Report on evaluation of KpqC candidates

Jolijn Cottaar, Kathrin Hövelmanns, Andreas Hülsing, Tanja Lange, Mairon Mahzoun, Alex Pellegrini, Alberto Ravagnani, Sven Schäge, Monika Trimoska, and Benne de Weger

Eindhoven University of Technology

31 October 2023

Contents

Chapter 1

Introduction

This report covers our work on the submissions to the KpqC competition. We analyzed all submitted KEMs and signature systems. For those that withstood cryptanalysis, we checked the security proofs to the extent that any were given and checked for obstacles to efficient implementation.
We found attacks against IPCC, Layered ROLLO-I, REDOG, AIMer, and MQ-Sign. The attack on IPCC was the first attack posted to the KpqC forum, we also broke the updated version, IPCC7, submitted in July. For Layered ROLLO-I we broke the second, third, and fourth version of the submission. For REDOG we found three different issues with the submission but also proposed ways to fix it. For AIMer we showed a weakness in the underlying block cipher AIM. For MQ-Sign we were the first to announce an attack on the sparse versions; another group then announced a follow-up attack. This report includes the first mention of an attack on the third version which we will announce soon.
For Enhanced pqsigRM and Peregrine other teams were first to announce attacks but did not post any writeup; we reconstructed and confirmed these attacks from the available information. Shortly before we delivered this report, a preprint detailing the attack on Peregrine became available. For GCKsign two serious concerns were pointed out by two other teams that the designers had not taken into account and ended up not fixinng.
For the other schemes we analyzed the complexity of generic attacks and identified gaps in the security proofs (if provided).

Chapter 2

Background on lattices

Multiple schemes submitted to the Korean Post Quantum competition are based on some form of the Learning with Errors (LWE) problem or some of its variants, including Ring (RLWE), Module (MLWE), Learning with Rounding (LWR or M/RLWR) and NTRU. Specifically we have SMAUG (MLWE), TiGER (RLWE, RLWR), HAETAE (MLWE), NCC-sign (RLWE), NTRU+ (NTRU) and SOLMAE (NTRU). Also the broken schemes GCKsign and Peregrine are based on lattices.
In this chapter we first cover generic lattice attacks and then intrduce the most common problems and tools in security proofs. For systems not based on lattices ths is given as part of the chapter introducing the system and short intros are given also for the lattice schemes to keep the chapters concise.

2.1 Generic lattice attacks

Each of these schemes have their own intricacies, but to each the generic lattice attacks do apply. Specifically we will be looking here at the BKZ lattice reduction algorithm [SE91]. There are quite a few differing opinions on how to properly estimate the costs of this attack, which can result in differing security categories. This all depends on the cost model used, i.e., the cost of solving the Shortest Vector Problem (SVP) in dimension β (by using sieving or enumeration) and the number of SVP oracle calls needed for the BKZ.
All the schemes, no matter the form of the problem, will be reduced to an LWE instantiation. On this LWE instantiation the estimation is then run. Note that this is indeed an estimation, and should be seen as a lower bound. The $o(1)$ terms that appear are assumed to be positive.
Albrecht, Curtis, Deo, Davidson, Player, Postlethwaite, Virdia and Wun-

derer have tested all lattice-based schemes from the NIST competition in [ACD+18]. Their open-source code is available to simply test other schemes. We have used their code to estimate the BKZ attack on all schemes mentioned above, except HAETAE. For each of the schemes tested, the results can be found in their specific chapters. For HAETAE it was not possible to use the code discussed here due to the designers' choice of error distribution. The code included in the HAETAE submission package is incomplete and even after linking in the estimator from CRYSTALS and decompiling the entropy estimator, the code did not finish successfully after more than two days and we terminated the run.
Note that for fairness sake for each of the schemes we have used the parameters found in their original submission to the competition. We skipped running the estimator for GCKsign and Peregrine.
Table 2.1 shows the names used in this report, with the model associated to it with β as the size of the block, d is a way of measuring the size of the length of the vector.

Name	Model
Q-Core-Sieve	$0.256\ \beta$
Q-Core-Sieve + O(1)	$0.265\beta + 16.4$
Q-Core-Sieve (min space)	0.2975β
Q-β -Sieve	$0.265\beta + \log(\beta)$
Q-8d-Sieve + O(1)	$0.265\ \beta + 16.4 + \log(8d)$
Core-Sieve	$0.292\ \beta$
Core-Sieve + O(1)	$0.292\ \beta + 16.4$
Core-Sieve (min space)	$0.386\ \beta$
β-Sieve	$0.292\ \beta + \log(\beta)$
8d-Sieve + O(1)	$0.292\ \beta + 16.4 + \log(8d)$
Q-Core-Enum + O(1)	$1/2(0.187\beta \log(\beta) - 1.019\beta + 16.1)$
Lotus	$0.125\beta \log(\beta) - 0.755\beta + 2.25$
Core-Enum + O(1)	$0.187\ \beta \log(\beta) - 1.019\beta + 16.1$
8d-Enum (quadratic fit) + O(1)	$0.000784\beta^2 + 0.366\beta - 0.9 + \log(8d)$

Table 2.1: The names used with their corresponding models.

2.2 Security assumptions

LWE-like security assumptions generally have the advantage that they can be related to classical assumptions in lattices via a worst-case to average-case reduction. This means that breaking the lattice-based assumption on

average is as hard as breaking a classical lattice-based assumption in the worst-case. The classical assumption is usually an approximate short vector problem. Often the classical lattices that the assumption is reduced to is more general but the reduction usually only works if the lattices in both assumptions share some key properties. As an example the corresponding reduction for the variant Ring LWE reduces to a worst-case assumption in *ideal* lattices only.

2.2.1 Classical lattice problems are not reflected in state-of-the-art cryptosystems

The average-to worst-case reduction is a major factor of the attractiveness of lattice-based assumptions. It guarantees that, ultimately, a lattice based scheme that relies on an LWE-like assumption can be based on a classical lattice assumption with worst-case guarantees. However, it is important to note that this implication is *not* reflected in modern cryptosystems. In particular, the reduction to classical assumptions has a large security loss (that quantifies efficiency of the reduction to break the LWE-like assumption relative to the efficiency of the attacker against the cryptosystem). This has far-reaching consequences. Instantiating the parameter sizes for Regev-like schemes in a theoretically-sound way when relying on the underlying classical assumption can increase the dimension n of the lattices by a factor of (at least) ten as compared to what is used in practice [Gär23]. This means that for the currently used parameters, to the best of our knowledge, all lattice-based encryption systems do not reduce to some underlying classical worst-case assumption in a theoretically-sound way. Put differently, when applying the existing security reductions to the parameter sizes used in practice they would merely imply that a classical lattice problem can be solved that has very small parameters. However, such a scheme could be practically solved efficiently anyways what makes the guarantees from the security proof vacuous.

2.2.2 Assumptions used in cryptography for encryption systems

An LWE-like assumption states that the distributions $A, b = A \cdot s + e$ and A, b' are computationally indistinguishable where A is a suitable uniformly drawn (quadratic) matrix of dimension n (a lattice), s is a secret vector, e is a small error vector, and b' is a random vector. All values are integers. We observe that essentially $s \mapsto A \cdot s$ is a linear operation that is perturbed by some error

term. Moreover, A spans a lattice. The closeness to purely linear operations gives these schemes its efficiency. In the literature, we can find several variants where the entries in A, s, e, b, b' range over different algebraic structures (and accordingly the operations $\cdot, +$ are defined differently). However, each of these assumptions requires that the mapping $s \mapsto A \cdot s + e$ is injective, giving important conditions on the size of the parameters. In classical LWE, the vector and matrix entries are elements of some field $\mathbb{Z}_q$ while $\cdot$ represents matrix multiplication modulo q. After the original Regev-system that uses the classical plain LWE assumption, several variants have been proposed. One of the most prominent is Ring LWE (RLWE) which works with polynomials in the ring $\mathbf{Z}[x]/f(x)$ for some polynomial $f(x)$, typically $f(x) = x^n + 1$ and additionally reduces modulo some number q. The equation $A \cdot s$ turns into the multiplication of the *polynomials* a and s modulo $f(x)$. This can also be written in matrix form where rows of A are $a \cdot x^i$ followed by reduction modulo $f(x)$ and modulo q. Module LWE (MLWE) can be interpreted as a generalization to a spectrum that has LWE and RLWE as its endpoints, parameterizing the additional polynomial structure introduced [AD17]. In general these variants require more algebraic structure, with RLWE introducing stronger requirements on the algebraic structure than general MLWE. Peikert and Pepin in [PP19] presented a general treatment of the Learning with Errors (LWE) assumption and its variants. Roughly, their framework gives a tight reduction from Ring-LWE (RLWE) to other algebraic LWE variants, including Module-LWE (MLWE), Order-LWE, and Middle-Product LWE. Their work shows that it is possible to use the hardness of Ring-LWE as a foundation for the hardness of all prior algebraic LWE problems. When focusing on LWE, MLWE, and RLWE, this is natural as RLWE places the strongest conditions on the additional structure. However, so far there are no attacks that can specifically exploit the additional structure induced by RLWE (or MLWE). A benefit is that the efficiency of the cryptosystem that are based on LWE-variants can be higher than in plain LWE, both in terms of size as well as in speed of computation. The overall result when using RLWE and MLWE is that in Regev-like encryption systems, we can have higher efficiency with these variants. Roughly in Regev's original crypto system, an encryption of a single message bit would require n values (field elements of $\mathbb{Z}_q$) in the ciphertext and public key. Using RLWE in Regev-like crypto systems, decreases this to a single value. Whereas the definition of RLWE relies on a single polynomial, MLWE considers vectors of polynomials. When comparing this to RLWE, this can be used to balance the number of components of the vector with the length of its entries. Intuitively, at the same level of security, RLWE has to compensate for considering less polynomials by having longer polynomials [AD17].

In general, the community seems to prefer the usage of MLWE over RLWE for fixed public key and ciphertext sizes, since MLWE is a weaker assumption. The RLWE assumption seems only acceptable if it leads to efficiency improvements over MLWE. However, in general the flexibility of the MLWE assumption for different parameter sizes allows to choose one that synergizes particularly well with the remaining system parameters. So although the assumption is weaker it can even lead to more efficient schemes than when based on RLWE.

2.2.3 Computational vs. decisional LWE-variants

The computational version of plain LWE and its variants requires us to compute the secret s from a A, b with $b = A \cdot s + e$ instead of distinguishing it from random (A, b'). The computational and decisional version of LWE and its variants are polynomial-time equivalent. In the following we will usually be concerned with decisional variants since we deal with encryption systems that intrinsically capture that ciphertexts do not reveal a single bit to the attacker by requiring that ciphertexts are indistinguishable from encryptions of random values. We note that there is still a security loss when reducing the computational variant to the decisional [STHY23].

2.2.4 LWE vs. LWR

Whereas the LWE problem adds a small random error e to an otherwise linear equation, the learning with rounding (LWR) assumption introduces a deterministic error that intuitively cuts-off some of the least significant bits (LSBs) of the linear equation. Essentially, instead of blinding the least significant bits with a discrete Gaussian error (the most common error distribution), they are simply deleted. This in general accounts for better bandwidth. It can be formally shown that LWR and LWE are related via this argument. Intuitively, a reduction from LWR to LWE says that in case the blinded LSBs do not give enough information to distinguish an LWE distribution of $A, b = As + e$ from a random distribution (A, b') with random b', then the LSBs can surely not help in case they are missing entirely $A, \lfloor = As \rceil_{p,q}$ where $\lfloor x \rceil_{p,q}$ denotes $\lfloor x(p/q) \rceil$ for integers $2 < p \leq q$ [BBD+19]. This maps values in $\mathbb{Z}_q$ to the smaller set $\mathbb{Z}_q$ and thus looses information. The resulting elements are smaller and improve bandwidth. The disadvantage of LWR is that when performing (homomorphic) operations on the underlying linear equation (which are used for encryption and decryption), the error can accumulate faster than with a Gaussian error in plain LWE. In general this accounts for larger correctness errors. Applying error correcting codes (ECC)

can weaken these effect. However, ECC can in turn increase the susceptibility to side-channel attacks [RRCB20]. However, it is important to note that for PKE and KEMs, the classical security definitions do not take side-channel attacks into account.

2.2.5 Partially-correct encryption system

Due to the introduction of errors (that can in rare cases accumulate quickly), most cryptosystems based on lattices feature non-perfect correctness. This means that in rare cases the decryption of a ciphertext may fail. The probability for this to happen is called decryption failure probability (DFP). For practical parameters, the DFP is usually very small, so that decryption failures will virtually not happen in most usage scenarios. However, an attacker that can find decryption failures learns valuable information on the secret key [DRV20]. Thus attackers might use strategies that specifically search for decryption failures. Such attacks are often called reactive attacks. Recent improvements on reactive attacks improve the success probability to find more decryption failures once a single one has been found for a given key pair [DB22]. So the probabilities of a lattice-based cryptosystem need to be chosen such that finding any decryption failure is hard *in the first place*. However, recent analysis reveals that the DFP of most schemes used in practice are low enough when fixing the maximum number of decryption queries in total to some practical values [DRV20]. It is important to note that when proving security against quantum attackers as opposed to classical attackers, Grover's search algorithm can be utilized by any attacker and in particular attackers that aim at finding decryption failures. We note again that the DFP can naturally be decreased using error-correcting codes. However as mentioned before, this can increases the susceptibility to side-channel attacks (and costs to have implementations that mitigate them).

2.2.6 Security loss

The security proof should have a low security loss that theoretically supports practical parameter choices. This however, highly depends on the security assumption that the security of the scheme is reduced to. As stated before, reductions to the worst-case hardness of classical lattice-based assumptions are very likely to not cover practical parameters since they have considerably high security losses. However, even if the reduction reduces to some decisional LWE-like assumption, the reduction is typically not tight, in particular, if it assumes quantum attackers in the so-called quantum random oracle model (QROM). This is mainly due to the application of the popular Fujisaki-

Okamoto transform (and its variants) that generically turns an IND-CPA secure PKE into a IND-CCA2 secure KEM [HHM22, BHH+19]. Recent results show that under certain conditions this loss is unavoidable [JZM21] (for measurement-based, black-box reductions).

2.2.7 Security assumptions

All schemes are essentially following the same template. For integers p, q with $2 < p \leq q$, if $w \in \mathbb{Z}_q$, let $\lfloor w \rceil_{p,q}$ denote $\lfloor w \cdot p/q \rceil$, where $\lfloor v \rceil$ represents rounding to the integer that is nearest to v. If $p = q$, no rounding is performed whatsoever. If $w \in R_q$ for some polynomial ring R_q with coefficients over $\mathbb{Z}_q$, $\lfloor w \rceil_{p,q}$ applies the rounding operation to each coefficient of w. Likewise, rounding a vector of elements will apply the rounding to each component of the vector. By convention we will understand that having an (error) distribution $\chi_x = \{0\}$ that always maps to zero implies that we will not draw an error at all. In the following we will, for simplicity, focus on atomic formulations of LWE-like assumptions that do not consider matrices A but rather a single row vector $\mathbf{a}$. All these assumptions can be generalized naturally by stacking rows over each other for fixed secret, i.e. by considering matrices as vectors of rows.

Definition 2.2.1 (LWE)**.** *Let n, q be positive integers and let χ_{LWE} be a probability distribution on $\mathbf{Z}_q^n$. Implicitly set $p = q$. Choosing a matrix $A \in \mathbf{Z}_q^{n \times n}$ uniformly at random and choosing $e \in \mathbf{Z}_q^n$ according to χ_{LWE}, define $A_{s,\chi_{\mathsf{LWE}}}$ as the probability distribution on $\mathbf{Z}_q^{n \times n} \times \mathbf{Z}_q^n$ that outputs $(A, b = A \cdot s + e)$ for given secret $s \in \mathbf{Z}_q^n$. Define $U_{\mathsf{LWE}} = (A, w)$ where w is uniform in $\mathbf{Z}_q^n$.*

Definition 2.2.2 (LWR)**.** *Let n, q, p be positive integers with $p < q$. Choosing a matrix $A \in \mathbf{Z}_q^{n \times n}$ uniformly at random, define $\chi_{\mathsf{LWR}} = \{0\}^n$ to be a distribution that always maps to zero and $A_{\mathbf{s},\chi_{\mathsf{LWR}}}$ as the distribution on $\mathbf{Z}_q^{n \times n} \times \mathbf{Z}_q^n$ that outputs $(A, b = \lfloor A \cdot s + e \rceil_{p,q})$ for given secret $s \in \mathbf{Z}_q^n$. Define U_{LWR} as the uniform distribution on $\mathbf{Z}_q^n \times \mathbf{Z}_q$. Define $U_{\mathsf{LWR}} = (A, w)$ where w is uniform in $\mathbf{Z}_q^n$.*

Definition 2.2.3 (RLWE)**.** *Let n be a power of two, let q be a prime integer. Define $R_q = \mathbf{Z}_q[x]/(x^n + 1)$ and let χ_{RLWE} be a probability distribution on R_q^n. Implicitly set $p = q$. Choosing a vector of polynomials $A \in R_q^n$ uniformly at random, drawing e at according to χ_{RLWE}, define $A_{s,\chi_{\mathsf{RLWE}}}$ as the probability distribution on $R_q^n \times R_q^n$ that outputs $(A, b = A \cdot s + e)$ for given secret $s \in R_q$. Define $U_{\mathsf{RLWE}} = (A, w)$ where w is uniform in R_q.*

Definition 2.2.4 (RLWR)**.** *Let n be a power of two, let q be a prime integer, and let p be an integer with $p < q$. Define $R_q = \mathbf{Z}_q[x]/(x^n + 1)$. Define*

$\chi_{\mathsf{RLWR}} = \{0\}^n$ *to be a distribution that always maps to zero. Choosing a vector of polynomials* $A \in R_q^n$ *uniformly at random define* $\chi_{\mathsf{RLWR}} = p$ *and* $A_{s,\chi_{\mathsf{RLWR}}}$ *as the probability distribution on* $R_q^n \times R_q^n$ *that outputs* $(A, b = \lfloor (A \cdot s) \rfloor_{p,q})$ *for given secret* $s \in R_q$*. Define* $U_{\mathsf{RLWR}} = (A, w)$ *where* w *is a vector of uniforms polynomial in* R_q^n*.*

Definition 2.2.5 (MLWE)**.** *Let* n' *be an integer dimension, let* n *be a power of two, let* q *be a prime integer. Define* $R = \mathbf{Z}[x]/(x^n + 1)$ *and* $R_q = (R/qR)^{n'}$ *and let* χ_{MLWE} *be a probability distribution on* R_q^n*. Implicitly set* $p = q$*. Choosing a matrix* $A \in R_q^n$ *uniformly at random and an error* e *according to* χ_{MLWE}*, define the probability distribution* $A_{s,\chi_{\mathsf{MLWE}}}$ *on* $R_q^n \times R_q^n$ *that outputs* $(A, b = A \cdot s + e)$ *for given secret* $s \in R_q$*. Define* $U_{\mathsf{MLWE}} = (A, w)$ *where* w *is uniform in* R_q^n*.*

Definition 2.2.6 (MLWR)**.** *Let* n' *be an integer dimension, let* n *be a power of two, let* q *be a prime integer, and let* p *be an integer with* $p < q$*. Define* $R = \mathbf{Z}[x]/(x^n + 1)$ *and* $R_q = (R/qR)^{n'}$*. Define* $\chi_{\mathsf{MLWR}} = \{0\}^n$ *to be a distribution that always maps to zero. Choosing a matrix* $A \in R_q^n$ *uniformly at random define* $\chi_{\mathsf{MLWR}} = p$ *and* $A_{s,\chi_{\mathsf{MLWR}}}$ *as the probability distribution on* $R_q^n \times R_q$ *that outputs* $(A, b = \lfloor (A \cdot s) \rfloor_{p,q})$ *for given secret* $s \in R_q$*. Define* $U_{\mathsf{MLWR}} = (A, w)$ *where* w *is uniform in* R_q^n*.*

Definition 2.2.7 (Alternative Version)**.** *Consider the* x*-assumption for* $x \in \{\mathsf{LWE}, \mathsf{RLWE}, \mathsf{MLWE}, \mathsf{LWR}, \mathsf{RLWR}, \mathsf{MLWR}\}$*. Using the same parameters, we say that* x' *is the alternative version of* x *if in the computation of the output distribution we compute* $s^T \cdot A$ *instead of* $A \cdot s$*.*

We refer to LWE, MLWE, RLWE (and to any of the corresponding alternative versions) generally as LWE-like and more specifically to LWR, MLWR, RLWR (and to any of the corresponding alternative versions) as LWR-like.

