta$
    if $\mathbf{v}_{\text{last}} = \perp$ **then**
      continue
    end if
    $\text{sk} \leftarrow \mathbf{B} \cup \{\mathbf{v}_{\text{last}}\}$
    $\text{pk} \leftarrow$ ComputePK(sk) ▷ Verify the lattice co-cyclicity, and derive pk
    if $\text{pk} = \perp$ **then**
      continue ▷ The sampled lattice is not co-cyclic
    end if
    return sk, pk
  end while

- $\mathbf{v_B}$ follows a Gaussian distribution in $\mathbf{B}$ of standard deviation $\frac{g_{\max}}{\sqrt{n}}$.

The vector $\mathbf{v}$ is then rounded to an integral vector by rounding each of its coordinates. This rounding introduces an error on the norm of the resulting Gram-Schmidt vector, but parameters are chosen so that when sampling in $[g_{0,min}, g_{0,\max}]$, the Gram-Schmidt vector after rounding will have a norm in $[g_{\min}, g_{\max}]$ with probability at least 90%. Thus, by default, we take $b_{\text{low}} = g_{0,\min}$ and $b_{\text{up}} = g_{0,\max}$ and reject vectors if their Gram-Schmidt norm is not in $[g_{\min}, g_{\max}]$ after rounding.

As noted before, we also control the distance to the target determinant at each step. At step i, we define the drift $\delta = \left(\sum_{1 \leqslant j < i} \log(||\tilde{\mathbf{b}}_j||)\right) - l_{\text{det}} \cdot (i-1)$. If $\delta > e_\delta$, then we update $b_{\text{up}} = \frac{b_{\text{up}} + 3 \cdot b_{\text{low}}}{4}$. If $\delta < -e_\delta$, then we update $b_{\text{low}} = \frac{3 \cdot b_{\text{up}} + b_{\text{low}}}{4}$.

This leads to algorithm 4.

4.2.2 Computation of the last secret vector

Once we have the $n-1$ first vectors, we determine a vector $\mathbf{v}_{\text{last}}$ such that the determinant of the extended basis is Δ and with a Gram-Schmidt norm in $[g_{\min}, g_{\max}]$.

We recall that given a matrix $\mathbf{B} \in \mathbf{R}^{n \times n}$, we can expand its determinant on its last row using

Algorithm 4 GenVectors(k)

Require: A number k of vectors to generate
Ensure: k vectors, with Gram-Schmidt norms between $g_{\min}$ and $g_{\max}$

```
B ← []                                              ▷ Contains the sampled basis
B̃ ← []                                              ▷ Contains the Gram-Schmidt orthogonalization of B
for i = 1, ..., k do
    while True do
        δ ← (Σ_{1⩽j<i} log(||b̃_j||)) − l_det · (i − 1)          ▷ Compute the drift
        b_up ← g_{0,max}
        b_low ← g_{0,min}
        if δ > e_δ then
            b_up ← (b_up + 3·b_low) / 4
        end if
        if δ < −e_δ then
            b_low ← (3·b_up + b_low) / 4
        end if

        c ← SampleOrthogonal(B, b_low, b_up)
        v ← Round(c)                                ▷ Round each coordinate (round-to-nearest-even)
        ṽ ← GramSchmidt(B, v)
        if g_min ⩽ ||ṽ|| ⩽ g_max then
            B ← B ∪ {v}
            B̃ ← B̃ ∪ {ṽ}
            break
        end if
    end while
end for
return B
```

Algorithm 5 SampleOrthogonal$(\mathbf{B}, b_{\text{low}}, b_{\text{up}})$

Require: $\mathbf{B}$ a set of ℓ independent vectors, $b_{\text{low}} < b_{\text{up}}$ two bounds
Ensure: A vector $\mathbf{v} = \mathbf{v_B} + \mathbf{v_{B^\perp}}$ such that $\mathbf{v_B}$ is a Gaussian vector with mean 0 and standard deviation $\frac{g_{\max}}{\sqrt{n}}$, and $\mathbf{v_{B^\perp}} \in \mathbf{B}^\perp$ uniform verifying $b_{\text{low}} \leqslant ||\mathbf{v_{B^\perp}}|| \leqslant b_{\text{up}}$

$\mathbf{v} \leftarrow$ SampleNormal$(g_{\max}/\sqrt{n}, n)$ ▷ Sample a Gaussian vector
$\mathbf{v_{B^\perp}} \leftarrow$ GramSchmidt$(\mathbf{B}, \mathbf{v})$ ▷ First, sample a direction in $\mathbf{B}^\perp$
$\mathbf{v_B} \leftarrow \mathbf{v} - \mathbf{v_{B^\perp}}$

$x \leftarrow$ a random double uniform in $[0, 1]$
$r \leftarrow b_{\text{low}} \cdot \left(x \cdot \left(\left(\frac{b_{\text{up}}}{b_{\text{low}}}\right)^{n-\ell} - 1\right) + 1\right)^{1/(n-\ell)}$ ▷ Sample a target norm
return $\mathbf{v_B} + \frac{\mathbf{v_{B^\perp}}}{||\mathbf{v_{B^\perp}}||} \cdot r$

minors:

$$\det(\mathbf{B}) = \sum_{1 \leqslant i \leqslant n} (-1)^{i+1} \mathrm{B}_{n,i} \cdot \mathsf{Minor}_{n,i}(\mathbf{B})$$

where $\mathsf{Minor}_{n,i}(\mathbf{B})$ is the determinant of the matrix $\mathbf{B}$ where we removed line n and column i.

If we can find a set of $\mathsf{Minor}_{n,i}$ which are co-prime, using Euclid's extended algorithm we can find c_i such that $\sum_{1 \leqslant i \leqslant n} c_i \cdot \mathsf{Minor}_{n,i} = 1$. Multiplying by Δ gives us a relation $\sum_{1 \leqslant i \leqslant n} c'_i \cdot \mathsf{Minor}_{n,i} = \Delta$. We can then simply take $\mathbf{v}_{\text{last}} = ((-1)^{i+1} c'_i)_{1 \leqslant i \leqslant n}$.

Remark. *For efficiency, we compute only the last 4 minors:* $(\mathsf{Minor}_{n,n+1-i})_{1 \leqslant i \leqslant 4}$. *We evaluated that there are co-prime with a probability close to 42%, so we can restart the key generation in case they are not without affecting the scheme security. In practice, the speedup gained by only computing* 4 *minors instead of* $n-1$ *is non-negligible, even with the restarts (about 2 on average).*

We also want the vector $\mathbf{v}_{\text{last}}$ to have small coefficients for efficiency so we need to reduce it:

1. we reduce the c'_i using the algorithm ComputeReducedXGCD from [34] so that their absolute value is lower than $\max((|\mathsf{Minor}_{n,n+1-i}|/2)_{1 \leqslant i \leqslant 4}, \Delta)$.
2. The coefficients c'_i are further improved by LLL reducing a matrix containing the c'_i and relationships $m_i / \gcd(m_i, m_j), -m_j / \gcd(m_i, m_j)$ which keep the gcd of the c'_i constant. In practice, fpLLL [18] is used for this step.

Remark. *The use of LLL in this step of the key generation is not strictly necessary to sample lattices of the desired shape, but it speeds up the computation of the last vector.*

3. we apply Babai Nearest Plane algorithm [9] to $(0,\dots,0,c'_4,-c'_3,c'_2,-c'_1)$ to reduce the part of the vector in $\mathbf{B}$. The part of the vector in $\mathbf{B}^{\perp}$ is guaranteed to be small by the control of the drift δ in the GenVectors algorithm.

The pseudo-code of this procedure is in Algorithm 6.

Algorithm 6 ComputeLastVector($\mathbf{B}$)

Ensure: vector $\mathbf{v}_{\text{last}}$, with Gram-Schmidt norm between $g_{\min}$ and $g_{\max}$, and such that $\mathbf{B}\cup\{\mathbf{v}_{\text{last}}\}$ has determinant Δ.
  $m \leftarrow []$
  for $k = 1,\dots,4$ **do**
    $m \leftarrow m \cup \{\det((\mathrm{B}_{i,j})_{i\in[1,n-1],j\in[1,n]\setminus\{n+1-k\}})\}$ ▷ Computes $\mathsf{Minor}_{n,k}(\mathbf{B})$
  end for
  if $\gcd(m) \neq 1$ **then**
    return $\perp$ ▷ We won't be able to find a combination of the minors equal to Δ
  end if
  $c' \leftarrow \mathsf{ComputeReducedXGCD}(m,\Delta)$ ▷ Algorithm from [34]
  $c' \leftarrow \mathsf{ReduceWithLLL}(c')$