Definition 2.2.8 (Computational LWE-like Problems)**.** *Assume we draw* s *according to some distribution* $\chi_{x,s}$*. Given an arbitrary number of independent samples from* A_{s,χ_x}*, with* $x \in \{\mathsf{LWE}, \mathsf{RLWE}, \mathsf{MLWE}, \mathsf{LWR}, \mathsf{RLWR}, \mathsf{MLWR}\}$ *(or a corresponding alternative version) the computational* x *problem asks to find* s*.*

Definition 2.2.9 ((Decisional) LWE-like Problems)**.** *Assume we draw* s *according to some distribution* $\chi_{x,s}$*. Given uniformly distributed* s*, the decision* $x \in \{\mathsf{LWE}, \mathsf{RLWE}, \mathsf{MLWE}, \mathsf{LWR}, \mathsf{RLWR}, \mathsf{MLWR}\}$ *problem asks to distinguish between samples from* A_{s,χ_x} *and samples from the uniform distribution* U_x*.*

When in the following we speak of any LWE-like assumption we specifically refer to its decisional variant. In classical formulations of the assumptions

we typically have that $\chi_{x,s}$ is the uniform distribution. We say that LWE and LWR have the same algebraic structure if they have the same parameters n, q and they share A. Likewise, we say that RLWE and RLWR have the same algebraic structure if they share the same parameters d, q and they share A. Finally we say that MLWE and MLWR have the same algebraic structure if they share the same parameters n, n', q and they share random A.
The results in [ACPS09] show that for LWE-like schemes the secret keys can come from the same (Gaussian) distribution as the error. This accounts for virtually no security loss. This justifies using small secret keys in the first place.

2.2.8 The basic Regev cryptosystem

Description

We now describe a general form of the Regev system. The Regev system can be based on the $x \in \{\mathsf{LWE}, \mathsf{RLWE}, \mathsf{MLWE}, \mathsf{LWR}, \mathsf{RLWR}, \mathsf{MLWR}\}$ assumption for key generation and the y assumption for ciphertext generation where x and y have the same algebraic structure but possibly distinct p, p'. Observe that these assumptions implicitly define all ambient spaces.
The PKE scheme consists of a collection of three algorithms $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{KeyGen}(1^\kappa)$: The key generator chooses uniformly random A. Next it samples $b = \lfloor A \cdot s + e \rceil_{p,q}$ where s is chosen according to some distribution $\chi_{x,s}$ and $e \leftarrow \chi_x$. The public key is $\mathsf{pk} = (A, b)$ while the secret key is $\mathsf{sk} = s$.

- $\mathsf{Encrypt}(\mathsf{pk}, m)$: To encrypt message m, we compute an ephemeral key $b' = \lfloor s'^T \cdot A + e' \rceil_{p',q}$ for some s' that has been drawn from distribution $\chi_{y,s'}$ and $e' \leftarrow \chi_y$. Next we use the public key of the receiver $\mathsf{pk} = (A, b)$ to compute (c_1, c_2) where $c_1 = b'$ and $c_2 = \lfloor s'^T \cdot b + e'' \rceil_{p',q} + \lfloor mp'/2 \rceil$ where e'' is drawn according to χ_y.

- $\mathsf{Decrypt}(\mathsf{sk}, c)$: To decrypt ciphertext (c_1, c_2) using $\mathsf{sk} = s$, we compute $mq/2 \approx c_2(q/p') - c_1(q/p') \cdot s$ from which we can easily compute m.

In this case we consider a message space $\mathcal{M} = \{0, 1\}^n$.

Correctness

We say PKE has correctness δ if it holds that the probability $\Pr[m = \mathsf{Decrypt}(\mathsf{sk}, \mathsf{Encrypt}(\mathsf{pk}, m)) | (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)] = \delta$. We define the decryption failure probability DFP as $\mathsf{DFP} = 1 - \delta$.

Security

Consider the following game played between attacker A and challenger C.

- The challenger draws a random key pair using $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. Next it computes uniformly random bit m^* and sends pk to the attacker.
- The attacker outputs a message m^*.
- The challenger draws random bit b and random message $m' \in \mathcal{M}$. It sets $m_0 := m^*$ and $m_1 := m'$. Next it computes $c^* \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_b)$ and sends it to A.
- The attacker outputs a bit b' indicating its guess for m^*.

The attacker wins if $b' = b$. The advantage of the attacker to win is defined as $Adv = |Pr[0 \leftarrow A(\mathsf{pk}, c^*)] - Pr[1 \leftarrow A(\mathsf{pk}, c^*)]|$ where the probability is over the random choices of A and C.

Security proof of generic Regev scheme

The security proof for this scheme is very simple. In the first game hop, we exchange the public key element b with a truly random value b'. Any attacker that can observe this change can immediately be used to break the x assumption. Next we change c_1 to random b'' and $\lfloor s'^T \cdot b + e'' \rceil_{p',q}$ to a truly random value b'''. Again, any attacker that can observe this change can be used to break the y assumption. Now we are in a game where the ciphertext is entirely independent of the public key and the message. Thus the attacker can only guess with probability 1/2.

Security loss

The security of the basic Regev scheme reduces tightly to that of the x or y assumption.

2.2.9 The Fujisaki-Okamoto transform (with implicit rejection)

The Fujisaki-Okamoto (FO) transform turns a weakly secure PKE to an IND-CCA secure one [FO99, FO13]. The FO transform has been originally described for PKE. Dent transferred it to construction of KEMs [Den03]. However, the transforms only cover schemes with perfect correctness. The application to partially-correct schemes got much attention in the last years

starting with [HHK17]. A good overview on the classical reductions is given in [FO13]. Some of the key insights are that:

- When using quantum reductions, there is a tight proof from deterministic PKEs with an additional strong property called disjoint simulatability to IND-CCA secure KEMs in the quantum random oracle model (QROM) [SXY18]. Disjoint simulatability is implied by sparse pseudo-randomness.

- When using classical reductions, there is a tight proof from a subclass of probabilistic PKEs to IND-CCA secure KEMs in the random oracle model [FO13].

- When using classical reductions, there is a tight proof from deterministic PKEs to IND-CCA secure KEMs in the random oracle model [FO13].

- When using quantum reductions, there is a non-tight proof from deterministic and probabilistic PKEs to IND-CCA security in the quantum random oracle model (QROM). To avoid high security losses, it is useful to require implicit rejection, where the decryption oracle outputs random responses in case of a decryption failure. All black-box reductions have a square loss of security. Given current knowledge, it seems hard to avoid this non-tight security loss [JZM21] for measurement-based reductions (that measure the output of the QROM). Under certain assumptions on the reversibility of the attacker a non-black box reduction presented at EC'20 in has a linear loss.

- To improve the tightness loss over the standard hybrid argument [BBM00] when analyzing in the more realistic multi-user setting, the public key of the receiver is usually used to derive the shared key. Essentially this works as a mechanism for domain separation of the random oracle used. However, as [DHK+21] show using the full pk is wasteful, a prefix suffices. This results in a considerable increase of the running time of Kyber.

We note that even though the results of [DHK+21] present a much more realistic setting it still does not provide corruption capabilities to the attacker. Strictly speaking, their result only holds if no party is ever corrupted. It is unclear if their improvements over the naive hybrid argument transfer to the setting with adaptive user corruptions as well.

Part I

Public-Key Encryption and Key-Establishment Algorithms

Chapter 3

IPCC – Improved Perfect Code Cryptosystems

IPCC [RKY+22] is a system based on graphs. The submission cites the perfect-code cryptosystems from Fellows and Koblitz [FK93] which is a system designed as a teaching tool rather than as as a secure cryptosystem; the paper is called "Kid Krypto" and this system has featured as an introductory example in Lange's masters course on cryptology at TU/e. The submission changes the system in various ways but still announces the system as based on graphs.

3.1 Security

We (Daniel J. Bernstein, Jolijn Cottaar, and Tanja Lange) have found two efficient attacks breaking IPCC. We have demonstrated both attacks experimentally, and the submitters have acknowledged both attacks.

First attack

As announced in [BCL22], we are able to efficiently compute the plaintext from the ciphertext. The system encrypts a message (some integer) by splitting m into summands m_i where each graph position gets a share of the message and encryption works by using the graph properties to obfuscate these shares, leading to the coefficients c_{1j} and c_{2j} of the ciphertext polynomials. The following states the core parts (mechanisms and costs) of the attack as we announced in December 2022, using the notation from the submission package.

Details of the first attack

The main problem that this attack exploits is that the ciphertext polynomials are very sparse. At best there are

$$|I| \cdot k \cdot 4 \cdot 4$$

variables involved and hence very little mixing happens. This expression comes from

- $|I|$ sets being used in the computation of $f^k_{G_i}$,
- k variables coming from each choice of S_i,
- 4, from each of these variable having 3 neighbors, and
- 4, from there being at most 4 polynomials in the given examples of F (2 for `f1`, 3 for `f3`, and 4 for `f4`)

This means that in the given example systems there are at most

$$3 \cdot 3 \cdot 4 \cdot 4 = 144$$

out of 400 variables and for most monomials no extra addition happens. This means that the shares of the messages m_i appear as coefficients of the monomials. E.g. in the case of `f1` − `f4` there are typically only 15 different coefficients: 9 from the cross products of the 3 c_{1j} shares of m_1 with the $3c_{2j}$ shares of m_2, and 3 more from the new shares of the polynomial for graph 1, and 3 more from the polynomial for graph 2 giving

$$m_1 \cdot m_2 + m_3 + m_4$$

where $m_3 = m_1, m_4 = m_2$ for `f1` and independently chosen for `f4`.
The core of the attack is that the sum of those 15 coefficients taken modulo p gives the plaintext.

Concrete example

Note that the KAT file as provided in the implementation package contains the hashes of the ciphertext which makes it of course good enough for verifying that the implementation matches but makes it hard to see that the ciphertexts are very sparse and that many coefficients repeat. To see the behavior as described in the following, modify the code to omit hashing the ciphertexts when producing the KATs, no other modification is needed.

As an example, consider the first example in the KAT for the case of `f1`. This is given a message $m = 18790$. The ciphertext produced by the reference implementation contains the following list of coefficients (here stated without their multiplicities): $[35, 9087, 14460, 16002, 16620, 21637, 22560, 24760, 33530, 36038, 36868, 38564, 39587, 39792, 62376]$. Summing these up gives us $411916 = 18790 \bmod 65521$, which indeed is the plaintext.
We ran this attack on ciphertexts produced by the KAT. There are some few cases (2 out of 100 for `f1`, 0 out of 100 for `f3`, 8 out of 100 for `f4`) where this simple attack does not give the plaintext: in these cases, there are more than 15 coefficients, because variables repeated, leading to combinations. However, the attack is very fast and succeeded with probability of more than 90% which means that the system is typically broken.
After the authors acknowledged the attack, we did not look further into solving these last few cases. Combinatorics arguments should be enough to determine which of the coefficients we need to skip in summing up. We think that counting the frequency of occurrence will give us information.

Larger parameters

We understand that the implementation is for smaller parameters than the authors suggest and that knowing F helps the attacker. However, there are only $k + 1$ different monomials of total degree k split over the two graphs (terms in $\mathcal{F}$) and $\sum_{i-1}^{k}(i + 1)$ polynomials of absolute degree $\leq k$, which means that the resulting polynomials are still likely to be sparse. Hence, even if only the general choices of $|I|$ and k are known the attacker knows how many distinct coefficients to expect and the problem seems to persist unless there are many more terms.

Second attack

In July 2023, a new version of IPCC, called IPCC7, was announced. This version uses larger polynomials as ciphertexts: monomials are of degree as large as 7, and there are tens of thousands of monomials in ciphertexts. The coefficients are integers modulo 2^{31}. The number of distinct coefficients appearing in a ciphertext is much larger than in the original IPCC.
The IPCC7 documentation, like the documentation for the original version of IPCC, says that key recovery requires finding a perfect dominating set in a 3-regular graph, and reports a conjecture that this problem is NP-hard.
The second attack recovers the secret key from the public key. We have demonstrated key recovery experimentally.

Details of the second attack

Key generation partitions $\{0, 1, \ldots, 255\}$ into four secret subsets A, B, C, D, each of cardinality 64. It then builds an undirected 3-regular graph by connecting each element of A to a random element of B; same for A and C; same for A and D; same for B and C; same for B and D; same for C and D. The public key communicates this graph. The secret key is A.

It is helpful to think of the secret key as specifying 256 secret variables $v_0, v_1, \ldots, v_{255}$, where $v_j = 1$ for $j \in A$ and $v_j = 0$ for $j \notin A$. These variables satisfy public equations: specifically, for each $i \in \{0, 1, \ldots, 255\}$, one has $v_i + v_j + v_k + v_\ell = 1$ where j, k, ℓ are the neighbors of i in the graph, since by construction exactly one of i, j, k, ℓ is in A.

The attack applies linear algebra to reduce these equations to echelon form. This produces equations expressing each v_i as a linear combination of independent variables. In experiments with 10 public keys, 7 of the keys used just four linear combinations (for example, classifying each i as $v_i = v_{253}$ or $v_i = v_{254}$ or $v_i = v_{255}$ or $v_i = 1 - v_{253} - v_{254} - v_{255}$), partitioning the 256 variables into the secrets A, B, C, D in some order; decryption works with any of the equivalent secret keys A, B, C, D.

For the remaining 3 public keys, the same process partitioned $\{0, 1, \ldots, 255\}$ into six sets of cardinality $1, 1, 63, 63, 64, 64$. In other words, the public equations left two possibilities for the partition. The attacker can simply take either of the two cardinality-64 sets.

Presumably there are further cases where linear algebra leaves additional ambiguities, but it is clear that the system is broken with high probability.

3.2 Implementation considerations

The IPCC system has very large ciphertexts and is rather slow.

The original reference implementation works well enough to run through some selected examples of key generation, encryption, and decryption, but for smaller parameters than the authors recommend. For IPCC7, the reference implementation appears to use the full recommended ciphertext degree, but encrypts only 31-bit messages. Both implementations would need to be extended to handle larger messages and to add CCA protection.

The test programs included with the reference implementations do not run known-answer tests for 100 pseudorandom messages, but only one single message for one single key pair. Adjusting the implementations to try more messages sometimes triggers buffer overflows caught by `gcc -fsanitize=address`, in part because of missing parentheses in macro defi-

nitions in IPCC's `params.h`. The reference implementation of IPCC7 sometimes does not decrypt correctly, possibly because of the following ambiguity in the data structure for representing monomials: monomials of degree below 7 are padded with byte 0, but byte 0 also represents a valid variable.
In the reference implementation of IPCC7, public keys include A-B edges, then A-C edges, etc., allowing a very simple attack that intersects the A-B edges with the A-C edges to find A. Sorting the list of edges before releasing a public key (along with sorting the two vertices in each edge, as the implementation already does) would stop this simple attack and would ensure that the public key does not leak any information beyond the public graph. This would not affect the second attack described above.

Chapter 4

Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes

Layered-ROLLO-I is a blockwise interleaved ideal low-rank parity-check (BII-LRPC) code-based KEM, which was proposed in [KKN23d] and submitted to the KpqC Competition under the name Layered ROLLO-I [KKN22]. Layered ROLLO-I is a modified version of ROLLO [ABD+19] (including ROLLO-I), which was deselected from the NIST competition after round 2 because of attacks on rank-metric codes decreasing the security below the required levels.
This chapter follows an upcoming preprint by Chee, Jeong, Lange, Lee, Pellegrini, and Ryu resulting from the attack announcements on the KpqC forum.
Notation In the specifications of this chapter, we will make use of the following objects. Denote by $S_w^n(\mathbb{F}_{q^m})$ the set of vectors of length n and rank weight w over $\mathbb{F}_{q^m}$:

$$S_w^n(\mathbb{F}_{q^m}) := \{\mathbf{x} \in \mathbb{F}_{q^m}^n \mid \mathsf{wt}_R(\mathbf{x}) = w\}.$$

The Rank Support Recovery ($\mathsf{RSR}(F, s, r)$) algorithm is used as a decoder in the decapsulation procedures of ROLLO-I and the follow-up designs. It recovers the support of (the $\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}$ generated by) the error vector given the support E of the secret key and the rank of the error. This corresponds to actually finding the error coordinates, by solving a linear system of equations (see p. 13 of the ROLLO specification [ABD+19]).
Let $P(x) \in \mathbb{F}_q[x]$ be a polynomial of degree n. We can identify the vector space $\mathbb{F}_{q^m}^n$ with the ring $\mathbb{F}_{q^m}[x]/(P(x))$, where $(P(x))$ is the ideal of $\mathbb{F}_{q^m}[x]$ generated by $P(x)$. Given $\mathbf{u} \in \mathbb{F}_{q^m}^n$, denote by $\mathbf{u}(x) \in \mathbb{F}_{q^m}[x]$ the polynomial

$\mathbf{u}(x) = \sum_{i=0}^{n-1} u_i x^i$. Given $\mathbf{u}, \mathbf{v} \in \mathbb{F}_{q^m}^n$, we define their product $\mathbf{uv}$ as the unique vector $\mathsf{w} \in \mathbb{F}_{q^m}^n$ such that $\mathsf{w}(x) = \mathbf{u}(x)\mathbf{v}(x) \bmod P(x)$. Similarly, we define $Q\mathbf{u} = Q(x)\mathbf{u}(x) \bmod P(x)$ for $Q(x) \in \mathbb{F}_{q^m}[x]$ and $\mathbf{u}^{-1}$ for $\mathbf{u}(x)$ invertible modulo $P(x)$.

4.1 ROLLO-I, Layered-ROLLO-I and reduction attacks

In this chapter we will introduce ROLLO-I and two versions Layered-ROLLO-I, namely Layered-ROLLO-I and Modified-Layered-ROLLO-I. For each of the versions we give a reduction that efficiently transforms any instance of such version of Layered-ROLLO-I to an instance of ROLLO-I. Since the BII-LRPC code-based KEM in [KKN23d] is a modified algorithm of ROLLO-I, we introduce ROLLO-I first.

4.1.1 ROLLO-I

The values (q, n, m, r, d, P) are the system parameters, where q, n, m, r, d are integers and $P(x) \in \mathbb{F}_{q^m}[x]$ is a primitive polynomial of degree n.

- KeyGen:
  - Pick random $\mathbf{x}, \mathbf{y} \in S_d^n(\mathbb{F}_{q^m})$.
  - Set $\mathbf{h}(x) = \mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P(x)$.
  - Return $\mathsf{pk} = \mathbf{h}$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- Encap(pk):
  - Pick random $\mathbf{e}_1, \mathbf{e}_2 \in S_r^n(\mathbb{F}_{q^m})$.
  - Set $E = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle$.
  - Return $K = \mathsf{hash}(E)$ and $\mathbf{c}(x) = \mathbf{e}_1(x) + \mathbf{e}_2(x)\mathbf{h}(x) \bmod P(x)$.
- Decap($\mathbf{c}, \mathsf{sk}$):
  - Set $\mathbf{s}(x) = \mathbf{x}(x)\mathbf{c}(x) \bmod P(x)$, $F = \langle \mathbf{x}, \mathbf{y} \rangle$ and $E = \mathsf{RSR}(F, \mathbf{s}, r)$, where $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the $\mathbb{F}_q$-vector space spanned by the columns of $\mathbf{x}$ and $\mathbf{y}$ (interpreted as vectors in $\mathbb{F}_q^m$).
  - Return $K = \mathsf{hash}(E)$.

Note that $\mathbf{s} = \mathbf{e}_1(x)\mathbf{x}(x) + \mathbf{e}_2(x)\mathbf{y}(x) \bmod P(x)$ is in F apart from the rank-r error E which is thus returned by $\mathsf{RSR}(F, \mathbf{s}, r)$.

4.1.2 Layered-ROLLO-I

This description follows [KKN23d] apart from skipping the explicit maps between coefficient vectors and polynomials. The values (q, n, m, r, d, b, P) are the system parameters, where q, n, m, r, d, b are integers, with n a multiple of b, and $P(x) \in \mathbb{F}_{q^m}[x]$ is a primitive polynomial of degree n/b. For all given parameter sets $b = 2$ and in any case $b < n/b$. The map $\Psi : \mathbb{F}_{q^m}[x]/(P(x)) \to \mathbb{F}_{q^m}[x]/(P(x)^b)$ casts polynomials of the first quotient into the second quotient by mapping the input to the unique polynomial of degree $< n/b$ that is congruent to it modulo $P(x)^b$. Similarly, the map $\Omega : \mathbb{F}_{q^m}[x]/(P(x)^b) \to \mathbb{F}_{q^m}[x]/(P(x))$ reduces the input modulo $P(x)$. Since $P(x)^b$ is a multiple of $P(x)$ this map is well-defined.