  $\mathbf{v}_{\text{last}} \leftarrow (0,\dots,0,\mathrm{c}'_4,-\mathrm{c}'_3,\mathrm{c}'_2,-\mathrm{c}'_1)$
  return $\mathbf{v}_{\text{last}} - \mathsf{NearestPlane}(\mathbf{B},\mathbf{v}_{\text{last}})$ ▷ Reduce $\mathbf{v}_{\text{last}}$ with Babai Nearest Plane

4.3 Interplay between parameters

To instantiate our signature scheme, we need to define concrete sets of parameters. We describe below relations and constraints on our parameters to ensure the correctness and security of our scheme.

The number of queries Q_s, targeted security level λ. The first parameters are the maximal number of signing queries Q_s, and the targeted security level λ. According to [1], we take $Q_s = 2^{64}$, and:

$\lambda = 128$ for NIST levels I and II
$\lambda = 192$ for NIST levels III and IV
$\lambda = 256$ for NIST level V

Bounds on sampled norms $g_{0,\min}$ and $g_{0,\max}$, and lattice determinant Δ. We wish to have the bounds $g_{0,\min}$ and $g_{0,\max}$ as close to each other as the greater the lower bound, the greater the key recovery security, and as the lower the upper bound, the lower the length of signatures. We are however constrained by the correction of the determinant drift: we need to ensure we can sample vectors with a norm smaller or larger than $\sqrt[n]{\Delta}$ as desired.

To correct the drift δ in the GenVectors, we sample vectors with a norm lower than $\frac{3\cdot g_{0,\min}+g_{0,\max}}{4}$ to reduce the drift, or higher than $\frac{g_{0,\min}+3\cdot g_{0,\max}}{4}$ to increase the drift. For this correction to be effective, we enforce that after sampling two-thirds of the vectors the parameters verified:

$$\frac{3\cdot g_{0,\min}+g_{0,\max}}{4}+e_{round}<\sqrt[n]{\Delta}<\frac{g_{0,\min}+3\cdot g_{0,\max}}{4}-0.5$$

where e_{round} is an upper bound on the rounding error after sampling two-thirds of the bases.
We chose as a determinant a product of large primes verifying the above inequalities.

Bounds on accepted Gram-Schmidt norms $g_{\min}$ and $g_{\max}$. These bounds extend the interval $[g_{0,\min}, g_{0,\max}]$ to take into account the rounding error and accept rounded vectors with high probability. Knowing these bounds in advance allows us to have a more precise security analysis of our scheme.

Standard deviation σ of the signatures. Signatures follow a discrete Gaussian distribution and are sampled using Klein's sampler. To ensure we lose at most $O(1)$ bits of security by using this sampler instead of a perfect distribution, it suffices to take $\varepsilon \leqslant 1/\sqrt{Q_s\cdot\lambda}$, and:

$$\sigma=\frac{1}{\pi}\cdot\sqrt{\frac{\log(2n(1+1/\varepsilon))}{2}}\cdot\underbrace{g_{\max}}_{\geqslant\|\mathbf{B}\|_{\mathrm{GS}}}$$

Maximal norm β of the signatures. Signatures have an expected norm of $\sqrt{n}\cdot\sigma$. We reject too large signatures by using a tail-cut τ_{sig} i.e. we reject signatures of norm larger than $\beta=\tau_{\mathrm{sig}}\cdot\sqrt{n}\cdot\sigma$.

We take $\tau_{\mathrm{sig}}=1.1$. Lemma 4.4 of [31] ensures that rejection happens only with a small probability.

Signature byte-length $\mathrm{sig}_{\mathrm{size}}$ and $\mathrm{sig}_{\mathrm{rate}}$. Given a $\mathrm{sig}_{\mathrm{rate}}$, we evaluate the corresponding

sig_{size} as the average compressed signature size of vectors of norms β. We choose the $\text{sig}_{\text{rate}} \in \{4,5,6,7\}$ giving the smallest sig_{size}.

Chapter 5

Evaluation

The previous sections defined the theoretical framework of SQUIRRELS, and constraints on the parameter set. In this section, we give details about its concrete instantiation and evaluation. Full parameters are included in SQUIRRELS submission package at https://squirrels-pqc.org/.

5.1 Concrete parameters

We decided to publish one set of parameters for each NIST security level. We selected them by exploring a grid of parameters respecting the constraints from section 4.3, and keeping the ones minimizing the signature size.

This selection process is completely reproducible using the script from the NIST specification `Supporting_Documentation/additional/params.py`, notably the determinant is computed using a prime random generator with a fixed seed. We give below a table summarizing the concrete security of these sets of parameters and public key and signature size. We again refer to SQUIRRELS specification for the full sets of parameters.

		SQUIRRELS-I	SQUIRRELS-II	SQUIRRELS-III	SQUIRRELS-IV	SQUIRRELS-V
Target NIST Level		I	II	III	IV	V
Lattice dimension n		1034	1164	1556	1718	2056
Key-recovery:	BKZ block-size B	433	491	666	736	887
	Core-SVP hardness (C)	126	143	194	214	259
	Core-SVP hardness (Q)	114	130	176	195	235
Forgery:	BKZ block-size B	431	499	709	784	968
	Core-SVP hardness (C)	125	145	207	228	282
	Core-SVP hardness (Q)	114	132	187	207	256
Public key byte-length		681 780	874 576	1 629 640	1 888 700	2 786 580
Signature byte-length		1 019	1 147	1 554	1 676	2 025

5.2 Description of the Reference Implementation

The submission package includes a reference implementation written exclusively in portable C, supporting all five security levels and adapted using a compilation flag.

These implementations use dynamically linked libraries, used only in the key generation for big integer and matrix computations:

- GMP [48]
- Flint [47]
- fplll [18]

All the computations in the key generation requiring big integers or matrix manipulations are performed using Flint structures, this includes notably computations of matrix determinants and HNF. The reference implementation also includes an efficient function to compute several minors at a time inspired by the DoubleDet algorithm from [40].

In the Verify procedure, we note that the variable sum always fits on 64-bits signed integers:

- $v_{\text{check},i} \bmod p$ is in $[0, 2^{31})$
- $c_i = s_i + h_i \in [-4096, 8192)$ as $|s_i| \leqslant \beta < 4096$ and $h_i \in [0, q)$ (in case $|s_i| > \beta$ the result is rejected later during norm checking in the procedure).

- there are $n < 4096 = 2^{12}$ summations of products $c_i \cdot (v_{\text{check},i} \bmod p)$

So we need no more than $31 + 13 + 12 = 57$ bits to store the variable sum, plus one bit for the sign. For efficiency, we thus directly compute it on a 64-bit signed integer and reduce it modulo reduction p only once.

Due to their large or varying size, many structures of our implementations go on the heap.

5.3 Test vectors

The submission package includes test vectors – a keypair along a signature generated from a given seed – which allows one to verify that our implementation is portable and to verify that their own implementation works as expected.

5.4 Performance on the NIST x64 Reference Target

In this section, we summarize our performance evaluation on a 2018 laptop Lenovo Y530, equipped with Intel Core i5-8300H (8 CPU threads at 2.3GHz), 32 GB of physical RAM and running Manjaro 22.1.

The benchmark program is compiled with GCC version 12.2.1 with flags `-O3 -march=native -Ofast` and calls the NIST API `crypto_sign_keypair()`, `crypto_sign()`, and `crypto_sign_open()`. We then ran it on one CPU thread. Execution time was measured with `clock` POSIX calls, number of cycles was measured with `rdtsc` instruction.

	keygen		sign		vrfy	
	seconds	RAM (kB)	cycles/sign	sign/s	cycles/vrfy	vrfy/s
SQUIRRELS-I	74	189 776	4 230 444	545.1	201 737	11429.8
SQUIRRELS-II	120	236 356	5 305 045	434.8	244 573	9427.3
SQUIRRELS-III	306	410 152	8 977 892	257.2	430 609	5375.5
SQUIRRELS-IV	425	433 164	10 693 130	215.6	481 334	4789.2
SQUIRRELS-V	1175	626 824	41 512 206	55.6	1 048 477	2207.4

5.5 Performance on x64 AMD

In this section, we evaluate the performance on a 2022 Lenovo Thinkpad P14s equipped with a Ryzen Pro 7 5850U (16CPU threads at 3GHz), boost disabled, and running Manjaro 22.1.