- KeyGen:
  - Pick random $\mathbf{x}, \mathbf{y} \in S_d^{n/b}(\mathbb{F}_{q^m})$.
  - Pick random irreducible $P_I(x) \in \mathbb{F}_{q^m}[x]/(P(x))$ of degree $(b-1)$.
  - Pick random $P_O(x), P_N(x) \in \mathbb{F}_{q^m}[x]/(P(x)^b)$, with $P_O(x)$ invertible (this last restriction is not stated but is required for functionality).
  - Set $\mathbf{z}(x) = P_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P(x)$.
  - Set $P_P(x) = P_O(x)\Psi(P_I(x)) \bmod P(x)^b$ and $P_H(x) = P_O(x)\Psi(\mathbf{z}(x)) + P_N(x)P(x) \bmod P(x)^b$.
  - Return $\mathsf{pk} = (P_P, P_H)$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y}, P_O, P_I)$.
- Encap(pk):
  - Pick random $E = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle \in S_r^{n/b}(\mathbb{F}_{q^m})$, with $\mathbf{e}_1, \mathbf{e}_2$ each corresponding to a polynomial of degree $< n/b - b$.
  - Set $P_{E_1}(x) = \Psi(\mathbf{e}_1(x))$ and $P_{E_2}(x) = \Psi(\mathbf{e}_2(x))$.
  - Compute $\mathbf{c}(x) = P_P(x)P_{E_1}(x) + P_H(x)P_{E_2}(x) \bmod P(x)^b$.
  - Return $K = \mathsf{hash}(E)$ and $\mathbf{c}$.
- Decap($\mathbf{c}, \mathsf{sk}$):
  - Compute $P_C(x) = P_O(x)^{-1}\mathbf{c}(x) \bmod P(x)^b$.
  - Compute $\mathbf{c}'(x) = P_I(x)^{-1}\Omega(P_C(x)) \bmod P(x)$.
  - Decode $E = \mathsf{RSR}(\langle \mathbf{x}, \mathbf{y} \rangle, \mathbf{x}\mathbf{c}', r)$.
  - Return $K = \mathsf{hash}(E)$.

4.1.3 Reduction of Layered-ROLLO-I to ROLLO-I

In [CJLR23] the authors propose a reduction of Layered-ROLLO-I to ROLLO-I by using exclusively the public key of the former. To start with, notice that P_O must have an inverse modulo P^b. This has not been declared in the specification but the decapsulation process requires P_O^{-1}. If not, decapsulation fails. Also, P_I is irreducible of degree $(b-1) < n/b = \deg P$, so it has an inverse modulo P and thus $\Psi(P_I)$ is invertible modulo P^b. Therefore, we can invert P_P modulo P^b and compute $P_P(x)^{-1}P_H(x)$ as

$$\Psi(P_I(x))^{-1}\Psi(\mathbf{z}(x)) + P_P(x)^{-1}P_N(x)P(x) + k(x)P(x)^b \tag{4.1}$$

for some $k(x) \in \mathbb{F}_{q^m}[x]$. Since P divides P^b we can reduce the equation modulo P, obtaining

$$P_P(x)^{-1}P_H(x) \equiv \Psi(P_I(x))^{-1}P_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P(x),$$

which is equal to $\mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P(x)$, where we use the fact that $\Psi(P_I(x)) \bmod P(x) = P_I(x) \bmod P(x)$ and $\Psi(\mathbf{z}(x)) \bmod P(x) = P_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x)$.
This shows that the public key of (q, n, m, r, d, b)-Layered-ROLLO-I can be reduced to the public key of $(q, n/b, m, r, d)$-ROLLO-I. The same can be done for ciphertexts by computing $P_P(x)^{-1}\mathbf{c}(x) \bmod P(x)$ which gives a ROLLO-I ciphertext. Therefore, it is possible to reduce an entire instance of (q, n, m, r, d, b, P)-Layered-ROLLO-I to an instance of $(q, n/b, m, r, d, P)$-ROLLO-I.
Attack costs before and after reduction The authors of [KKN23d] suggested a possible attack scenario using the structure of BII-LRPC codes, which can be applied to the instance of $(q, n/b, m, r, d)$-ROLLO-I. The structural attack costs also reveal the proposed parameters do not satisfy the required security levels. Notice that the structural attack complexity in [KKN23d] needs to be corrected to

$$S'_S = \left(\frac{n}{b}\right)^3 m^3 q^{(b-1)m+d\lceil\frac{m}{2}\rceil-m-\frac{n}{b}}. \tag{4.2}$$

After the attacks but before being deselected by NIST, the ROLLO team presented new parameters to defend against the then three best-known attacks; combinatorial, structural, and algebraic attacks [ABD+20]. Note that Layered ROLLO-I partially applied the combinatorial and structural attacks for its parameter selection. That is, it suggests finding the correct P_I by searching the keyspace with the complexity $\mathcal{O}(q^{(b-1)m})$ to obtain $\mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P(x)$, then applying the combinatorial or the structural attack.

We recompute the costs of rank decoding attacks, finding out that the proposed parameters are not suitable for the requested security levels. In this report we consider the breakthrough algebraic attack [BBB+20] and followup modifications [BBC+20a, BBB+23]. We computed complexities of these attacks using the script available at https://gitlab.tue.nl/tlange/kpqc-public/-/tree/master/lrollo, adapted from [LPR23]. The Sage script performs puncturing of the public code to find the optimal complexity. For more details on these attacks see Chapter 7 on REDOG.
Table 4.1 reports the best complexities among the attacks considered in the original paper and these new attacks, where we discard the options in [BBC+20a] that have been proved too optimistic in [BBB+23].

Security	(q, n, m, r, d, b)	Cost [KKN23d]	Cost [BBC+20a]	Cost red. [BBC+20a]
128	$(2, 74, 67, 3, 2, 2)$	130.83	48.76	40.65
192	$(2, 86, 79, 4, 3, 2)$	199.19	66.21	55.16
256	$(2, 106, 97, 5, 3, 2)$	274.98	85.68	72.05

Table 4.1: Suggested parameters, values of the $\log_2$ of attack costs for Layered-ROLLO-I's considered in the original paper, values of the costs considering also the new attacks and values of the costs after our reduction to ROLLO-I.

As per the table, the most efficient attack comes from [BBC+20a], for both the original Layered-ROLLO-I and the reduced version after our attack. These costs also show that the suggested parameters of Layered-ROLLO-I cannot provide the claimed security.

4.1.4 Modified Layered-ROLLO-I

This section extracts the description of the modified system from [KKN23a]. The values $(q, n_1, n_2, d_I, m, r, d, b)$, where $d_I < n_1 < n_2$ are the system parameters. The two polynomials P_1 and P_2 are primitive of degrees n_1 and n_2 respectively. These are not stated among the system parameters but are needed for the functioning of the system. In the following, we assume that P_1 and P_2 are part of the system parameters.

- KeyGen:
  - Pick random $\mathbf{x}, \mathbf{y} \in S_d^{n_1}(\mathbb{F}_{q^m})$.
  - Pick random irreducible $P_I(x) \in \mathbb{F}_{q^m}[x]/(P_1(x))$ of degree d_I.
  - Pick random $P_O(x) \in \mathbb{F}_{q^m}[x]/(P_2(x))$.

- Set $\mathbf{z}(x) = P_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P_1(x)$.
- Set $P_P(x) = P_O(x)\Psi(P_I(x)) \bmod P_2(x)$ and $P_H(x) = P_O(x)\Psi(\mathbf{z}(x)) \bmod P_2(x)$.
- Return $\mathsf{pk} = (P_P, P_H)$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y}, P_O, P_I)$.

The proposed parameters for the Modified-Layered-ROLLO-I along with the attack costs are displayed the third column in Table 4.2. The table shows that the security is still lower for these parameters than the targeted security levels, even though the designers were now aware of the attacks in [BBC+20a]. The modified version of Layered-ROLLO-I overcomes the reduction in section 4.1.3 by replacing the two moduli P and P^b by two primitive polynomials P_1 and P_2 of degree n_1 and n_2, respectively. In this setting, one cannot simply reduce equation (4.1) modulo P_1 as the term $k(x)P_2(x)$ would not vanish.

4.1.5 Reduction of Modified-Layered-ROLLO-I to ROLLO-I

In this section, we will describe a reduction of the Modified-Layered-ROLLO-I to ROLLO-I, which we (Tanja Lange and Alex Pellegrini) found and announced on the KpqC Bulletin [LP23]. Along the way we compute P_I and P_O which are part of the secret key, meaning that the system leaks private information. The authors of Layered ROLLO-I reacted by modifying the scheme, see the next section, thereby acknowledging the attack.
The idea of the reduction remains the same, observing that $P_H(x)/P_P(x)$ cancels the P_O. However, because of the coprimality of the moduli, we cannot proceed directly from there to reducing modulo P_1. Nevertheless, we know that the polynomials involved have very low degrees. Let $R(x) = P_H(x)/P_P(x) \bmod P_2(x)$ then $\deg(R) < n_2$ and $R(x) = \Psi(\mathbf{z}(x))/\Psi(P_I(x)) \bmod P_2(x)$ with $\deg(\mathbf{z}) < n_1$ and d_I small. Note that the division might cancel common factors of P_I and $\mathbf{z}$, however, given the degrees this is unlikely.

Remark 4.1.1. *When viewing a polynomial $\mathbf{v}(x) \in \mathbb{F}_{q^m}[x]$ of degree at most n as an element of $\mathbf{v} \in \mathbb{F}_{q^m}^{n+1}$ corresponding to its coefficient vector, we consider the entries of $\mathbf{v}$ to be ordered in a way such that $\mathbf{v}(x) = \sum_{i=0}^{n} v_i x^i$.*

Let M_R be the $(d_I + 1) \times n_2$ matrix over $\mathbb{F}_{q^m}$ representing multiplication of

a polynomial of degree up to d_I by R modulo P_2, i.e.

$$M_R = \begin{pmatrix} R(x) \bmod P_2(x) \\ R(x)x \bmod P_2(x) \\ \vdots \\ R(x)x^{d_I} \bmod P_2(x) \end{pmatrix}, \tag{4.3}$$

where each row consists of the coefficient vector of $R(x)x^i \bmod P_2$ for $i = 0, \ldots, d_I$.

Remark 4.1.2. *Let A be any $n \times m$ matrix, with $n, m \in \mathbf{N}$. We denote $A[a : b, c : d]$, with $a < b \in [1, n]$ and $c < d \in [1, m]$, the submatrix of A consisting of the rows in the range $[a, b]$ and columns in the range $[c, d]$. We omit a and b, i.e. $A[:, c : d]$ to denote the submatrix consisting of all the rows and taking columns in $[c, d]$. Similarly, for all the columns. With this notation $A = A[:, :]$. If $S_1 \subset [1, n]$ and $S_2 \subset [1, m]$ we denote $A[S_1, S_2]$ the submatrix of A consisting of rows indexed by S_1 and columns indexed by S_2.*

Note that multiplication by R defines an automorphism of the field $\mathbb{F}_{q^m}[x]/(P_2(x))$, thus the associated matrix M has rank n_2. Therefore, since $M_R = M[1 : d_I + 1, :]$, it has rank $d_I + 1$. Let $\pi : \mathbb{F}_{q^m}^{n_2} \to \mathbb{F}_{q^m}^{d_I+1}$ be the projection of an element of $\mathbb{F}_{q^m}^{n_2}$ onto its first $d_I + 1$ coordinates. Consider

$$\pi(\Psi(P_I(x)))M_R = \Psi(\mathbf{z}(x)) \tag{4.4}$$

as a linear system of equations in the coefficients of $\Psi(P_I)$ and $\Psi(\mathbf{z})$, where in this case we view $\Psi(\mathbf{z})$ as an element of $\mathbb{F}_{q^m}^{n_2}$ consisting of the unknown coefficients of $\Psi(\mathbf{z})$ and $n_2 - n_1$ trailing zeroes. Note that π does not induce any loss of information due to the degree of $\Psi(P_I)$. Since $\deg(\Psi(P_I)) + n_1 = d_I + n_1 < n_2$, the system has a solution corresponding to the representatives of P_I and $\mathbf{z}$ modulo P_1 (here we remove the Ψ notation as the solutions will have degree lower than n_1).
We can actually compute P_I from a subset of the equations defined by (4.4). Indeed, $\pi(\Psi(P_I))$ lies in the left kernel of the submatrix of M_R that consists of the last $n_2 - n_1$ columns, meaning that such submatrix has rank at most d_I, and typically exactly d_I as this system is defined over $\mathbb{F}_{q^m}$. Hence, let $J \subset \{n_1 + 1, \ldots, n_2\}$ having cardinality $\#J = d_I$. We only require $M_R[:, J]$ to have rank d_I, which holds for most choices of J, so typically we take the last d_I columns. This makes explicit that the system is underdetermined, and in case $M_R[:, J]$ has rank lower than d_I, we can include further columns. From (4.4) we can compute P_I by solving

$$\pi(\Psi(P_I(x)))M_R[:, J] = \mathbf{0} \tag{4.5}$$

Since also $\lambda\pi(\Psi(P_I(x)))M_R[:, J] = \mathbf{0}$ for any constant $\lambda \in \mathbb{F}_{q^m}$ we can recover P_I only up to such a constant factor. We will now show that this is not a problem. Let $P'_I(x) = \lambda P_I(x)$. We can recover $P'_O(x) = P_P(x)/P'_I(x) = P_O(x)/\lambda$, then $\mathbf{z}'(x) = P_H(x)/P'_O(x) = P'_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x) = \lambda P_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x)$, and finally $\mathbf{x}(x)^{-1}\mathbf{y}(x) = \mathbf{z}'(x)/P'_I(x)$ which corresponds to a ROLLO-I public key.
Similarly, for the ciphertext, we can recover $\lambda\mathbf{c}''(x) = (P'_O(x))^{-1}\mathbf{c}(x) = \lambda\Psi(P_I(x))P_{E_1}(x) + \lambda\Psi(\mathbf{z}(x))P_{E_2}(x) \bmod P_2(x)$. Since λ is constant, the degree of the right-hand side is below n_2 and we can reduce modulo P_1 and divide by $P'_I(x) = \lambda P_I(x)$ to get $P_{E_1}(x) + \lambda\mathbf{x}(x)^{-1}\mathbf{y}(x)P_{E_2}(x)$, matching the ROLLO-I ciphertexts. Note that the degree constraint on $\deg(P_{E_i}) < n_2 - n_1 - d_I$ for all proposed parameters implies that $\deg(P_{E_i}) < n_1$, hence, this is a valid ROLLO-I ciphertext. While this is not pointed out in the slides, this is also required for the Modified-Layered ROLLO-I decoder to work as $\mathsf{RSR}(\langle\mathbf{x}, \mathbf{y}\rangle, \mathbf{xc}', r)$ in the regular decapsulation procedure.
The reduction in code length and dimension from using n_2 to using n_1 reduces the security further by more than 10 bits for each security level, see the fourth column in Table 4.2.
We implemented our reduction in SageMath. The time in seconds to compute the public key transformation described in this section, on a Linux Mint virtual machine, is stated in Table 4.2 (Time).
Note that here we use P_I with $\deg P_I = d_I$ as stated on the slides. The parameters file in the implementation package instead uses $\deg P_I = 4$ for all security levels.

Security	(q, n_1, n_2, m, r, d)	Cost	Cost red.	Time (s)
128	$(2, 37, 61, 67, 6, 2)$	103.83 [BBB+23]	96.95 [BBB+23]	1.85
192	$(2, 43, 71, 79, 7, 3)$	185.52 [BBB+20]	156.16 [BBB+23]	2.42
256	$(2, 53, 103, 97, 7, 3)$	187.91 [BBB+23]	151.11 [BBB+23]	4.211

Table 4.2: Values of the $\log_2$ of attack costs for Modified-Layered-ROLLO-I's suggested parameters, before and after our reduction, and time consumed by the reduction.

4.2 Message Recovery Attacks

We are going to introduce another two versions of Layered-ROLLO-I, namely the New-Modified-Layered-ROLLO-I [KKN23b] and Layered-ROLLO-I-20231020 [KKN23c], that were posted by the authors of Layered

ROLLO-I after the announcement of the last reduction attack. First we describe the New-Modified-Layered-ROLLO-I, we then give an efficient message recovery attack on this version and all the previous versions of Layered-ROLLO-I. Thereafter, we describe Layered-ROLLO-I-20231020 and give an efficient message recovery attack for security levels 128 and 192.

4.2.1 New-Modified Layered-ROLLO-I

In this section, we describe the system from [KKN23b]. The new version of Layered-ROLLO-I uses polynomial masking techniques in order to avoid the reduction to ROLLO-I described in section 4.1.5. To this end, the new system patch introduces an auxiliary polynomial P_N of small degree and modifies the P_P-part of the public key.
The values $(q, n_1, n_2, n_I, m, r, d)$, where $n_I < n_1 < n_2$ are the system parameters. There is also a primitive polynomial P_2 of degree n_2 which is a system parameter. We will report here only the key generation procedure, as the rest is the same as for Modified Layered-ROLLO-I except for the degree of the error polynomials. The key generation procedure of the new system works as follows.

- KeyGen:
  - Pick random $\mathbf{x}, \mathbf{y} \in S_d^{n_1}(\mathbb{F}_{q^m})$.
  - Pick random primitive $P_1(x) \in \mathbb{F}_{q^m}[x]$ of degree n_1.
  - Pick random $P_I(x) \in \mathbb{F}_{q^m}[x]/(P_1(x))$ of degree n_I.
  - Pick random $P_O(x), P_N(x) \in \mathbb{F}_{q^m}[x]/(P_2(x))$, with $\deg P_N = n_N$.
  - Set $\mathbf{z}(x) = P_I(x)\mathbf{x}(x)^{-1}\mathbf{y}(x) \bmod P_1(x)$.
  - Set $P_P(x) = P_O(x)(\Psi(P_I(x)) + P_N(x)P_1(x)) \bmod P_2(x)$ and $P_H(x) = P_O(x)\Psi(\mathbf{z}(x)) \bmod P_2(x)$.
  - Return $\mathsf{pk} = (P_P, P_H)$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y}, P_O, P_I, P_1)$.

The encapsulation mechanism with updated error-weights is equivalent to that of Modified Layered-ROLLO-I except that the random vectors $\mathbf{e}_1, \mathbf{e}_2$ should correspond to a polynomial of degree $n_E < n_2 - n_1 - n_I - n_N - 2$.

4.2.2 Message recovery attack on Layered-ROLLO-I

We describe a message recovery attacks that we mounted against all the versions of the Layered-ROLLO-I cryptosystem described so far. The idea is to reduce the modular equation in the encapsulation to a system of linear

equations and exploit the knowledge of zero positions of the error vectors to solve the system.
Recall that encapsulation computes the ciphertext

$$\mathbf{c}(x) = P_{E_1}(x)P_P(x) + P_{E_2}(x)P_H(x) \bmod P_2(x).$$

We can multiply the ciphertext polynomial by $P_H(x)^{-1} \bmod P_2(x)$ and obtain

$$\bar{\mathbf{c}}(x) = \mathbf{c}(x)P_H(x)^{-1} = P_{E_1}(x)R(x) + P_{E_2}(x) \bmod P_2(x) \tag{4.6}$$

where $R(x) := P_P(x)P_H(x)^{-1} \bmod P_2(x)$. View equation (4.6) in terms of $\mathbb{F}_{q^m}$ vectors corresponding to the coefficient vectors of the polynomials involved. As in section 4.1.5, we can regard R as the $(n_E + 1) \times n_2$ full rank matrix M_R over $\mathbb{F}_{q^m}$ representing the multiplication of a polynomial of degree up to n_E by R modulo P_2, defined as in (4.3). In other words, M_R generates a linear $[n_2, n_E + 1]$-code over $\mathbb{F}_{q^m}$.
With this in mind we can rewrite (4.6) as

$$\bar{\mathbf{c}} = \mathbf{e}_1 M_R + \mathbf{e}_2, \tag{4.7}$$

which corresponds to a McEliece-like encryption of the message $\mathbf{e}_1$. In the settings of all the versions of Layered-ROLLO-I, we can exploit the low degree of the polynomial P_{E_2}. Indeed, we can find an invertible submatrix of M_R that consists of a subset of columns corresponding to error-free positions in the ciphertext. By remark 4.1.1 the error vector $\mathbf{e}_2$ has non-zero entries only in the first $n_E + 1$ coordinates. Therefore, we can search for an invertible submatrix of $M_R[:, n_E + 2 : n_2]$. Picking $n_E + 1$ random columns of a rank n_E+1 matrix over $\mathbb{F}_{q^m}$, where q and m are given by the suggested parameters, will constitute an invertible matrix with overwhelming probability. We can also just take the last $n_E + 1$ columns.
Let $M_{R\mathsf{inv}}$ be such a matrix. The last step is to compute $\mathbf{e}_1 = \bar{\mathbf{c}}' M_{R\mathsf{inv}}^{-1}$, where $\bar{\mathbf{c}}'$ consists of the coordinates of $\bar{\mathbf{c}}$ corresponding to the columns of $M_{R\mathsf{inv}}$. Finally, compute $\mathbf{e}_2 = \bar{\mathbf{c}}[1 : n_E + 1] - \mathbf{e}_1 M_R[:, 1 : n_E + 1]$.
We implemented this attack in SageMath. An average of the time required, on a Linux Mint virtual machine, to recover the plaintext for the proposed parameters of the version described in section 4.2.1 is given in table 4.3.

Security	n_E	Time (s)
128	17	2.21
192	19	3.18
256	39	6.65

Table 4.3: Average time in seconds (on 50 samples for each security level) needed to recover a plaintext.

Remark 4.2.1. *We would like to remark that this message recovery attack works for all the versions of layered-ROLLO-I. Even for the November 2022 submission, the degrees of* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *were smaller than half of* n_2*, which is relevant for the positions in* $M_{R\mathsf{inv}}$ *not to overlap with the positions in* $\mathbf{e}_2$*. The attack was prompted by the version the authors announced on 22 Sep (New Modified Layered ROLLO-I) and Alex Pellegrini posted his attack on 3 Oct as [Pel23a].*

4.2.3 Layered-ROLLO-I-20231020

In this section, we describe the system from [KKN23c]. The new version of Layered-ROLLO-I uses polynomial masking in the ciphertext in order to overcome the message recovery attack described in section 4.2. We will only display the parts in the specification of KeyGen and Encap that differ from that of New-Modified Layered-ROLLO-I (see Section 4.2.1). The values $(q, n_1, n_2, n_I, n_A, m, r, d)$, where $n_I = n_1 < n_2$ and $n_A = 4$ are the system parameters. The updates to key generation procedure of the new system are as follows.

- KeyGen:
  - Pick random $P_{N,A}(x), P_{N,B}(x) \in \mathbb{F}_{q^m}[x]/(P_2(x))$ of degree n_A
  - Set $P_P(x) = P_O(x)(\Psi(P_I(x)) + P_{N,A}(x)P_1(x)) \bmod P_2(x)$,
    $P_H(x) = P_O(x)\Psi(\mathbf{z}(x)) \bmod P_2(x)$ and
    $P_B(x) = P_O(x)P_{N,B}(x)P_1(x) \bmod P_2(x)$.
  - Return $\mathsf{pk} = (P_P, P_H, P_B)$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y}, P_O, P_I, P_1)$.

The updates to the encapsulation mechanism with updated error-weights are as follows.

- Encap(pk):
  - Compute $\mathbf{c}(x) = P_P(x)P_{E_1}(x) + P_H(x)P_{E_2}(x) + P_B(x)P_{N,C}(x) \bmod P_2(x)$.

where P_{E_1}, P_{E_2} and $P_{N,C}$ have degree $n_E < n_2 - n_1 - n_A - 1$. Let $\ell = n_2 - n_1 - n_A - 1$. The decapsulation procedure has not been updated. Finally, the suggested parameters for the 20231020 version are

Security parameter	$(q, m, n_I, n_1, n_2, n_A)$
128	$(2, 67, 37, 37, 61, 4)$
192	$(2, 79, 43, 43, 71, 4)$
256	$(2, 97, 53, 53, 103, 4)$

4.2.4 Message recovery attack on Layered-ROLLO-I-20231020

We describe a fast message recovery attack on the security levels 128 and 192 of Layered-ROLLO-I-20231020, that uses only linear algebra.
Compute the polynomials

$$\begin{aligned} A_1(x) &= P_P(x)P_B^{-1}(x), \\ B_1(x) &= P_H(x)P_B^{-1}(x), \\ A_2(x) &= P_P(x)P_H^{-1}(x), \\ C_2(x) &= P_B(x)P_H^{-1}(x), \\ B_3(x) &= P_H(x)P_P^{-1}(x), \\ C_3(x) &= P_B(x)P_P^{-1}(x) \end{aligned}$$

and let $M_{A_1}, M_{B_1}, M_{A_2}, M_{C_2}, M_{B_3}$ and M_{C_3} be the corresponding matrices as in (4.3). Set

$$\begin{aligned} \mathbf{c}_1(x) &= \mathbf{c}(x)P_B^{-1}(x), \\ \mathbf{c}_2(x) &= \mathbf{c}(x)P_H^{-1}(x) \text{ and} \\ \mathbf{c}_3(x) &= \mathbf{c}(x)P_P^{-1}(x). \end{aligned}$$

From these values we derive the following equations

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{e}_1 M_{A_1} + M_{B_1}\mathbf{e}_2 + \mathbf{p} \\ \mathbf{c}_2 &= \mathbf{e}_1 M_{A_2} + \mathbf{e}_2 + \mathbf{p}M_{C_2} \\ \mathbf{c}_3 &= \mathbf{e}_1 + \mathbf{e}_2 M_{B_3} + \mathbf{p}M_{C_3} \end{aligned} \tag{4.8}$$

where we denote the coefficient vector of $P_{N,C}$ with $\mathbf{p}$. A first key observation is that, if we restrict to the last $n_2 - \ell$ columns of each matrix, corresponding to the terms of degree $\geq \ell$, we can remove the terms $\mathbf{p}, \mathbf{e}_2$ and $\mathbf{e}_1$ from the first, second and third equation in (4.8), respectively. A second key observation is that, thanks to the size of n_2 (and because essentially any random matrix over $\mathbb{F}_{q^m}$ has full rank), we can find three sets $S_1, S_2, S_3 \subset [\ell+1, n_2]$ of cardinality ℓ such that $\overline{M}_{A_1} = M_{A_1}[:, S_1], \overline{M}_{B_1} = M_{B_1}[:, S_1], \overline{M}_{A_2} = M_{A_2}[:, S_2], \overline{M}_{C_2} = M_{C_2}[:, S_2], \overline{M}_{B_3} = M_{B_3}[:, S_3], \overline{M}_{C_3} = M_{C_3}[:, S_3]$ are all invertible $\ell \times \ell$ matrices. Denote by $\overline{\mathbf{c}}_1, \overline{\mathbf{c}}_2$ and $\overline{\mathbf{c}}_3$ the subvectors of $\mathbf{c}_1, \mathbf{c}_2$ and $\mathbf{c}_3$ of consisting of entries indexed by S_1, S_2 and S_3, respectively.

$$\begin{aligned} \overline{\mathbf{c}}_1 &= \mathbf{e}_1 \overline{M}_{A_1} + \mathbf{e}_2 \overline{M}_{B_1} \\ \overline{\mathbf{c}}_2 &= \mathbf{e}_1 \overline{M}_{A_2} + \mathbf{p}\overline{M}_{C_2} \\ \overline{\mathbf{c}}_3 &= \mathbf{e}_2 \overline{M}_{B_3} + \mathbf{p}\overline{M}_{C_3} \end{aligned} \tag{4.9}$$

Setting $M_{\mathbf{p}} := \overline{M}_{C_2}\overline{M}_{A_2}^{-1}\overline{M}_{A_1}\overline{M}_{B_1}^{-1} + \overline{M}_{C_3}\overline{M}_{B_3}^{-1}$ and $\mathbf{c_p} = \overline{\mathbf{c}}_2\overline{M}_{A_2}^{-1}\overline{M}_{A_1}\overline{M}_{B_1}^{-1} - \overline{\mathbf{c}}_1\overline{M}_{B_1}^{-1} + \overline{\mathbf{c}}_3\overline{M}_{B_3}^{-1}$, one can solve

$$\mathbf{p}M_{\mathbf{p}} = \mathbf{c_p}$$

for $\mathbf{p}$. Substituting $\mathbf{p}$ into (4.9) we recover $\mathbf{e}_1$ and $\mathbf{e}_2$.
We implemented this attack in SageMath. An average of the time required, on a Linux Mint virtual machine, to recover the plaintext for the proposed parameters is given in table 4.4.

Security	n_E	Time (s)
128	17	11.66
192	21	16.32

Table 4.4: Average time in seconds (on 50 samples for each security level) needed to recover a plaintext.

Alex Pellegrini announced this attack on 22 Oct in [Pel23b] breaking two out of three security levels of the 20231020-version of Layered ROLLO-I, announced two days prior.

Remark 4.2.2. *This attack does not apply to the 256-bit parameters. For the parameters of any security level we always have that* $3(n_2 - \ell) > n_2$ *where there exist at most* n_2 *linearly independent equations in the system (4.8) because there are only* n_2 *variables. For level 128 and 192 we have* $3\ell < n_2$ *ensuring unique solution of the system, which is not the case for security level 256.*

Chapter 5

NTRU+: Compact Construction of NTRU Using Simple Encoding Method

NTRU+ [KP22] is a lattice-based submission to the KpqC competition. It builds on the NTRU system introduced by Hoffstein, Pipher and Silverman in 1998 [HPS98] and many of the improvements since, in particular the NTTRU system introduced by Lyubashevsky and Seiler in 2019 [LS19]. We have analyzed this system in detail (see below for a summary), for a more in detail analysis see the bachelor thesis of Luc Steenbakkers [Ste23].

5.1 System description

NTRU+ works with three rings

$$\begin{aligned} R &= \mathbf{Z}[x]/(x^n - x^{n/2} + 1) \\ R_q &= (\mathbf{Z}/q\mathbf{Z})[x]/(x^n - x^{n/2} + 1) \\ R_3 &= (\mathbf{Z}/3\mathbf{Z})[x]/(x^n - x^{n/2} + 1), \end{aligned}$$

where n and q are integers with $\gcd(q, 3) = 1$ and $n = 2^i 3^j$, with $i, j > 0$ to ensure n is even. This is a large deviation from the original NTRU, and follows the NTTRU paper by choosing a ring and an n and q such that the Number Theoretic Transform (NTT) can be computed efficiently, but avoid some of the concerns with subfield attacks.
The key generation goes as follows. The sparse polynomials f' and g are generated using the Centered Binomial Distribution, which means the probability for a coefficient to be -1 or 1 is $1/4$ for both, and the probability for

a 0 is 1/2. Using this distribution approximates a narrow discrete Gaussian distribution.
The secret key is then created as $f = 3f' + 1$ and g. This shape of f goes back to a paper by Hoffstein and Silverman [HS01] and is also used in NTTRU. It gives the benefit of saving one division by f mod 3 in the decryption/decapsulation process. This does mean q needs to be somewhat larger to avoid decryption failures since the coefficients of f are larger.
The public key then is computed as $h = 3g/(3f' + 1) \bmod q$.
In encapsulation a random message $m^+ \in \{0,1\}^n$ is sampled. The shared symmetric key K and the randomness r are then generated by hashing m^+. Then m is created as an element of $\{-1,0,1\}^n$ using the Semi-generalized One-Time pad (SOTP) introduced by the authors. So $m = \text{SOTP}(m^+, \text{hash}(r))$, which basically splits up the hash of r into $u_1||u_2$ and then finds $m = (m^+ \oplus u_1) - u_2$. Afterwards the encryption of the message m is similar as in NTRU:

$$c = r \cdot h + m \in R_q.$$

Decapsulation starts with the following calculation:

$$m' = (f \cdot c \bmod^{\pm} q) \bmod^{\pm} 3,$$

where $\bmod^{\pm}$ indicates that the set of representatives of the residue classes is taken centered around 0, e.g., $\bmod^{\pm} 3$ means a result in $\{-1,0,1\}$. Note that the mixing of moduli can lead to decryption errors here. Then r' is obtained by $r' = (c - m') \cdot h^{-1}$. If $m = m'$ and $r = r'$, then using m' and r' as input to the inverse function of the SOTP, named Inv, will return the message m^+ (by splitting up the hash of r' again into u_1' and u_2' and computing $m'^+ = (m' + u_2') \oplus u_1'$). Given m'^+ the user can compute $\text{hash}(m'^+)$ and compare this to r'. If encapsulation was done correctly and no decryption error appeared, $m'^+ = m^+$ and thus $r' = \text{hash}(m'^+)$. If this matches, the user can compute K.

5.2 Security considerations

We analyzed this system to check applicability of known attacks.

Key-search attacks

Odlyzko's meet-in-the-middle attack for approximate collisions [HGSW03] and the low-memory adaptation to golden-collision searches by van Vreden-daal [van16] are generic attacks on NTRU-style systems that exploit the sparsity of the involved polynomials. These attacks apply directly to NTRU+.

The more recent faster version by May [May21] is described for RLWE but also applies to NTRU. The choice of parameters by the authors seem sufficient to be secure from these three attacks.
The choice of polynomial of NTRU+ also influences the security. Since the coefficients that overflow to higher powers of the polynomial than n are sent to two places in the polynomial reduction, there is mixing of coefficients. Where normal NTRU has $2n$ equivalent keys, coming from $(\pm x^i f, \pm x^i g)$, for NTRU+ only some of these have the correct sparsity to work. The exact number of equivalent keys depends on the degrees of f and g and on the smallest power of x that has a non-zero coefficient in one of these polynomials as well as the required sparsity of the private polynomials to avoid decryption failures. It is safe to bound the number of equivalent keys by $2n$.

Structural attacks

We have also looked at some of the structural attacks, mostly on the lattice structure.

Evaluate-at-one attack

This attack exploits that the polynomial $x^n - 1$ used in original NTRU has $x - 1$ as factor. The ring polynomial in NTRU+ is chosen to not have any polynomial factor over the integers and thus avoids this attack by design.

Generic lattice attacks

Table 5.1 contains the results of the estimator for the BKZ lattice attacks from Albrecht, Curtis, Deo, Davidson, Player, Postlethwaite, Virdia and Wunderer found in [ACD+18]. For more information see Section 2.1. The input of the code is given in Table 5.2.
Key search and lattice attacks combine into hybrid attacks as described by Howgrave-Graham [How07]. This is an interesting field of ongoing research and we have not found a faster attack.

5.2.1 Reaction attacks

The NTRU+ scheme uses a CCA-II conversion to avoid reaction attacks. As noted below, the scheme uses a non-standard transformation that has recently been shown to be insecure by Lee [Lee23a].
In principle this conversion should protect against reaction attacks, a type of attack going back to 1999 by Hall, Goldberg, and Schneier [HGS99] that

Parameter n	576	768	864	1152
Type	primal	primal	primal	primal
Q-Core-Sieve	104	149	171	273
Q-Core-Sieve + O(1)	120	165	187	289
Q-Core-Sieve (min space)	117	167	192	306
Q-β -Sieve	113	158	180	283
Q-8d-Sieve + O(1)	134	178	201	303
Core-Sieve	115	164	189	300
Core-Sieve + O(1)	131	180	205	316
Core-Sieve (min space)	144	206	237	378
β-Sieve	123	173	198	310
8d-Sieve + O(1)	144	194	219	331
Q-Core-Enum + O(1)	124	200	242	447
Lotus	127	216	264	508
Core-Enum + O(1)	248	399	482	887
8d-Enum (quadratic fit) + O(1)	276	460	564	1071

Table 5.1: Estimations for security level found with code by [ACD+18]

n	576	768	864	1152
Standard deviation	$\sqrt{288/576}$	$\sqrt{384/768}$	$\sqrt{432/864}$	$\sqrt{576/1152}$
q	3457	3457	3457	1152
Secret distribution	$(-1,1),288$	$(-1,1),384$	$(-1,1),432$	$(-1,1),576$
m	576	768	864	443
Norm f	$\sqrt{288\cdot 3^2}$	$\sqrt{384\cdot 3^2}$	$\sqrt{432\cdot 3^2}$	$\sqrt{576\cdot 3^2}$
Norm g	$\sqrt{288}$	$\sqrt{384}$	$\sqrt{432}$	$\sqrt{576}$
Claimed security	115	164	188	264
Category	1	1	3	5
Ring	$x^n-x^{n/2}+1$	$x^n-x^{n/2}+1$	$x^n-x^{n/2}+1$	$x^n-x^{n/2}+1$

Table 5.2: Input for the code by [ACD+18]

recovers private keys from information about failed decryptions of maliciously modified ciphertexts.
In the thesis of Steenbakkers [Ste23] we have worked out details on how one could modify ciphertexts to recover the key if the CCA transform as not deployed. We also worked out solutions if the message space is restricted to $\{0, 1\}$ (omitting -1). None of these are applicable to NTRU+ in its current form though, since the transform indeed protects against these.
Lee [Lee23a] however has shown that the IND-CCA property is violated. As a reminder the IND-CCA game goes as follows. The attacker creates two messages m_0 and m_1 which he sends to the challenger. The challenger randomly chooses one of the messages, encrypts it and sends the ciphertext c to the attacker. Then the IND-CCA game is broken if the attacker has a more than $1/2$ chance of choosing the encrypted message from m_0 and m_1, while having access to a decryption oracle and not being allowed to decrypt c.
Lee's attack basically exploits the fact that when looking at the Inv map of the SOTP that $m + u_2$ need not be binary if you tamper with the ciphertext. But in the implementation $m + u_2$ is reduced modulo 2, so a tampered ciphertext can be decrypted properly.
The important part here is that there are options to tamper with the ciphertext, specifically adding 2 to a certain term, by changing both it and its corresponding message. Then

$$\begin{aligned} r' &= (c' - m') \cdot h^{-1} \\ &= (c + 2 - (m + 2) \cdot h^{-1} \\ &= (c - m) \cdot h^{-1} \end{aligned} \tag{5.1}$$

So since we can now add 2 to a certain ciphertext and its message we can input $c' = c+2$ and $m' = m+2$ into the Inv, with the correct r' for both m, c and m', c'. If the first coefficient of m is equal to -1 and of u_2 is equal to 1, adding 2 will get us the same m^+, but we can put c' in our decryption oracle. Thus in about $1/8$ of the cases for one coefficient we can create a ciphertext unequal to c, but will decrypt to the original m_0 or m_1. There are of course many coefficients to play with, so statistically speaking this is broken.
The authors of NTRU+ changed their Inv function to fight this attack, by putting in a check that $m + u_2$ must be binary or it will output a decryption error. Note that this check does present new options for reaction attacks, but we haven't been able to exploit this yet.

5.3 Implementation considerations

On Linux systems the reference implementation failed, but a small tweak (adding `return 0` to the `crypto_kem_enc` functions) made the code compile and the outputs match the KAT files.
Like its relatives, NTRU+ has the relatively small ciphertexts and public keys of lattice-based systems. Decapsulation is faster because of the choice of $f = 3f' + 1$. Key generation is slower than for other lattice-based systems due to the NTRU quotient (rather than product NTRU) structure in the public key $h/3g/f$.

5.4 Provable security

The authors of NTRU+ [KP22] chose to use a two-stage CCA-II transformation instead of deploying existing adaptations [DHK+21, SXY18, HHK17] of the Fujisaki–Okamoto transform [FO99]. A first step checks that the plaintext is in $\{0,1\}$ and then a (relatively standard) transformation is used.
The attack by Lee [Lee23a], explained in the section on reaction attacks, shows that, at least in the reference implementation, this step leads to a practical attack. Following an initial misunderstanding the NTRU+ submitters have now acknowledged the attack and that it also applies on the specification.
One of the more puzzling parts is the fact that somewhere in the proof for IND-CCA security there needs to be a flaw, since this attack exists. We expect the flaw to be in the fact that for the proof they use a flawed definition of rigidity, which only holds for valid ciphertexts as input. They miss the fact that any element outside of the co-domain of the encryption function should not be decryptable. This is something that definitely needs the attention of the authors. We also note that the injectivity definition seems to be flawed since it all-quantifies over all involved variables, thus eradicating the probability space.
Another potential point of attention is that attackers can gather some information about the involved secret key by crafting ciphertexts that fail to decrypt [BS20, DRV20]. To mitigate such attacks, NTRU+ involves an additional step that aims to push the chance of crafting such ciphertexts out of reach. We were not able, however, to verify that the additional step indeed realizes this goal since we found an – albeit very artificial – counterexample to the claim by linking worst-case decryption failures with some specially crafted $\mathsf{G}(r)$. This did not translate into an attack, but the step needs reconsideration.

Chapter 6

PALOMA: Binary Separable Goppa-based KEM

PALOMA [KJKK22] is a code-based KEM, based on binary Goppa codes. While PALOMA is close to the NIST submission Classic McEliece [ABC+22] and references it frequently, there are several differences.
The Goppa polynomial g is chosen to split completely over $\mathbf{F}_{2^m}$ while it is chosen to be irreducible in Classic McEliece. This means that the support and the t roots of g need to share $\mathbf{F}_{2^m}$ and the parameters are chosen so that $n + t < 2^m$ and for the given parameters this is a strict inequality. There are also some other differences in how the system achieves CCA security.
Below, we first discuss cryptanalysis results regarding PALOMA. Afterwards, we discuss the provable security claims made in the PALOMA specification and implementation considerations. Finally, we give our general assessment.