The benchmark program was compiled with GCC version 12.2.1 with flags `-O3 -Ofast -march=native` and ran on one CPU thread.

	keygen (s)	sign		vrfy	
		cycles/sign	sign/s	cycles/vrfy	vrfy/s
SQUIRRELS-I	34	3 164 772	601.6	145 351	13099.4
SQUIRRELS-II	52	3 732 387	509.0	159 887	11871.9
SQUIRRELS-III	127	7 139 278	266.2	287 974	6594.4
SQUIRRELS-IV	179	9 097 631	208.7	329 066	5765.5
SQUIRRELS-V	351	10 670 614	177.9	481 938	3937.5

Chapter 6

Related Work

There has been intense work in the field of post-quantum cryptography, and lattice-based signature schemes specifically. The two NIST finalists Falcon [44] and Dilithium [32] are good points of comparison for our scheme.

Comparison with Falcon. Falcon is also based on the GPV framework and a Gaussian sampler with similar quality to the Klein sampler. Its high-level design is thus very similar to the one of SQUIRRELS. However, Falcon uses NTRU lattices which allows it to drastically compress its keys (wins a factor $\mathcal{O}(n)$), and cut in half the size of its signatures thanks to the additional algebraic structure.

Falcon achieves a signature size of 666 bytes at level I, while SQUIRRELS achieves 1019 bytes. Interestingly that is less than twice the size of Falcon's signatures. This can be explained by the production of smaller Gram-Schmidt secret vectors in SQUIRRELS compared to Falcon, which allows it to compensate in part for the lack of structure of its lattices.

In terms of performance, SQUIRRELS achieves signature generation and verification about 10 times slower than Falcon, which still makes it a very competitive competitor as Falcon is known for being an extremely fast signature scheme.

Comparison with Dilithium. Dilithium followed a quite different design path and is based on Fiat-Shamir with rejection sampling. Its security relies on hardness problems over module lattices and LWE. These structured lattices allow it to also have a small public key. However, its signature size is at minimum 2420 bytes which is significantly larger than what we achieve with SQUIRRELS. The signature generation of SQUIRRELS is about 20 times slower than

the one from Dilithium, and its verification is about 3 times slower. This again shows that SQUIRRELS is still competitive as Dilithium is also a very fast scheme.

SQUIRRELS however has a much larger public key size than Falcon and Dilithium (600kB at level I, against 1-2kB) and a slow keypair generation. This is due to the use of unstructured lattices in our scheme. This reduces the applicability of our scheme, but we believe that SQUIRRELS still offers great security guarantees that may be worth the trade-off for critical applications, or long-term signature keys pre-delivered with an OS for instance.

Our signature scheme also compares favorably to other schemes based on unstructured problems such as Wave [17] which obtains signatures of 13kB and public keys of 3000kB. We can also mention qTESLA [7] which relies on a plain version of LWE, achieves 14kB public key size, and 2.5kB signature size.

Hence, our scheme is competitive with schemes based on both structured and unstructured problems. The size of its public key limits its usage to long-term or very critical use, but we believe this is an interesting trade-off that was not achievable with the existing literature.

Chapter 7

Conclusion

In this thesis, we presented SQUIRRELS, a novel digital signature scheme based on plain lattice hardness assumptions. It instantiates the GPV framework and optimizes its key generation to generate short Gram-Schmidt vectors and sample co-cyclic lattices. This allows our scheme to generate short signatures, and to have a very fast verification despite the lack of structure. We showed that it has a signature size and performance competitive with the two lattice-based NIST finalists Falcon and Dilithium, with a signature size of 1019 bytes at NIST level I, and signature generation and verification of about 10 times slower.

We performed an extended analysis of the security of our scheme with a proof that the GPV framework adapts to co-cyclic lattices, and evaluation of SQUIRRELS bit security against key recovery and forgery using standard methodology. We defined 5 parameter sets, one for each security level defined by NIST. Test vectors and reproducible scripts are included in the package submitted to NIST which allows one to reproduce and verify our parameters selection.

SQUIRRELS has a larger public key of about 600kB at level I, but it still represents an interesting and competitive trade-off compared to the existing literature. SQUIRRELS provides strong security guarantees while providing short signatures. We believe its use is especially interesting in critical applications, or applications using long-term signature keys.