6.1 System description

Let $q = 2^m$. A binary Goppa code is defined by

- a list $L = (\alpha_1, \ldots, \alpha_n)$ of n distinct elements in $\mathbf{F}_q$, called the *support*.
- a square-free polynomial $g(x) \in \mathbf{F}_q[x]$ of degree t such that $g(\alpha_i) \neq 0$ for all $1 \leq i \leq n$. This $g(x)$ is called the *Goppa polynomial*.

The corresponding binary Goppa code $\Gamma(L, g)$ is

$$\left\{ \mathbf{c} \in \mathbf{F}_2^n \,\middle|\, S(\mathbf{c}) = \frac{c_1}{x - \alpha_1} + \frac{c_2}{x - \alpha_2} + \cdots + \frac{c_n}{x - \alpha_n} \equiv 0 \bmod g(x) \right\}$$

- This code $\Gamma(L, g)$ has length n, dimension $k \geq n - mt$ and minimum distance $d \geq 2t + 1$.

The parity-check matrix of this code is

$$H = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \alpha_3^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & 0 & \ldots & 0 \\ 0 & \frac{1}{g(\alpha_2)} & 0 & \ldots & 0 \\ 0 & 0 & \frac{1}{g(\alpha_3)} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{1}{g(\alpha_n)} \end{pmatrix}$$

PALOMA chooses $g(x) = \prod_{\alpha \in T}(x - \alpha)$ for $T \subseteq \mathbf{F}_q \setminus \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ with $|T| = t$. Hence, $g(x)$ splits completely over $\mathbf{F}_q$. Classic McEliece chooses g irreducible over $\mathbf{F}_q$. This difference impacts what decoding algorithms are readily available and the submission puts significant effort into extending Patterson's decoding algorithm to deal with reducible g.
All secrets in the PALOMA KeyGen are derived from the main secret, a string r:

1. $(\alpha_1, \alpha_2, \ldots, \alpha_q) = \textsc{shuffle}_r(\mathbf{F}_q)$
2. $L = (\alpha_1, \alpha_2, \ldots, \alpha_n),\ T = (\alpha_{n+1}, \alpha_{n+2}, \ldots, \alpha_{n+t})$.
3. Compute $g = \prod_{\alpha \in T}(x - \alpha)$ and corresponding parity-check matrix H.
4. Try to bring H to systematic form, GOTO 1 if this fails.

H can be put in systematic form with 29% probability (no change compared to Classic McEliece). For PALOMA, any choice of T leads to valid g; Classic McEliece has a procedure to construct an irreducible g that succeeds almost always.
Binary Goppa codes are alternant codes and can be decoded efficiently using the Patterson decoder [Pat75], however, this decoder requires computing the inverse of the syndrome polynomial modulo g and there is no reason a-priory that 4his polynomial should be co-prime to g in the case considered in PALOMA. The submission thus develops a version of Patterson decoding that is suitable for this choice of g. We first describe the regular version and then point out where changes were needed.
To decode the received vector $\mathbf{x} = \mathbf{c} + \mathbf{e}$ to $\mathbf{c} \in \Gamma(L, g)$, first compute the syndrome polynomial

$$s(x) = \sum_{i=1}^{n}(c_i + e_i)/(x - \alpha_i) \equiv \left(\sum_{i=1}^{n} e_i \prod_{j \neq i}(x - \alpha_j)\right) / \prod_{i=1}^{n}(x - \alpha_i) \bmod g(x).$$

If $\mathbf{e} \neq 0$ this polynomial $s(x) \neq 0$ by definition of the code.
Put $f(x) = \prod_{i=1}^{n}(x - \alpha_i)^{e_i}$ with $e_i \in \{0, 1\}$. Then, using the chain and product rules on derivatives, $f'(x) = \sum_{i=1}^{n} e_i \prod_{j \neq i}(x - \alpha_j)^{e_j}$. Thus $s(x) \equiv f'(x)/f(x) \bmod g(x)$.
Split $f(x)$ into odd and even terms: $f(x) = A^2(x) + xB^2(x)$ and observe that over binary fields $f'(x) = B^2(x)$ as all even powers of x have derivative 0.
Thus $B^2(x) \equiv f(x)s(x) \equiv (A^2(x) + xB^2(x))s(x) \bmod g(x)$ which Patterson normally transforms by dividing by $s(x) \bmod g(x)$. However, as mentioned above, $s(x)$ need not be co-prime to $g(x)$ if g is reducible. If $g(x)$ is irreducible it is possible to compute $B^2(x)(x + 1/s(x)) \equiv A^2(x) \bmod g(x)$ and to recover A and B from a half-gcd computation on the polynomials $v(x) \equiv \sqrt{x + 1/s(x)}$ and $g(x)$. At every step $A(x) = B(x)v(x) + h(x)g(x)$ and the half-gcd computation stops when the degrees of A and B are balanced: $\deg(A) \leq \lfloor t/2 \rfloor, \deg(B) \leq \lfloor (t-1)/2 \rfloor$.
PALOMA designs a new version of Patterson's decoder for reducible g, dealing with $\gcd(g, s) \neq 1$.
Let $s^* = 1 + xs$ and $g_1 = \gcd(g, s), g_2 = \gcd(g, s^*)$. By construction, s and s^* are co-prime. They then define polynomials g_1, g_2 with $g = g_1 g_2$, do the computation modulo g_1 and g_2 separately and eventually combine the congruences using the Chinese Remainder Theorem. This makes it possible to use a Patterson-style decoder for reducible g at the expense of splitting the computation.
To hide timing information on secret g they would need to use s and s^* of maximum-possible degree t and thus require more work and extra effort to hide the actual degrees.
So far, protection against timing attacks is not implemented but decoding is already very slow (see later section).
Classic McEliece uses a Berlekamp-Massey decoder instead of the Patterson decoder. This decoder does not require g to be irreducible and could thus be used for the g in PALOMA.
The main observation for using this decoder is that $\mathbf{c} \in \Gamma(L, g)$ implies $\mathbf{c} \in \Gamma(L, g^2)$. This fact is normally shown when proving that the minimum distance is at least $2t + 1$.
Let $\mathbf{c} \in \Gamma(L, g)$, then

$$s(x) = \sum_{i=1}^{n} c_i/(x - \alpha_i) = \left(\sum_{i=1}^{n} c_i \prod_{j \neq i}(x - \alpha_j) \right) / \prod_{i=1}^{n}(x - \alpha_i) \equiv 0 \bmod g(x).$$

over $\mathbf{F}_{2^m}$:

$$(f_{2i+1}x^{2i+1})' = f_{2i+1}x^{2i}, \quad (f_{2i}x^{2i})' = 0 \cdot f_{2i}x^{2i-1} = 0,$$

thus

$$f'(x) = \sum_{i=0}^{(w-1)/2} f_{2i+1} x^{2i} = \left(\sum_{i=0}^{(w-1)/2} \sqrt{f_{2i+1}} x^i \right)^2 = F^2(x).$$

Having $s(x) \equiv F^2(x)/f(x) \equiv 0 \bmod g(x)$ for squarefree g means $g|F$, thus $g^2|F^2$.
Let $\mathbf{c} \in \Gamma(L, g^2)$, then $s \equiv 0 \bmod g^2 \Rightarrow s \equiv 0 \bmod g$ holds obviously.

The Berlekamp–Massey decoder can be used for any generalized Reed-Solomon code and is used here for $\Gamma(L, g^2)$. This algorithm goes back to computing the feedback polynomial for an LFSR given twice as many output bits as the state length.

1. Let $\mathbf{v} = \mathbf{c} + \mathbf{e}$. Then

$$B(x) = \sum_{i=1}^{n} \frac{v_i}{g^2(\alpha_i)} \prod_{j \neq i} (x - \alpha_j).$$

   Then $B(\alpha_i) = v_i \left(\prod_{j \neq i} (\alpha_i - \alpha_j) \right) / g^2(\alpha_i)$.

2. Put $A(x) = \prod_i (x - \alpha_i)$.

3. Use Berlekamp-Massey to compute approximant b/a to B/A such that $\gcd(a, b) = 1, \deg(a) \leq t, \deg(b) < t$, and $\deg(aB - bA) < \deg(A) - t$.

4. If a divides A, compute $f = B - bA/a$ and $\mathbf{v} = (B(\alpha_1) - f(\alpha_1), B(\alpha_2) - f(\alpha_2), \ldots, B(\alpha_n) - f(\alpha_n))$

Classic McEliece chooses the Berlekamp-Massey decoder for ease of safe implementation (correctness and constant timeness). The same approach could work for PALOMA and would probably be faster than their adaptation of Patterson.
See https://cr.yp.to/papers.html#goppadecoding for a full explanation of the Berlekamp–Massey decoder for coding Goppa codes.

6.2 Security considerations

For most purposes, the choice of g between PALOMA and Classic McEliece to does not matter. While limiting g to polynomials that split completely over $\mathbf{F}_{2^m}$ limits the key space, this number is so large that it is not even close to the

fastest known attacks to recover messages. Within the keyspace covered by PALOMA there are several equivalent codes and an ongoing research project is to see whether there are proportionally larger equivalence classes. This work is currently under development by Lorenz Panny, a former TU/e PhD student.
A concern expressed in [For18] in comparing choices for NTS-KEM and Classic McEliece is that not irreducible choices of g can bring structural or algebraic attacks into reach. Fur algebraic attacks we checked [FOPT10, COT14, EM22, CMT23] and several more. Note, again, that the public key is a hidden parity-check matrix SH of $\Gamma(L, g)$ for

$$H = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \alpha_3^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & 0 & \ldots & 0 \\ 0 & \frac{1}{g(\alpha_2)} & 0 & \ldots & 0 \\ 0 & 0 & \frac{1}{g(\alpha_3)} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{1}{g(\alpha_n)} \end{pmatrix}$$

Let $M = (m_{ij})$ be the generator matrix of the same code, then $SHM^T = 0$, hence

$$\sum_{j=1}^{n} m_{ij}\alpha_j^u / g(\alpha_j) = 0,\ 1 \le i \le k,\ 0 \le u \le t-1.$$

Use variables X_j for the unknown $\alpha_j \in \mathbf{F}_{2^m}$ and Y_j for the unknown $1/g(\alpha_j) \in \mathbf{F}_{2^m}$ for $1 \le j \le n$. Using Gröbner basis computations to solve the system $\sum_{j=1}^{n} m_{ij} X_j^u Y_j = 0$ requires high rate (dimension divided by length of the code) and does not make use of g (Y is general). This means, that the difference of PALOMA and Classic McEliece is ignored in the modeling. None of the attacks we considered gave an improvement over message-recovery attacks and we have not yet found a modeling that makes use of the structure of g.
See Table 6.1 for our estimates of the security levels of PALOMA-128, PALOMA-192, and PALOMA-256.

6.3 Implementation considerations

The reference implementation works and produces the correct KAT files.
PALOMA shares with Classic McEliece the large public keys and small ciphertexts. The PALOMA private key includes a matrix even through decoding algorithms for binary Goppa codes do not require this.

	doc	isd0	isd1	isd2
PALOMA-128	166.21	162.84	159.76	153.74
PALOMA-192	267.77	245.67	241.94	229.63
PALOMA-256	289.66	277.20	272.80	255.45

Table 6.1: PALOMA attack costs in $\log_2$(bit operations). The "isd0" column is the 0-level ISD (Prange + Lee–Brickell + Leon) cost predicted by CryptAttackTester. The "isd1" column is the 1-level ISD (Stern + Dumer) cost predicted by CryptAttackTester. The "isd2" column is the 2-level ISD (MMT + BJMM) cost predicted by CryptAttackTester. CryptAttackTester does not account for the costs of long-distance communication; the real-world speedup from isd1 to isd2 is smaller than the bit-operation speedup in the table. Computing this table took about a day on a dual AMD EPYC 7742: e.g., CryptAttackTester used 2800 core-hours searching many attack parameters for PALOMA-256. For comparison, the "doc" column is BJMM-ISD bit operations estimated on page 39 of the PALOMA documentation.

A benefit in key generation is that g can be sampled by sampling t random elements in $\mathbf{F}_{2^m}$ and defining them as roots of g. However, operations requiring arithmetic in $\mathbf{F}_{2^m}$ are more expensive as this field is relatively larger than for Classic McEliece.

While PALOMA advertises better speeds than Classic McEliece, the available implementation is much slower in our benchmarks.

See Table 6.2 for our benchmarks of the PALOMA software.

The implementation of decapsulation chose to use a new adaptation of Patterson's algorithm which seems slower than a direct reuse of the Berlekamp–Massey implementation in Classic McEliece. That system chose not to use Patterson as a constant-time implementation it would likely be less efficient than Berlekamp—Massey and PALOMA's adaptation of Patterson in parts doubles the work. While this makes the benchmarks look very slow this choice can be modified and the faster code from Classic McEliece could be used.

The implementation is additionally slower than necessary in the form of plaintext confirmation chosen to involve generating a matrix from a seed and doing a matrix multiplication. A simpler hash function call should suffice and would also safe space in addition to time.

While we understand the rationale for choosing split g and have not been able to show any security degradation from this choice, we do not think the other choices (Patterson, use of S, e, e') are beneficial for speed (they look detrimental to it) nor do they achieve better security compared to alternatives.

	cycles	ms	doc
PALOMA-128 init	10342016	5.011	
PALOMA-128 keypair	154491948	74.851	64.00
libmceliece 348864f keypair	71569974	34.675	
PALOMA-128 enc	406522	0.197	0.03
libmceliece 348864f enc	19178	0.009	
PALOMA-128 dec	19770110	9.579	9.00
libmceliece 348864f dec	235124	0.114	
PALOMA-192 init	10329976	5.005	
PALOMA-192 keypair	646889506	313.415	261.00
libmceliece 460896f keypair	218800512	106.008	
PALOMA-192 enc	819752	0.397	0.04
libmceliece 460896f enc	40764	0.020	
PALOMA-192 dec	122390384	59.298	59.00
libmceliece 460896f dec	651966	0.316	
PALOMA-256 init	10331352	5.005	
PALOMA-256 keypair	630017080	305.241	323.00
libmceliece 6960119f keypair	368109154	178.347	
PALOMA-256 enc	1000180	0.485	0.04
libmceliece 6960119f enc	76454	0.037	
PALOMA-256 dec	123652778	59.909	60.00
libmceliece 6960119f dec	682152	0.331	

Table 6.2: Measurements of PALOMA vs. libmceliece speed, Apple M1, Icestorm core, gcc 13.2.0. The "cycles" column is the median of 31 measurements of the PALOMA software and of libmceliece. The "ms" column is milliseconds calculated from cycles and 2.064GHz clock speed. For comparison, the "doc" column is milliseconds reported on page 30 of the PALOMA documentation. The PALOMA documentation does not specify 3.2GHz Firestorm cores vs. 2.064GHz Icestorm cores, but the Icestorm dec measurements are a good match. The PALOMA documentation also reports milliseconds for an Intel Core i5; libmceliece uses 256-bit vectors on Intel and uses portable code on the M1, so the speed gap will be larger on Intel.

Slowdowns due to their generalization of Patterson's decoder can be eliminated using the same Berlekamp-Massey decoder as for Classic McEliece. However, there are some other quirks in PALOMA that make it slower than explained:

- KeyGen shuffles the columns of H even though the order in L was random already.
- The secret key includes S with $\hat{H} = SH$ in systematic form. They comment that key size could be saved as S depends on r, but miss that S is not necessary at all in decoding. Note that syndrome $\mathbf{s} \in \mathbf{F}_2^{n-tm}$ expands to noisy codeword $\mathbf{v} = \mathbf{s}00\ldots0$ for $\hat{H}$ in systematic form.

It would thus be easy to decrease the size of the private key by skipping S. We understand that, just like for Classic McEliece, PALOMA does not use only the seed as private key but keep some parts expanded. This makes KeyGen and decap a lot faster than if the full expansion was needed.
Further unexplained slowdowns are the use of extra permutations in encapsulation and decapsulation.

6.4 Provable security claims

The PALOMA specification provides a provable security result: to construct an IND-CCA secure Key Encapsulation Mechanism, PALOMA uses its own modified version of the Fujisaki-Okamoto (FO) design paradigm [FO99, HHK17], the paradigm that was also used by most of the NIST proposals (including the winner Kyber).
The core idea of the FO paradigm is to use a given public-key encryption scheme to encrypt a randomly chosen message, from which a key can be derived by feeding the message (and sometimes some auxiliary information) into a hash function. Intuitively, this makes the key unpredictable unless the attacker can break the encryption algorithm. To make chosen-ciphertext attacks unfeasible, the encryption algorithm is modified in a certain way. This modification prevents that attackers can build dishonest ciphertexts that will be accepted by the decapsulation algorithm.

6.4.1 Security proof of PALOMA

At a first glance, PALOMA-KEM is constructed from PALOMA-PKE by applying a variant of the FO design paradigm. To argue IND-CCA security of PALOMA-KEM, the specification relatively briefly recalls the security reasoning for FO that was given in [HHK17].

There are, however, several gaps in the proof: PALOMA-KEM deviates from the established FO paradigm in several different ways (detailed below). The submission does not address at all that/why it introduces these modifications and if/why they do not decrease security.
While it is not straightforward to assess whether/how these deviations lead to an attack, each one on its own makes it impossible to apply the established security proof.
Undesirable dependencies in the decapsulation algorithm. To deal with chosen-ciphertext attacks, FO-KEMs react to dishonest ciphertexts by returning a pseudorandom value. This pseudorandom value is derived from a secret random seed which is part of the secret key (besides the secret key needed for decrypting). The FO paradigm picks this seed independently from the secret key used for decrypting ciphertexts, unlike PALOMA-KEM, which reuses a seed that was already used to generate the secret decryption key. This leads to undesirable leakage on the secret decryption key during decapsulations of dishonest ciphertexts, and leaves a gap in the security proof.
Plaintext permuting not covered by FO paradigm. Following the FO paradigm, one would expect that security of PALOMA-KEM is based on security of the encryption algorithm introduced as algorithm 6. Instead, it is based on a modification of algorithm 6, called algorithm 18, which introduces a permutation step. This leaves a gap in the security proof – to close this gap, it would have been necessary to show that security of algorithm 18 can be based on security of algorithm 6.
One possible explanation for this modification might be that the full FO design can only be applied to probabilistic schemes: it could be that the random sampling of the permutation matrix was introduced to make the encryption algorithm probabilistic. In this case, the submission could consider switching to FO-alternatives for deterministic schemes (e.g., [BHH+19]) instead.
Treatment of dishonest ciphertexts not covered by FO paradigm. As described above, dishonest ciphertexts are treated by FO-KEMs in a specific way to mitigate chosen-ciphertext attacks. To identify such dishonest ciphertexts, PALOMA-KEM deviates from the standard check imposed by the FO paradigm. This leaves a gap in the security proof – to close this gap, it would have been necessary to show that the alternative check performed by PALOMA-KEM is equivalent to the one that is imposed by the FO paradigm.
Additional gap - sampling of messages not covered by FO paradigm. FO-KEMs sample messages uniformly at random. This is needed to be able to base security of the KEM on security of the involved encryption algorithm – the security definition for the encryption algorithm assumes uniform messages. PALOMA-KEM instead samples messages using algorithm 13. This leaves a gap in the security proof – to close this gap, it would be necessary to

analyze whether algorithm 13 yields the required uniform distribution. (See section 5.5.4 of [ABC+22] as an example for such a discussion.)

6.4.2 Interpretation of provable security results

The formal security arguments discussed in subsection 6.4.1 do not seem sound, and have an additional significant shortcoming: the arguments only consider classical adversaries. This also means that the bounds do not apply against quantum adversaries that at least can gain a polynomial advantage using Grover.
A result might be obtained by switching to the design paradigm underpinning McEliece, and then using recent work on this design paradigm [BP18, BHH+19], but this would require a redesign of PALOMA.

6.5 General assessment

While several shortcomings with regard to the cryptanalytic security as well as the provable security were raised above, no vulnerabilities have been found.

Chapter 7

REDOG

This chapter analyzes the security of the REinforced modified Dual-Ouroboros based on Gabidulin codes, REDOG [KHL+22a], a public-key encryption system submitted to KpqC, the Korean competition on post-quantum cryptography. REDOG is a code-based cryptosystem using rank-metric codes, aiming at providing a rank-metric alternative to Hamming-metric code-based cryptosystems.

Rank-metric codes were introduced by Delsarte [Del78] and independently rediscovered by Gabidulin [Gab85] in 1985, who focused on those that are linear over a field extension. Gabidulin, Paramonov, and Tretjakov [GPT91] proposed their use for cryptography in 1991. The GPT system was attacked by Overbeck [Ove05, Ove08] who showed *structural* attacks, permitting recovery of the private key from the public key.

During the mid 2010s new cryptosystems using rank-metric codes were developed such as Ouroboros [DGZ17] and the first round of the NIST competition on post-quantum cryptography saw 5 systems based on rank-metric codes: LAKE [ABD+17a], LOCKER [ABD+17b], McNie [GKK+17], Ouroboros-R [AAB+17a], and RQC [AAB+17b]. For all these systems see NIST's Round-1 Submissions page. Gaborit announced an attack weakening McNie and the McNie authors adjusted their parameters. A further attack was published in [LT18] and NIST did not advance McNie into the second round of the competition.

ROLLO, a merger of LAKE, LOCKER and Ouroboros-R, and RQC made it into the the second round but got broken near the end of it by significant advances in the cryptanalysis of rank-metric codes and the MinRank problem in general, see [BBB+20] and [BBC+20a]. In their report at the end of round 2 [AASA+20], NIST wrote an encouraging note on rank-metric codes: "Despite the development of algebraic attacks, NIST believes rank-based cryptography should continue to be researched. The rank metric cryp-

tosystems offer a nice alternative to traditional hamming metric codes with comparable bandwidth." (capitalization as in the original).
Kim, Kim, Galvez, and Kim [KKGK21] proposed a rank-metric system in 2021 which was then analyzed by Lau, Tan, and Prabowo in [LTP21] who also proposed some modifications to the issues they found. REDOG resembles the system in [LTP21]. We will show that some of the issues

7.1 Preliminaries and background notions

This section gives the necessary background on rank-metric codes for the rest of the chapter.
Let $\{\alpha_1, \ldots, \alpha_m\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Write $x \in \mathbb{F}_{q^m}$ uniquely as $x = \sum_{i=1}^{m} X_i \alpha_i$, $X_i \in \mathbb{F}_q$ for all i. So x can be represented as $(X_1, \ldots, X_m) \in \mathbb{F}_q^m$. We will call this the *vector representation* of x. Extend this process to $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_{q^m}^n$ defining a map $\mathsf{Mat} : \mathbb{F}_{q^m}^n \to \mathbb{F}_q^{m \times n}$ by:

$$\mathbf{v} \mapsto \begin{bmatrix} V_{11} & V_{21} & \ldots & V_{n1} \\ V_{12} & V_{22} & \ldots & V_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{1m} & V_{2m} & \ldots & V_{nm} \end{bmatrix}.$$

Definition 7.1.1. *The rank weight of* $\mathbf{v} \in \mathbb{F}_{q^m}^n$ *is defined as* $\mathsf{wt}_R(\mathbf{v}) := \mathsf{rk}_q(\mathsf{Mat}(\mathbf{v}))$ *and the rank distance between* $\mathbf{v}, \mathbf{w} \in \mathbb{F}_{q^m}^n$ *is* $d_R(\mathbf{v}, \mathbf{w}) := \mathsf{wt}_R(\mathbf{v} - \mathbf{w})$.

Remark 7.1.2. *It can be shown that the rank distance does not depend on the choice of the basis of* $\mathbb{F}_{q^m}$ *over* $\mathbb{F}_q$. *In particular, the choice of the basis is irrelevant for the results in this document.*

When talking about the space spanned by $\mathbf{v} \in \mathbb{F}_{q^m}^n$, denoted as $\langle \mathbf{v} \rangle$, we mean the $\mathbb{F}_q$-subspace of $\mathbb{F}_q^m$ spanned by the columns of $\mathsf{Mat}(\mathbf{v})$.
For completeness, we introduce the Hamming weight and the Hamming distance. These notions will be used in our message recovery attack against REDOG's implementation.
The *Hamming weight* of a vector $\mathbf{v} \in \mathbb{F}_{q^m}^n$ is defined as $\mathsf{wt}_H(\mathbf{v}) := \#\{i \in \{1, \ldots, n\} \mid v_i \neq 0\}$ and the Hamming distance between vectors $\mathbf{v}, \mathbf{w} \in \mathbb{F}_{q^m}^n$ is defined as $d_H(\mathbf{v}, \mathbf{w}) := \mathsf{wt}_H(\mathbf{v} - \mathbf{w})$.
Let $D = d_R$ or $D = d_H$. Then an $[n, k, d]$-code C with respect to D over $\mathbb{F}_{q^m}$ is a k-dimensional $\mathbb{F}_{q^m}$-linear subspace of $\mathbb{F}_{q^m}^n$ with *minimum distance*

$$d := \min_{\mathbf{a}, \mathbf{b} \in C,\, \mathbf{a} \neq \mathbf{b}} D(\mathbf{a}, \mathbf{b})$$

and *correction capability* $\lfloor (d-1)/2 \rfloor$. If $D = d_R$ (resp. $D = d_H$) then the code C is also called a *rank-metric* (resp. *Hamming-metric*) code. All codes in this document are linear over the field extension $\mathbb{F}_{q^m}$.
We say that G is a *generator matrix* of C if its rows span C. We say that H is a *parity check matrix* of C if C is the right-kernel of H.
A very well-known family of rank metric codes are *Gabidulin codes* [Gab85], which have $d = n - k + 1$.
In this analysis we can mostly use these codes as a black box, knowing that there is an efficient decoding algorithm using the parity-check matrix of the code and decoding vectors with errors of rank up to $\lfloor (d-1)/2 \rfloor$.
We will, however, use that the parity-check matrix of a Gabidulin code over $\mathbb{F}_q$, is a Moore matrix.

Definition 7.1.3. *Let $\mathbb{F}_{q^m}$ be a finite field. A matrix $M \in \mathbb{F}_{q^m}^{k \times n}$ is a* Moore matrix *if each row is the q-th power of the previous one.*

This structural property was used in the structural attacks by Overbeck [Ove08] to find the secret Gabidulin code hidden in the GPT system [GPT91]. This structure is also used in the analysis of [KKGK21] in [LTP21]. For a more general definition and further details on Moore matrices in this cryptographic context, see [HTMR15].
A final definition necessary to understand REDOG is that of isometries.

Definition 7.1.4. *Consider vectors in $\mathbb{F}_{q^m}^n$. An* isometry with respect to the rank metric *is a matrix $P \in \mathsf{GL}_n(\mathbb{F}_{q^m})$ satisfying that $\mathsf{wt}_R(\mathbf{v}P) = \mathsf{wt}_R(\mathbf{v})$ for any $\mathbf{v} \in \mathbb{F}_{q^m}^n$.*

Obviously matrices $P \in \mathsf{GL}_n(\mathbb{F}_q)$ are isometries as $\mathbb{F}_q$-linear combinations of the coordinates of $\mathbf{v}$ do not increase the rank and the rank does not decrease as P is invertible. The rank does also not change under scalar multiplication by some $\alpha \in \mathbb{F}_{q^m}^*$: $\mathsf{wt}_R(\alpha\mathbf{v}) = \mathsf{wt}_R(\mathbf{v})$. Note that the latter corresponds to multiplication by $P = \alpha I_n$.
Berger [Ber03] showed that any isometry is obtained by composing these two options.

Theorem 7.1.5. *[Ber03, Theorem 1] The isometry group of $\mathbb{F}_{q^m}^n$ for the rank metric is generated by scalar multiplications by elements in $\mathbb{F}_{q^m}^*$ and elements of $\mathsf{GL}_n(\mathbb{F}_q)$. This group is isomorphic to the product group $\left(\mathbb{F}_{q^m}^*/\mathbb{F}_q^*\right) \times \mathsf{GL}_n(\mathbb{F}_q)$.*

7.2 System specification

This section introduces the specification of REDOG. We follow the notation of [LTP21], with minor changes.

The system parameters are positive integers $(n, k, \ell, q, m, r, \lambda, t)$, with $\ell < n$ and $\lambda t \leq r \leq \lfloor (n-k)/2 \rfloor$, as well as a hash function $\mathsf{hash} : \mathbb{F}_{q^m}^{2n-k} \to \mathbb{F}_{q^m}^{\ell}$.

KeyGen:

1. Select $H = (H_1 \mid H_2)$, $H_2 \in \mathsf{GL}_{n-k}(\mathbb{F}_{q^m})$, a parity check matrix of a $[2n-k, n]$ Gabidulin code, with syndrome decoder Φ correcting r errors.
2. Select a full rank matrix $M \in \mathbb{F}_{q^m}^{\ell \times n}$ and isometry $P \in \mathbb{F}_{q^m}^{n \times n}$ (w.r.t. the rank metric).
3. Select a λ-dimensional subspace $\Lambda \subset \mathbb{F}_{q^m}$, seen as $\mathbb{F}_q$-linear space, containing 1 and select $S^{-1} \in \mathsf{GL}_{n-k}(\Lambda)$; see Section 7.3 for the definition.
4. Compute $F = MP^{-1}H_1^T \left(H_2^T\right)^{-1} S$ and publish the public key $\mathsf{pk} = (M, F)$. Store the secret key $\mathsf{sk} = (P, H, S, \Phi)$.

Encrypt $(\mathbf{m} \in \mathbb{F}_{q^m}^{\ell}, \mathsf{pk})$

1. Generate uniformly random $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{F}_{q^m}^{2n-k}$ with $\mathsf{wt}_R(\mathbf{e}) = t$, $\mathbf{e}_1 \in \mathbb{F}_{q^m}^{n}$ and $\mathbf{e}_2 \in \mathbb{F}_{q^m}^{n-k}$.
2. Compute $\mathbf{m}' = \mathbf{m} + \mathsf{hash}(\mathbf{e})$.
3. Compute $\mathbf{c}_1 = \mathbf{m}'M + \mathbf{e}_1$ and $\mathbf{c}_2 = \mathbf{m}'F + \mathbf{e}_2$ and send $(\mathbf{c}_1, \mathbf{c}_2)$.

Decrypt $((\mathbf{c}_1, \mathbf{c}_2), \mathsf{sk})$

1. Compute $\mathbf{c}' = \mathbf{c}_1 P^{-1} H_1^T - \mathbf{c}_2 S^{-1} H_2^T = \mathbf{e}' H^T$ where the vector $\mathbf{e}' := (\mathbf{e}_1 P^{-1}, -\mathbf{e}_2 S^{-1})$.
2. Decode $\mathbf{c}'$ using Φ to obtain $\mathbf{e}'$, recover $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ using P and S.
3. Solve $\mathbf{m}'M = \mathbf{c}_1 - \mathbf{e}_1$. Output $\mathbf{m} = \mathbf{m}' - \mathsf{hash}(\mathbf{e})$.

Suggested parameters

We list the suggested parameters of REDOG for 128,192 and 256 bits of security, following [KHL+22a] submitted to KpqC.

Security parameter	$(n, k, \ell, q, m, r, \lambda, t)$
128	$(44, 8, 37, 2, 83, 18, 3, 6)$
192	$(58, 10, 49, 2, 109, 24, 3, 8)$
256	$(72, 12, 61, 2, 135, 30, 3, 10)$

Table 7.1: Suggested parameters; see [KHL+22a].

7.3 Incorrectness of decryption

This section shows that decryption typically fails for the version of REDOG specified in [KHL+22a, LTP21]. The novelty of this specification, compared to that introduced in [KKGK21], lies in the selection of the invertible matrix S^{-1} in Step 3, which is selected with the property that $S^{-1} \in \mathsf{GL}_{n-k}(\Lambda)$, where Λ is a λ-dimensional $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$. This method has been first proposed by Loidreau in [Loi17], but it appears to be incorrectly applied in REDOG. Before providing more details about this claim and proving the incorrectness of REDOG's decryption process, we will shed some light on the object $\mathsf{GL}_{n-k}(\Lambda)$. Unlike the notation suggests, this is not a group, but a potentially unstructured subset of $\mathsf{GL}_{n-k}(\mathbb{F}_{q^m})$ defined as follows:
Let $\{1, \alpha_2, \ldots, \alpha_\lambda\} \subset \mathbb{F}_{q^m}$ be a set of elements that are $\mathbb{F}_q$-linearly independent. Let $\Lambda \subset \mathbb{F}_{q^m}$ be the set of $\mathbb{F}_q$-linear combinations of these α_i's. This set forms an $\mathbb{F}_q$-linear vectorspace. Now, $S^{-1} \in \mathsf{GL}_{n-k}(\Lambda)$ is defined to mean that S is an invertible $(n-k) \times (n-k)$ matrix with the property that the entries of S^{-1} are elements of Λ. Note that such an S exists because $\lambda \geq 1$ by assumption. The REDOG documentation [KHL+22a] points out that this does not imply that $S \in \mathsf{GL}_{n-k}(\Lambda)$, hence, despite what the notation may suggest, $\mathsf{GL}_{n-k}(\Lambda)$ is not a group in general.
We continue by giving a proof, and an easy generalization for any q, of [Loi17, Proposition 1].

Proposition 1. *Let λ, t, n be positive integers such that $\lambda t \leq n$, $A \in \mathsf{GL}_n(\Lambda)$ where $\Lambda \subset \mathbb{F}_{q^m}$ is a λ-dimensional subspace of $\mathbb{F}_{q^m}$, and $\mathbf{x} \in \mathbb{F}_{q^m}^n$ with $\mathsf{wt}_R(\mathbf{x}) = t$. Then*

$$\mathsf{wt}_R(\mathbf{x}A) \leq \lambda t.$$

Proof. Let Γ be the subspace of $\mathbb{F}_{q^m}$ generated by the entries of $\mathbf{x} = (x_1, \ldots, x_n)$. Since Γ has dimension t, we can write $\Gamma = \langle y_1, \ldots, y_t \rangle$ with $y_i \in \mathbb{F}_{q^m}$. Similarly for Λ, we can write $\Lambda = \langle \alpha_1, \ldots, \alpha_\lambda \rangle$ with $\alpha_i \in \mathbb{F}_{q^m}$.

Express $\mathbf{x}A$ as

$$\mathbf{x}A = \left(\sum_{i=1}^{n} x_i A_{i,1}, \ldots, \sum_{i=1}^{n} x_i A_{i,n} \right).$$

Fix $j \in \{1, \ldots, n\}$. Then

$$(\mathbf{x}A)_j = \sum_{i=1}^{n} x_i A_{i,j} = \sum_{i=1}^{n} \left(\left(\sum_{h=1}^{t} x_{i,h} y_h \right) \left(\sum_{k=1}^{\lambda} A_{i,j,k} \alpha_k \right) \right),$$

with $x_{i,h}, A_{i,j,k} \in \mathbb{F}_q$. By rearranging the terms we obtain

$$(\mathbf{x}A)_j = \sum_{h=1}^{t} \sum_{k=1}^{\lambda} \left(\sum_{i=1}^{n} x_{i,h} A_{i,j,k} \right) y_h \alpha_k. \tag{7.1}$$

Therefore each entry of $\mathbf{x}A$ can be expressed as an $\mathbb{F}_q$-linear combination of the λt elements of the form $y_h \alpha_k$. □

We will now show that REDOG typically does not decrypt correctly. In order to do so, we need some preliminary results and tools. The proof of the next lemma uses some tools from combinatorics. It computes the probability of containing a basis of a vector space of dimension t over $\mathbb{F}_q$ by randomly selecting a t-tuple of its elements.

Lemma 7.3.1. *Let V be a t-dimensional subspace $V \subseteq \mathbb{F}_q^m$ and let $S \in V^s$ be a uniformly random s-tuple of elements of V. The probability $p(q, s, t)$ that $\langle S_i \mid i \in \{1, \ldots, s\} \rangle = V$ is*

$$p(q, s, t) = \begin{cases} 0 & if \quad 0 \leq s < t; \\ \sum_{i=0}^{t} \begin{bmatrix} t \\ i \end{bmatrix}_q (-1)^{t-i} q^{s(i-t)+\binom{t-i}{2}} & otherwise, \end{cases} \tag{7.2}$$

where $\begin{bmatrix} t \\ i \end{bmatrix}_q$ is the q-binomial coefficient, counting the number of subspaces of dimension i of $\mathbb{F}_q^t$, and $\binom{a}{b} = 0$ for $a < b$. In particular, this probability does not depend on m or on the choice of V, but only on its dimension.

Proof. Let $(\mathcal{P}, \subseteq)$ be the poset (partially ordered set) of subspaces of $\mathbb{F}_q^m$ ordered by inclusion. Recall that the Möbius function of $\mathcal{P}$, and of any finite poset, is defined, for $A, B \in \mathcal{P}$, as

$$\mu(B, A) = \begin{cases} 1 & \text{if } B = A, \\ -\sum_{C \mid B \subseteq C \subset A} \mu(B, C) & \text{if } B \subset A, \\ 0 & \text{otherwise.} \end{cases}$$

This is computed e.g. in [Sta11, Example 3.10.2] as

$$\mu(B,A) = \begin{cases} (-1)^k q^{\binom{k}{2}} & \text{if } B \subseteq A \text{ and } \dim(A) - \dim(B) = k, \\ 0 & \text{otherwise.} \end{cases} \tag{7.3}$$

We want to compute the function $f : \mathcal{P} \to \mathbb{N}$ defined as

$$f(A) = \#\left\{S \in \left(\mathbb{F}_q^m\right)^s \mid \langle S \rangle = A\right\}.$$

Clearly, if $s < \dim A$, there does not exist any s-tuple S spanning A, hence $f(A) = 0$, which gives the first case of (7.2). We can therefore restrict ourselves to the case $s \geq \dim A$. Define the auxiliary function $g : \mathcal{P} \to \mathbb{N}$ as

$$\begin{aligned} g(A) &= \sum_{B \subseteq A} f(B) \\ &= \#\left\{S \in \left(\mathbb{F}_q^m\right)^s \mid \langle S \rangle \subseteq A\right\} \\ &= |A|^s = q^{s \dim A}. \end{aligned}$$

Then by Möbius inversion we can compute:

$$f(A) = \sum_{B \subseteq A} g(B)\mu(B,A). \tag{7.4}$$

Splitting the sum over the dimensions, and substituting the values in Equation 7.3, we can obtain

$$\begin{aligned} f(V) &= \sum_{i=0}^{t} \sum_{U \subseteq V,\, \dim U = i} g(U)\mu(U,V) \\ &= \sum_{i=0}^{t} q^{si}(-1)^{t-i} q^{\binom{t-i}{2}} \sum_{U \subseteq V,\, \dim U = i} 1 \\ &= \sum_{i=0}^{t} \begin{bmatrix} t \\ i \end{bmatrix}_q (-1)^{t-i} q^{si + \binom{t-i}{2}}. \end{aligned}$$

The probability can be computed by dividing $f(V)$ by the number of s-tuples of elements of V, that is, q^{st}.

□

Remark 7.3.2. *The probability given in Lemma 7.3.1 can be interpreted as the ratio of the number of surjective linear maps from $\mathbb{F}_q^s$ onto $\mathbb{F}_q^t$ over the total number of linear maps.*

We next compute the probability that by truncating a rank t vector, the rank stays the same.

Theorem 7.3.3. *Let $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{F}_{q^m}^{2n-k}$, with $\mathbf{e}_1 \in \mathbb{F}_{q^m}^{n}$ and $\mathbf{e}_2 \in \mathbb{F}_{q^m}^{n-k}$, be a uniformly random error with $\mathsf{wt}_R(\mathbf{e}) = t$. Then $\mathsf{wt}_R(\mathbf{e}_1) = t$ and $\mathsf{wt}_R(\mathbf{e}_2) = t$ with probability $p(q,n,t)/p(q,2n-k,t)$ and $p(q,n-k,t)/p(q,2n-k,t)$ respectively.*

Proof. By definition, the probability that $\mathsf{wt}_R(\mathbf{e}_1) = t$ is the ratio

$$\pi = \frac{\#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \mathsf{wt}_R(\mathbf{e}) = t \text{ and } \mathsf{wt}_R(\mathbf{e}_1) = t\}}{\#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \mathsf{wt}_R(\mathbf{e}) = t\}}. \tag{7.5}$$

We can split the cardinalities above over all the subspaces of $\mathbb{F}_q^m$ of dimension t as follows:

$$\pi = \frac{\sum_{V \subset \mathbb{F}_q^m,\, \dim V = t} \#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \langle \mathbf{e} \rangle = \langle \mathbf{e}_1 \rangle = V\}}{\sum_{V \subset \mathbb{F}_q^m,\, \dim V = t} \#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \langle \mathbf{e} \rangle = V\}}. \tag{7.6}$$

It is not hard to prove that the summands in (7.3) are independent of the space V. Therefore

$$\pi = \frac{\#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \langle \mathbf{e} \rangle = \langle \mathbf{e}_1 \rangle = V\}}{\#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \langle \mathbf{e} \rangle = V\}} = \frac{\#\{\mathbf{e}_1 \in \mathbb{F}_{q^m}^{n} \mid \langle \mathbf{e}_1 \rangle = V\}\, q^{t(n-k)}}{\#\{\mathbf{e} \in \mathbb{F}_{q^m}^{2n-k} \mid \langle \mathbf{e} \rangle = V\}},$$

where V is any subspace of $\mathbb{F}_q^m$ of dimension t. By applying Lemma 7.3.1 we then get

$$\pi = \frac{p(q,n,t)\, q^{nt} q^{t(n-k)}}{p(q,2n-k,t)\, q^{(2n-k)t}} = \frac{p(q,n,t)}{p(q,2n-k,t)},$$

as claimed. The probability for $\mathbf{e}_2$ can be computed with the same arguments as for $\mathbf{e}_1$. □

Remark 7.3.4. *In the context of a REDOG instance, the data q, n and t is fixed, hence, for the sake of reading simplicity, we denote the probability given in Theorem 7.3.3 by*

$$\bar{p}(r,t) = \frac{p(q,r,t)}{p(q,2n-k,t)}.$$

Example 7.3.5. *Consider the suggested parameters of REDOG for 128 bits of security from Table 7.1. Using SageMath [S+21] we computed the probability that $\mathsf{wt}_R(\mathbf{e}_1) = t$, that is*

$$\bar{p}(44,6) = 0.999999999996419,$$

and the probability that $\mathsf{wt}_R(\mathbf{e}_2) = t$, that is

$$\bar{p}(36,6) = 0.999999999083229.$$

We are ready to state the following theorem, which directly implies that REDOG's decryption process fails with extremely high probability.