Bibliography

[1] Call for Proposals - Post-Quantum Cryptography: Digital Signature Schemes | CSRC | CSRC — csrc.nist.gov. https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals.

[2] Squirrels: Square Unstructured Integer Euclidean Lattice Signature. https://squirrels-pqc.org/.

[3] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[4] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9. Springer, Heidelberg, July 1999.

[5] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th ACM STOC*, pages 284–293. ACM Press, May 1997.

[6] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 717–746. Springer, Heidelberg, May 2019.

[7] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Krämer, Patrick Longa, and Jefferson E. Ricardini. The lattice-based digital signature scheme qTESLA. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part I*, volume 12146 of *LNCS*, pages 441–460. Springer, Heidelberg, October 2020.

[8] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 327–343. USENIX Association, August 2016.

[9] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[10] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24. ACM-SIAM, January 2016.

[11] André Chailloux and Johanna Loyer. Lattice sieving via quantum random walks. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 63–91. Springer, Heidelberg, December 2021.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Heidelberg, December 2017.

[13] David A. Cooper, Daniel C. Apon, Quynh H. Dang, Michael S. Davidson, Morris J. Dworkin, and Carl A. Miller. Recommendation for Stateful Hash-Based Signature Schemes. NIST Special Publication SP 800-208, October 2020.

[14] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 559–585. Springer, Heidelberg, May 2016.

[15] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 324–348. Springer, Heidelberg, April / May 2017.

[16] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Heidelberg, August 2020.

[17] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51. Springer, Heidelberg, December 2019.

[18] The FPLLL development team. fplll, a lattice reduction library, Version: 5.4.4. Available at https://github.com/fplll/fplll, 2023.

[19] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 125–145. Springer, Heidelberg, April / May 2018.

[20] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Heidelberg, May / June 2022.

[21] Thomas Espitau and Paul Kirchner. The nearest-colattice algorithm. Cryptology ePrint Archive, Report 2020/694, 2020. https://eprint.iacr.org/2020/694.

[22] Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 528–558. Springer, Heidelberg, May 2016.

[23] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[24] Elizabeth Gibney. Hello quantum world! Google publishes landmark quantum supremacy claim. *Nature*, 574(7779):461–462, October 2019.

[25] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 112–131. Springer, Heidelberg, August 1997.

[26] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*, pages 267–288. Springer, Heidelberg, June 1998.

[27] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS$^+$. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[28] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *11th SODA*, pages 937–941. ACM-SIAM, January 2000.

[29] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Cryptology ePrint Archive, Report 2012/090, 2012. https://eprint.iacr.org/2012/090.

[30] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.

[31] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.

[32] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[33] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

[34] Bohdan S. Majewski and George Havas. The complexity of greatest common divisor computations. In Leonard M. Adleman and Ming-Deh Huang, editors, *Algorithmic Number Theory*, pages 184–193, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[35] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[36] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 820–849. Springer, Heidelberg, May 2016.

[37] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[38] Phong Q. Nguyen and Igor E. Shparlinski. Counting Co-Cyclic Lattices. *SIAM Journal on Discrete Mathematics*, 30(3):1358–1370, July 2016.

[39] Azaria Paz and Claus-Peter Schnorr. Approximating integer lattices by lattices with cyclic factor groups. In *International Colloquium on Automata, Languages and Programming*, 1987.

[40] Clément Pernet and William Stein. Fast computation of Hermite normal forms of random integer matrices. *Journal of Number Theory*, 130(7):1675–1683, 2010.

[41] Thomas Pöppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[42] Thomas Prest. *Gaussian Sampling in Lattice-Based Cryptography*. PhD thesis, École Normale Supérieure, 2015.

[43] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 347–374. Springer, Heidelberg, December 2017.

[44] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[45] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[46] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[47] The FLINT team. *FLINT: Fast Library for Number Theory*, 2023. Version 2.9.0, https://flintlib.org.

[48] The GMP team. *GNU Multiple Precision Arithmetic Library*, 2023. Version 6.2.1, https://gmplib.org/.

[49] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2009/616, 2009. https://eprint.iacr.org/2009/616.

[50] Yang Yu and Léo Ducas. Second order statistical behavior of LLL and BKZ. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017*, volume 10719 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2017.