Theorem 7.3.6. *Let (n, k, q, m, λ, t) be integers with $k < n < m$ and $\lambda t \leq m$. Let $\Lambda \subset \mathbb{F}_{q^m}$ be a λ-dimensional subspace of $\mathbb{F}_{q^m}$ and $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ as in Theorem 7.3.3. Let $P \in \mathbb{F}_{q^m}^{n \times n}$ be a random isometry matrix (w.r.t. the rank metric) and $S^{-1} \in \mathsf{GL}_{n-k}(\Lambda)$. Then $\mathbf{e}' := (\mathbf{e}_1 P^{-1}, -\mathbf{e}_2 S^{-1})$ has rank weight $\mathsf{wt}_R(\mathbf{e}') \geq \lambda t + 1$ with probability bounded from below by*

$$p_{\mathsf{fail}}(n, k, q, m, \lambda, t) := \bar{p}(n, t)\, \bar{p}(n - k, \lambda t)\, \bar{p}(n - k, t) \left(1 - \frac{\begin{bmatrix} \lambda t \\ t \end{bmatrix}_q}{\begin{bmatrix} m \\ t \end{bmatrix}_q}\right).$$

Proof. By Theorem 7.1.5, the isometry P is of the form $\alpha \bar{P}$ for $\alpha \in \mathbb{F}_{q^m}^*$ and $\bar{P} \in \mathsf{GL}_n(\mathbb{F}_q)$, where $q^m \gg q$ and thus typically $\alpha \notin \mathbb{F}_q$. Because of the multiplication by α^{-1}, we can assume that the linear transformation induced by P^{-1} takes a t-dimensional subvectorspace of $\mathbb{F}_q^m$ to a random t-dimensional subspace. Similarly we assume that S^{-1} sends a t-dimensional subspace of $\mathbb{F}_q^m$ to a random subspace of dimension at most λt, by Proposition 1. We get the lower bound on the failure probability by showing the following:

1. $\mathsf{wt}_R(\mathbf{e}_1 P^{-1}) = t$ with probability $\bar{p}(n, t)$;
2. $\mathsf{wt}_R(-\mathbf{e}_2 S^{-1}) = \lambda t$ with probability $\bar{p}(n - k, t)\bar{p}(n - k, \lambda t)$;
3. under the conditions in (1) and (2), $\langle \mathbf{e}_1 P^{-1} \rangle \not\subset \langle -\mathbf{e}_2 S^{-1} \rangle$ with probability
$$1 - \frac{\begin{bmatrix} \lambda t \\ t \end{bmatrix}_q}{\begin{bmatrix} m \\ t \end{bmatrix}_q}.$$

Note that (1) follows directly from Theorem 7.3.3 and the fact that P is an isometry of the space w.r.t the rank metric.
Likewise, $\mathsf{wt}_R(-\mathbf{e}_2) = t$ with probability $\bar{p}(n - k, t)$. The proof of Proposition 1 shows that for $\mathbf{e}_2$ with $\mathsf{wt}_R(-\mathbf{e}_2) = t$ we have that $-\mathbf{e}_2 S^{-1}$ is contained in a λt-dimensional subspace of $\mathbb{F}_q^m$. Again by Theorem 7.3.3 we obtain that $\langle -\mathbf{e}_2 S^{-1} \rangle$ spans the entire space with probability $\bar{p}(n - k, \lambda t)$, proving (2).
To prove (3) we will compute the opposite, i.e. the probability that $\langle \mathbf{e}_1 P^{-1} \rangle$ is a subspace of $\langle -\mathbf{e}_2 S^{-1} \rangle$. As mentioned at the beginning of the proof, we treat $\langle \mathbf{e}_1 P^{-1} \rangle$ as a random t-dimensional subspace of $\mathbb{F}_{q^m}$. Thus we can compute this probability as the ratio between the number of t-dimensional subspaces of $\langle -\mathbf{e}_2 S^{-1} \rangle$ and of $\mathbb{F}_q^m$, that is,

$$\frac{\begin{bmatrix} \lambda t \\ t \end{bmatrix}_q}{\begin{bmatrix} m \\ t \end{bmatrix}_q}.$$

Combining the probabilities and observing that (1 – 3) imply $\mathsf{wt}_R(\mathbf{e}') \geq \lambda t+1$ gives the result. □

Remark 7.3.7. *There are more ways to get* $\mathsf{wt}_R(\mathbf{e}') \geq \lambda t + 1$ *by relaxing the first two requirements in the proof of Theorem 7.3.6 and studying the dimension of the union in the third, but* p_{fail} *is large enough for the parameters in REDOG to prove the point.*

Remark 7.3.8. *The proof of property (3) relies on* $\mathbf{e}_1 P^{-1}$ *being a random subspace of dimension* t*. We note that for* $\alpha \in \mathbb{F}_q$ *we have* $\langle \mathbf{e}_1 \rangle = \langle \mathbf{e}_1 P^{-1} \rangle \subset \langle \mathbf{e}_2 S^{-1} \rangle$ *for* $S^{-1} \in \mathsf{GL}_{n-k}(\Lambda)$ *and* $1 \in \Lambda$*. The latter constraint is stated in [KHL+22a] and [LTP21] and it is possible that the authors were not aware of the full generality of isometries. See also [LPR23, Appendix B] for further observations on [LTP21] which are consistent with this misconception.*

Corollary 7.3.9. *Let* $(n, k, \ell, q, m, r, \lambda, t)$ *be the parameters of a instance of REDOG with* $r = \lambda t$*. Then REDOG will produce decryption failures with probability at least* $p_{\mathsf{fail}}(n, k, q, m, \lambda, t)$*.*

Proof. Recall that the decoder Φ can only correct errors up to rank weight $r = \lambda t$. By Theorem 7.3.6 we have that $\mathbf{e}'$ has rank weight $\geq \lambda t + 1$, hence producing decoding failure, with probability at least $p_{\mathsf{fail}}(n, k, q, m, \lambda, t)$. □

Note that a $[2n - k, n]$ Gabidulin code has minimum distance $d_R = 2n - k - n + 1 = n - k + 1$ and can thus correct at most $\lfloor (n - k)/2 \rfloor$ errors and that all instances of REDOG in Table 7.1 satisfy $\lfloor (n - k)/2 \rfloor = r = \lambda t$.

Example 7.3.10. *As in Example 7.3.5, consider the suggested parameters for 128 bits of security. Then Theorem 7.3.6 states that* $\mathsf{wt}_R(\mathbf{e}') \geq 19$ *with probability at least*

$$p_{\mathsf{fail}}(44, 8, 2, 83, 3, 6) = \bar{p}(44, 8)\bar{p}(36, 6)\bar{p}(36, 18)\left(1 - \frac{\begin{bmatrix} 18 \\ 6 \end{bmatrix}_2}{\begin{bmatrix} 83 \\ 6 \end{bmatrix}_2}\right)$$
$$= 0.999996184401789.$$

Table 7.2 reports the value of p_{fail} for each set of security parameters given in Table 7.1. This shows that REDOG's decryption process fails almost always.

Security parameter	p_{fail}
128	0.999996184401789
192	0.999999940394453
256	0.999999999068677

Table 7.2: Value of decryption failure probability p_{fail} per suggested parameters.

7.4 Message recovery attack on REDOG's implementation

Theorem 7.3.6 and the numerical examples show that, with probability almost 1, REDOG will fail decrypting. However, it is not exactly 1 and there exist some choices of $\mathbf{e}$ for which decryption still succeeds. One extreme way to avoid decryption failures, chosen in the reference implementation of REDOG, is to build errors as follows:

Algorithm 1. *(REDOG's error generator)*

1. *Pick* $\beta_1, \ldots, \beta_t \in \mathbb{F}_{q^m}$ *being* $\mathbb{F}_q$*-linearly independent.*
2. *Pick random permutation* π *on* $2n-k$ *symbols.*
3. *Set* $\mathbf{e}_{\mathsf{init}} = (\beta_1, \ldots, \beta_t, 0, \ldots, 0) \in \mathbb{F}_{q^m}^{2n-k}$. *Output* $\mathbf{e} = \pi(\mathbf{e}_{\mathsf{init}})$.

Error vectors in REDOG's reference implementation[1], whose performance is analyzed in [KHL+22b], are generated in an equivalent way to Algorithm 1. Indeed, $\mathbf{e}'$ has rank weight

$$\mathsf{wt}_R(\mathbf{e}') = (\mathbf{e}_1 P^{-1}, -\mathbf{e}_2 S^{-1}) \leq \lambda t$$

and can therefore be decoded using Φ.

Remark 7.4.1. *Algorithm 1 produces an error vector* $\mathbf{e}$ *such that* $\mathsf{wt}_H(\mathbf{e}) = \mathsf{wt}_R(\mathbf{e}) = t$ *as only* t *coordinates of* $\mathbf{e}$ *are nonzero.*

We are ready to give the description of an efficient message recovery algorithm.

Algorithm 2. *(Message recovery attack)*
Input*: REDOG's public key* pk *and a REDOG's ciphertext* $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) =$ *Encrypt*$(\mathbf{m}, \mathsf{pk})$ *generated by the reference implementation.*
Output*:* $\mathbf{m}$

1. *Let* C' *be the linear* $[2n-k, \ell]$*-code in the Hamming metric generated by* $G = (\mathsf{pk}_1 \mid \mathsf{pk}_2)$.
2. *Put* $f = 0$.
3. *While* $f = 0$*:*

[1] https://www.kpqc.or.kr/images/zip/REDOG.zip

(a) *Randomly select ℓ columns of G to form the matrix A. Let $\mathbf{c}_A$ be the matching positions in $\mathbf{c}$.*

(b) *If A is invertible*

i. *Compute $B = A^{-1}$ and $\bar{\mathbf{m}} = \mathbf{c}_A B$.*

ii. *Compute $\bar{\mathbf{c}}_1 = \bar{\mathbf{m}}\mathsf{pk}_1$.*

iii. *If $\mathsf{wt}_H(\mathbf{c}_1 - \bar{\mathbf{c}}_1) = t_1 \leq t$*

A. *Compute $\bar{\mathbf{c}}_2 = \bar{\mathbf{m}}\mathsf{pk}_2$.*

B. *If $\mathsf{wt}_H(\mathbf{c}_2 - \bar{\mathbf{c}}_2) = t - t_1$*
Put $\mathbf{m}' = \bar{\mathbf{m}}, \mathbf{e} = (\mathbf{c}_1, \mathbf{c}_2) - (\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2)$ and $f = 1$.

4. *Compute $\mathbf{m} = \mathbf{m}' - \mathsf{hash}(\mathbf{e})$.*

The inner loop is Prange's information-set decoding algorithm [Pra62] in the generator-matrix form with early aborts. If the chosen ℓ positions are not all error free then $\bar{\mathbf{m}}$ equals $\mathbf{m}$ with one or more rows of B added to it. Then $\bar{\mathbf{m}}\mathsf{pk}_1$ will be random vector and thus differ from $\mathbf{c}_1$ in more than t positions. If the initial check succeeds there is a high chance of the second condition succeeding as well leading to $\mathbf{e}$ with $\mathsf{wt}_H(\mathbf{e}) = t$.
We now analyze the success probability of each iteration of the inner loop of Algorithm 2. The field $\mathbb{F}_{q^m}$ is large, hence A very likely to be invertible. The algorithm succeeds if the ℓ positions forming A are chosen outside the positions where $\mathbf{e}$ has non-zero entries. This happens with probability

$$\frac{\binom{2n-k-t}{\ell}}{\binom{2n-k}{\ell}}.$$

Each trial costs the inversion of an $\ell \times \ell$ matrix and up to three matrix-vector products, where the vector has length ℓ and the matrices have ℓ, n, and $n-k$ columns respectively, in addition to minor costs of two vector differences and two weight computations.
We implemented the attack in Algorithm 2 in SageMath 9.5; see section 7.8 for the code. We perform faster early aborts, testing $\bar{\mathbf{m}}$ on only $t+3$ columns of pk_1. The probability that a coordinate matches between $\mathbf{c}_1$ and $\bar{\mathbf{c}}_1$ for $\bar{\mathbf{m}} \neq \mathbf{m}$ is q^{-m} and thus negligible for large m. Hence, most candidate vectors $\bar{\mathbf{m}}$ are discarded after $(t+3)\ell^2$ multiplications in $\mathbb{F}_{q^m}$. Running the attack on a Linux Mint virtual machine we broke the KAT ciphertexts included in the submission package for all the proposed parameters. We also generated a bunch of ciphertexts corresponding to randomly chosen public keys and messages and measured the average running time of our algorithm.

As can be seen from Table 7.3, the attack on outputs of the reference implementation succeeds in few steps and is very fast to execute for all parameter sets.

Security parameter	$\log_2(\mathsf{Prob})$	Time_{KAT} (sec.)	$\text{Time}_{100}(sec.)$
128	-5.62325179726894	~ 8.01	~ 9.17
192	-7.51182199577027	~ 108.13	~ 112
256	-9.40052710879827	~ 167.91	~ 133.43

Table 7.3: Prob is the probability of success of one iteration of the inner loop of Algorithm 2. Time_{KAT} is the average timing of message recovery attack over entries in the KAT file (30 for 128 bits, 15 for 192 bits, 13 for 256 bits). Time_{100} is the average timing of message recovery attack over 100 ciphertext generated by REDOG's encryption.

7.5 Recomputing attacks costs

In this section we deal with the computation of complexities of general attacks against cryptosystems relying on the rank decoding problem. We noticed that the official REDOG submission [KHL+22a], as well as [LTP21] do not consider attack algorithms proposed in [BBC+20a] and [BBB+23]
Our computations are reported in Tables 7.4, 7.5 and 7.6. The tables show that parameters suggested for REDOG provide significantly less security than expected. The tables also confirm that the parameters do provide the claimed security under attacks prior to [BBC+20a] when using a realistic exponent for matrix multiplication. Note that the computations in these tables ignore all constants and lower-order terms in the big-$\mathcal{O}$ complexities. This is in line with how the authors of the attack algorithms use their results to determine the security of other systems, but typically constants are positive and large. We apply the same to [BBB+23] although their magma code makes different choices.

7.5.1 Overview of rank decoding attacks

Recall that the public code is generated by the $\ell \times 2n - k$ matrix $(M \mid F)$ over $\mathbb{F}_{q^m}$. The error vector added to the ciphertext is chosen to have rank t. In the description of the attacks we will give formulas for the costs using the notation of this chapter, i.e., the dimension is ℓ and the error has rank t; we denote the length by N for reasons that will become clear later. The complexity of algorithms also depends on the matrix multiplication exponent ω.

The GRS [GRS16] algorithm is a combinatorial attack on the rank decoding problem. The idea behind this algorithm is to guess a vectorspace containing the space spanned by the error vector. In this way the received vector can be expressed in terms of the basis of the guessed space. The last step is to solve the linear system associated to the syndrome equations. This has complexity

$$\mathcal{O}\left((N-\ell)^{\omega} m^{\omega} q^{\min\{t\lfloor \ell m/N \rfloor, (t-1)\lfloor (\ell+1)m/N \rfloor\}}\right). \tag{7.7}$$

Note that we use ω here while the result originally was stated with exponent 3. These matrices are not expected to be particularly sparse but should be large enough for fast matrix multiplication algorithms to apply. The same applies to the next formulas.
The second attack, introduced in [GRS16], which we denote GRS-alg, is an algebraic attack. Under the condition that $\ell > \lceil ((t+1)(\ell+1) - N - 1)/t \rceil$ the decoding problem can be solved in

$$\mathcal{O}\left(t^{\omega} \ell^{\omega} q^{t(\lceil ((t+1)(\ell+1)-N-1)/t \rceil)}\right). \tag{7.8}$$

The attack AGHT [AGHT18] is an improvement over the GRS combinatorial attack. The underlying idea is to guess the space containing the error in a specific way that provides higher chance of guessing a suitable space. It has complexity

$$\mathcal{O}\left((N-\ell)^{\omega} m^{\omega} q^{t(\ell+1)m/N - m}\right). \tag{7.9}$$

The BBB+ attack [BBB+20] translates the rank metric decoding problem into a system of multivariate equations and then uses Gröbner-basis methods to find solutions. Much of the analysis is spent on determining the degree of regularity, depending on the length, dimension, and rank of the code and error. If $m\binom{N-\ell-1}{t} + 1 \geq \binom{N}{t}$ then the problem can be solved in

$$\mathcal{O}\left(\left(\frac{((m+N)t)^t}{t!}\right)^{\omega}\right). \tag{7.10}$$

If the condition is not satisfied then the complexity of solving the decoding problem becomes

$$\mathcal{O}\left(\left(\frac{((m+N)t)^{t+1}}{(t+1)!}\right)^{\omega}\right) \tag{7.11}$$

or the same for $t+2$ in place of $t+1$. The authors of [BBB+20] use (7.11) in their calculations and thus we include that as well.
The BBC+-Overdetermined,BBC+-Hybrid and BBC+-SupportMinors improvements that will follow are all introduced in [BBC+20a]. They make explicit the use of extended linearization as a technique to compute Gröbner

bases. For solving the rank-decoding problem it is not necessary to determine the full Gröbner basis but to find a solution to this system of equations. Extended linearization introduces new variables to turn a multivariate quadratic system into a linear system. The algorithms and complexity estimates differ in how large the resulting systems are and whether they are overdetermined or not, dependent on the system parameters.
BBC+-Overdetermined applies to the overdetermined case, which matches $m\binom{N-\ell-1}{t} + 1 \geq \binom{N}{t}$, and permits to solve the system in

$$\mathcal{O}\left(m\binom{N-\ell-1}{t}\binom{N}{t}^{\omega-1}\right). \tag{7.12}$$

These costs match matrix computations on a matrix with $m\binom{N-\ell-1}{t}$ rows and $\binom{N}{t}$ columns.
In case of an undetermined system, BBC+-Hybrid fixes some of the unknowns in a brute-force manner to produce to an overdetermined system in the remaining variables. The costs are testing all possible values for j positions, where j is the smallest non-negative integer such that $m\binom{N-\ell-1}{t} + 1 \geq \binom{N-j}{t}$, and for each performing the same matrix computations as in BBC on j columns less. This leads to a total complexity of

$$\mathcal{O}\left(q^{jt}m\binom{N-\ell-1}{t}\binom{N-j}{t}^{\omega-1}\right). \tag{7.13}$$

The brute-force part in BBC+-Hybrid quickly becomes the dominating factor. The BBC+-SupportMinors algorithm introduces terms of larger degrees first and then linearizes the system. This consists in multiplying the equations by some homogeneous monomials of degree b so as to obtain a system of homogeneous equations. However, for the special case of $q = 2$ the equations in the system might not be homogeneous. In this case, homogeneous equations coming from smaller values of b are considered. To state the conditions for this algorithm we first introduce some notation from [BBC+20a].

$$A_b := \sum_{j=1}^{b} \binom{N}{t}\binom{m\ell+1}{j}, \quad \text{and}$$
$$C_b := \sum_{j=1}^{b}\sum_{s=1}^{j}\left((-1)^{s+1}\binom{N}{t+s}\binom{m+s-1}{s}\binom{m\ell+1}{j-s}\right).$$

The degree of the equations formed in BBC+-SupportMinors depends on b, where $0 < b < 2 + t$ is minimal such that $A_b - 1 \leq C_b$ if such a b exists. In this case the problem can be solved with complexity

$$\mathcal{O}\left((m\ell + 1)(t + 1)A_b^2\right). \tag{7.14}$$

Remark 7.5.1. *We do not report the last two attacks presented in [BBC+20a] as the underlying approach has been pointed out to be incorrect in [BBB+23]. More precisely, [BBB+23] show that the independence assumptions made in [BBC+20a] are incorrect.*

SupportMinors in [BBC+20a] are not independent, [BBB+23] introduces a new approach that combines them while keeping independence, at least conjecturally and matched by experiments. They again multiply by monomials of degree up to $b - 1$ but a relevant difference is that the equations from the SupportMinors system are kept over $\mathbb{F}_{q^m}$. They introduce the following notation:

$$\begin{aligned}
\mathcal{N}_b^{\mathbb{F}_q} &= \mathcal{N}_b^{\mathbb{F}_{q^m}} - \mathcal{N}_{b,syz}^{\mathbb{F}_q}, \\
\mathcal{N}_b^{\mathbb{F}_{q^m}} &= \sum_{s=1}^{\ell} \binom{N-s}{t}\binom{\ell+b-1-s}{b-1} - \binom{N-\ell-1}{t}\binom{\ell-b-1}{b}, \\
\mathcal{N}_{b,syz}^{\mathbb{F}_q} &= (m-1)\sum_{s=1}^{b}(-1)^{(s+1)}\binom{\ell+b-s-1}{b-s}\binom{N-\ell-1}{t+s}, \text{ and} \\
\mathcal{M}_b^{\mathbb{F}_q} &= \binom{\ell+b-1}{b}\left(\binom{N}{t} - m\binom{N-\ell-1}{t}\right).
\end{aligned}$$

The problem can then be solved by linearization whenever $\mathcal{N}_b^{\mathbb{F}_q} \geq \mathcal{M}_b^{\mathbb{F}_q} - 1$. The complexity of solving the system is

$$T(m, N, \ell, t) = \mathcal{O}\left(m^2 \mathcal{N}_b^{\mathbb{F}_q}\left(\mathcal{M}_b^{\mathbb{F}_q}\right)^{\omega-1}\right).$$

Moreover, [BBB+23] introduce a hybrid strategy. Compared to BBC+-Hybrid it randomly picks matrices from $\mathsf{GL}_N(\mathbb{F}_q)$ to randomly compute $\mathbb{F}_q$-linear combinations of the entries of the error vector and applies the same transformation to the generator matrix, hoping to achieve that the last a positions of the error vector are all 0 and then shortening the code while also reducing the dimension.
This technique has complexity

$$\min_{a \geq 0}\left(q^{ta} \cdot T(m, N-a, \ell-a, t)\right). \tag{7.15}$$

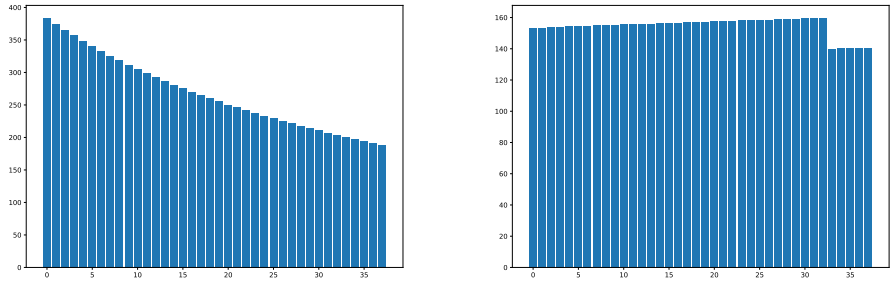


Figure 7.1: Plots showing the $\log_2$ of the costs for AGHT and BBB+ for the parameters at the 128–bit security level for different choices of code length.

7.5.2 Lowering the attack costs beyond the formulas stated

The combinatorial attacks GRS and AGHT perform best for longer codes, however, algebraic attacks that turn each column into a new variable perform best with fewer variables. For each attack strategy we search for the best number of columns that we should consider in order to obtain the cheapest cost of a successful break of REDOG. This is why we presented the above formulas using N rather than the full code length $2n - k$. The conditions given above determine the minimum length required relative to dimension and rank of the error.
We then evaluate the costs for each algorithm for each choice of length $N = \ell + t + i$, for every value of $i = 0, 1, \ldots, 2n - k - \ell - t$ satisfying the conditions of the attacks.
Figure 7.1 shows the different behavior of the algorithms for fixed ℓ and t and increasing i. The jump in the BBB+ plot is at the transition between the two formulas.
We point out that [BBC+20a] also considered decreasing the length of the code for the case of overdetermined systems, see [BBC+20a, Section 4.2] on puncturing the code in the case of "super"-overdetermined systems. We perform a systematic scan for all algorithms as an attacker will use the best possible attack.

7.5.3 The recomputed values

We computed complexity costs for all the attacks introduced in the previous subsection, taking into consideration two values of matrix multiplication exponent, namely $\omega = 2.807$ and $\omega = 2.37$. For each possible length $N + i$ for

$N = \ell+t$ and $i = 0, 1, \ldots, 2n-k-\ell-t$ we computed the costs for each attack strategy, keeping the lowest value per strategy. For the two cases of BBB+ and the three strategies described for the BBC+-* algorithms, we selected the best complexity among them. For the sake of completeness, we report the value of i in Tables 7.4–7.6 as well and the value of a for [BBB+23]. All the values are stated as the $\log_2$ of the costs resulting from the complexity formulas. The lowest costs of the best algorithm are stated in blue. Note the above-mentioned caveats regarding evaluating big-$\mathcal{O}$ estimates for concrete parameters.

Algorithm	Complexity formula	$\log_2$ of cost		i
		$\omega = 2.807$	$\omega = 2.37$	
GRS [GRS16]	7.7	228.03	-	36
GRS-alg [GRS16]	7.8	207.88	-	36
AGHT [AGHT18]	7.9	186.68	-	37
BBB+ [BBB+20]	7.10	140.06	118.25	33
BBC+ [BBC+20a]	7.12,7.13,7.14	77.83	65.73	33
Mixed-attack [BBB+23]	7.15	94.69	82.36	32

Table 7.4: Values of the $\log_2$ of attack costs for REDOG's suggested parameters for 128-bit security (see Table 7.1).

Algorithm	Complexity formula	$\log_2$ of cost		i
		$\omega = 2.807$	$\omega = 2.37$	
GRS [GRS16]	7.7	392.30	-	48
GRS-alg [GRS16]	7.8	368.18	-	48
AGHT [AGHT18]	7.9	337.69	-	49
BBB+ [BBB+20]	7.10 and 7.11	210.26	150	0
BBC+ [BBC+20a]	7.12,7.13,7.14	177.52	159.57	48
Mixed-attack [BBB+23]	7.15	181.21	164.03	49

Table 7.5: Values of the $\log_2$ of attack costs for REDOG's suggested parameters for 192-bit security (see Table 7.1).

Algorithm	Complexity formula	$\log_2$ of cost		i
		$\omega = 2.807$	$\omega = 2.37$	
GRS [GRS16]	7.7	604.07	-	60
GRS-alg [GRS16]	7.8	595.97	-	60
AGHT [AGHT18]	7.9	536.22	-	61
BBB+ [BBB+20]	7.10 and 7.11	269.03	227.15	0
BBC+ [BBC+20a]	7.12,7.13,7.14	337.92	318.01	61
Mixed-attack [BBB+23]	7.15	347.38	326.92	61

Table 7.6: Values of the $\log_2$ of attack costs for REDOG's suggested parameters for 256-bit security (see Table 7.1).

As shown in the table, suggested parameters of REDOG for 128 and 192 levels of security do not resist BBC+ attack and Mixed-attack for any choice of ω, and BBB+ for $\omega = 2.37$. Suggested parameters for level 256 resist all attacks except BBB+ for $\omega = 2.37$. In [LPR23] solutions to the decryption failure that also boost the security are proposed. of the suggested parameters of REDOG.

7.6 Revisiting results in [LTP21] on attacking REDOG

In this section we consider the key-recovery attack, as stated in [LTP21, Section 4] and show that it does not apply in the full generality as claimed but only works for $P \in \mathsf{GL}_n(\mathbb{F}_q)$ while [LTP21] states it for P being an isometry. This is another indication that the authors may not have understood the full generality of isometries, see also Remark 7.3.8.
The key-recovery attack is described for $S \in \mathsf{GL}_{n-k}(\mathbb{F}_q)$ and is the main reason for choosing S from a larger set. The attack computes an alternative parity-check matrix for the secret Gabidulin code by solving a system of equations. This matrix can be used in place of the secret key to decrypt any ciphertext The attack relies on the fact that, given M, the second part of the public key, i.e. $F = MP^{-1}H_1^T \left(H_2^T\right)^{-1} S$, can be written as a system of linear equations over $\mathbb{F}_q$ in terms of the coordinates of $H_1' = H_1(P^{-1})^T$ and $H_2' = H_2(S^{-1})^T$. If H_1' and H_2' are Moore matrices the number of unknowns is reduced so that the system becomes overdetermined and [LTP21] states that they are Moore matrices without giving a proof. The same claim is used in the plaintext-recovery attack in [LTP21, Section 5, Proposition 1] to show that the public code is a subcode of a Gabidulin code.

We show that $H_1(P^{-1})^T$ is a Moore matrix if $P \in \mathsf{GL}_n(\mathbb{F}_q)$ but not in the general case where P is an isometry. The first part also shows that $H_2(S^{-1})^T$ is a Moore matrix for $S \in \mathsf{GL}_n(\mathbb{F}_q)$
The following lemma shows that the product of a $k \times n$ Moore matrix A and an isometry P is an $\mathbb{F}_{q^m}$-multiple of a Moore matrix.

Lemma 7.6.1. *Let A be a $k \times n$ Moore matrix over $\mathbb{F}_{q^m}$ and let P be an isometry. Then $AP = \gamma B$ for $\gamma \in \mathbb{F}_{q^m}^*$ such that $P = \gamma Q$ for $Q \in \mathsf{GL}_n(\mathbb{F}_q)$ and B a $k \times n$ Moore matrix. This representation is unique up to $\mathbb{F}_q^*$ factors in γ and Q.*

Proof. Theorem 7.1.5 states that P is of the form $P = \gamma Q$ for some $\gamma \in \mathbb{F}_{q^m}^*$ and $Q \in \mathsf{GL}_n(\mathbb{F}_q)$. Any other factorization $P = \gamma' Q'$ satisfies $(\gamma/\gamma')I_n = Q'Q^{-1}$ which implies $\gamma/\gamma' \in \mathbb{F}_q^*$.
For this choice of γ write $AP = \gamma AQ$. It is then enough to prove that AQ is a Moore matrix. Write $Q = (Q_{i,j})$ for $i, j = 1, \ldots, n$. Let A_ℓ be the ℓ-th row of A and write $A_1 = (a_1, \ldots, a_n)$. Then $A_\ell = \left(a_1^{q^{\ell-1}}, \ldots, a_n^{q^{\ell-1}}\right)$ for $\ell = 1, \ldots, k$. We can write the entry $(AQ)_{i,j}$ as

$$(AQ)_{i,j} = \sum_{h=1}^{n} a_h^{q^{i-1}} Q_{h,j} = \sum_{h=1}^{n} (a_h Q_{h,j})^{q^{i-1}} = \left(\sum_{h=1}^{n} a_h Q_{h,j}\right)^{q^{i-1}} = (AQ)_{1,j}^{q^{i-1}}$$

for $i = 1, \ldots, k$ and $j = 1, \ldots, n$ using that $Q_{h,j} \in \mathbb{F}_q$ and that q-th powers are linear over $\mathbb{F}_{q^m}$. This proves that AQ is a Moore matrix. Setting $B = AQ$ we obtain the result in the statement. □

For $\gamma = 1$ we get the following corollary.

Corollary 7.6.2. *If $P = \mathsf{GL}_n(\mathbb{F}_q)$ and A is a $k \times n$ Moore matrix then AP is a Moore matrix.*

The attack proposed in [LTP21, Section 4.1] claims that given a $k \times n$ Moore matrix A over $\mathbb{F}_{q^m}$ and an isometry P, there exists a Moore matrix A' such that $A' = AP$. However, this statement is false except for the special case in Corollary 7.6.2.

Lemma 7.6.3. *Under the conditions of Lemma 7.6.1, AP is not a Moore matrix unless $P \in \mathsf{GL}_n(\mathbb{F}_q)$, or $A = 0$, or $k = 1$.*

Proof. By Lemma 7.6.1 $AP = \gamma B$ for $P = \gamma Q$ with $Q \in \mathsf{GL}_n(\mathbb{F}_q), \gamma \in \mathbb{F}_{q^m}^*$, and B a $k \times n$ Moore matrix. If $A = 0$ then $B = 0$ and the result holds trivially. Similarly, if $k = 1$ then AP has only a single row so that there is no constraint.

As P is invertible, $A \neq 0$ implies $B \neq 0$. Let B_ℓ denote the ℓ-th row of B. As B is a Moore matrix, we have for $B_1 = (b_1, \ldots, b_n)$ that $B_\ell = \left(b_1^{q^{\ell-1}}, \ldots, b_n^{q^{\ell-1}}\right)$ for $\ell = 1, 2, \ldots, k$. Because $B \neq 0$ there is at least one b_i that is nonzero.
The matrix γB for $\gamma \in \mathbb{F}_{q^m}^*$ has rows $\gamma B_1 = (\gamma b_1, \ldots, \gamma b_n)$ and $\gamma B_\ell = \left(\gamma b_1^{q^{\ell-1}}, \ldots, \gamma b_n^{q^{\ell-1}}\right)$ for $\ell = 1, 2, \ldots, k$. Hence, γB is a Moore matrix if and only if $\gamma b_i^{q^{\ell-1}} = (\gamma b_i)^{q^{\ell-1}}$ for $i = 1, 2, \ldots, n$ and $\ell = 1, 2, \ldots, k$. For $k > 1$ and $b_i \neq 0$ this holds if and only if $\gamma = \gamma^q$, which is equivalent to $\gamma \in \mathbb{F}_q$. Hence, $\gamma \in \mathbb{F}_q^*$ and thus $P = \gamma Q \in \mathsf{GL}_n(\mathbb{F}_q)$. □

The key recovery attack in [LTP21] builds a system of linear equations from the public key $(M \mid F)$ with $F = MP^{-1}H_1^T \left(H_2^T\right)^{-1} S$ rewriting it as

$$F\left(H_2(S^{-1})^T\right)^T = M\left(H_1\left(P^{-1}\right)^T\right)^T, \tag{7.16}$$

where the entries of $(H_1', H_2') = \left(H_1 \left(P^{-1}\right)^T \mid H_2 \left(S^{-1}\right)^T\right)$ are considered as unknowns and M and F are known. At first sight this is a system of $\ell(n-k)$ equations in $(n-k)(2n-k)$ variables and ℓ is much less than $2n-k$, hence, the system is severely underdetermined. Considering the system over $\mathbb{F}_q$ instead of over $\mathbb{F}_{q^m}$ just multiplies the number of equations and the number of variables by m. This is where [LTP21] uses that H_1' and H_2' are Moore matrices. From the above considerations this holds for H_2' but not for H_1'. The next proposition shows that the $\mathbb{F}_q$-linear system of equations obtained by the public key of REDOG is underdetermined if $P \notin \mathsf{GL}_n(\mathbb{F}_q)$, even if $S \in \mathsf{GL}_{n-k}(\mathbb{F}_q)$.

Proposition 2. *Let M be a full-rank $\ell \times n$ matrix over $\mathbb{F}_{q^m}$, $(H_1 \mid H_2)$ a $(n-k) \times (2n-k)$ Moore matrix over $\mathbb{F}_{q^m}$, P a rank-metric isometry of the space $\mathbb{F}_{q^m}^n$, and $S \in \mathsf{GL}_{n-k}(\mathbb{F}_q)$. Consider the linear system of equations defined by $FS^{-1}H_2^T = MP^{-1}H_1^T$, where the entries of $\left(H_1 \left(P^{-1}\right)^T \mid H_2 \left(S^{-1}\right)^T\right)$ are considered as unknowns. Use an explicit basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ to turn this into a system of linear equations over $\mathbb{F}_q$. For $P \notin \mathsf{GL}_n(\mathbb{F}_q)$ this system has $m\ell(n-k)$ equations and $m^2n + m(n-k)$ variables.*

Proof. By Lemma 7.6.1, $H_1 \left(P^{-1}\right)^T$ is an $\mathbb{F}_{q^m}$-multiple of a $(n-k) \times n$ Moore matrix, say γB, with $\gamma \in \mathbb{F}_{q^m}^* \setminus \mathbb{F}_q^*$ unless $P \in \mathsf{GL}_n(\mathbb{F}_q)$. Moreover, $H_2(S^{-1})^T$ is an $(n-k) \times (n-k)$ Moore matrix by Corollary 7.6.2.
Consider the right-hand side of (7.16). Fix a normal basis $\{\alpha, \alpha^q, \ldots, \alpha^{q^{m-1}}\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Write $\gamma = \sum_{i=1}^m \gamma_i \alpha^{q^{i-1}}$ with $\gamma_i \in \mathbb{F}_q$. Let $(b_1, \ldots, b_n)$ be

the first row of B, and write its h-th coordinate as $b_h = \sum_{i=1}^{m} b_{h,i}\alpha^{q^{i-1}}$ with $b_{h,i} \in \mathbb{F}_q$. Then

$$\gamma b_h = \left(\sum_{i=1}^{m} \gamma_i \alpha^{q^{i-1}}\right)\left(\sum_{j=1}^{m} b_{h,j}\alpha^{q^{j-1}}\right) = \sum_{i,j=1}^{m} \gamma_i b_{h,j}\alpha^{q^{i-1}+q^{j-1}}.$$

The h-th entry of the ℓ-th row of γB becomes

$$\gamma\left(b_h\right)^{q^{\ell-1}} = \sum_{i,j=1}^{m} \gamma_i b_{h,j}\alpha^{q^{i-1}+q^{j+\ell-2}}.$$

By setting the variables as $x_{i,h,j} = \gamma_i b_{h,j}$, we obtain $m^2 n$ variables from the $M\left(H_1\left(P^{-1}\right)^T\right)^T$ part of the system. If $\gamma \in \mathbb{F}_q$ then $\gamma_1 = \gamma_2 = \cdots = \gamma_m$ and the m^2 term collapses to m. For a general isometry P no such assumptions can be made.
Consider now the left hand side of (7.16). By following the same steps as above, but now using that $H_2\left(S^{-1}\right)^T$ is a Moore matrix, there are $m(n-k)$ unknowns for the equations corresponding to the first row of $F\left(H_2(S^{-1})^T\right)^T$ part, and all the other equations will share the same unknowns. □

For the parameters used in REDOG, $m > 2n-k$ and $n > \ell$. By Proposition 2 there are $u = m(mn - (\ell-1)(n-k)) > mn$ more variables than equations. Solving such an underdetermined system of equations requires trying all $q^u > (q^m)^n$ possibilities which makes the attack complexity exponential.

Remark 7.6.4. *Note also that since REDOG considers $S \in \mathsf{GL}_{n-k}(\mathbb{F}_{q^m})$ (via $S^{-1} \in \mathsf{GL}_{n-k}(\Lambda)$), these attacks could not be automatically applied even if P were limited to $\mathsf{GL}_n(\mathbb{F}_q)$. This can be shown by slightly tweaking the proof of Proposition 2 to having the left side of (7.16) require more variables. Note that in general the λ-dimensional space Λ is not invariant under the q-power Frobenius map, leading to the full $m^2(n-k)$ variables instead of $m(n-k)$ in Proposition 2. However, if $\mathbb{F}_{q^m}$ has m divisible by λ then it is possible to choose $\Lambda \cong \mathbb{F}_{q^\lambda}$ which would lead to only $\lambda m(n-k)$ variables for that part. The 128- and 192-bit parameters for REDOG chose prime m but the 256-bit parameters have $m = 135 = 3^3 \cdot 5$ and $\lambda = 3$, permitting this choice. While it is unlikely that a random choice of Λ picks this case and P being a general isometry also avoids having too few variables, we recommend choosing prime values of m to completely rule out this concern.*

Remark 7.6.5. *Despite Proposition 2 showing that the mentioned attacks do not apply to the case where P is an isometry of $\mathbb{F}_{q^m}^n$ and $S^{-1} \in \mathsf{GL}_{n-k}(\mathbb{F}_q)$, we do not advise to choose this combination to instantiate REDOG.*

7.7 Further work on REDOG

Lange, Pellegrini and Ravagnani [LPR23, Section 7,8] provide two ways to make REDOG's decryption correct. The first is a minimal change to fix the system by changing the space from which some matrix P^{-1} is chosen in a way that differs from the choice in REDOG and avoids the issue mentioned above. However, this still requires choosing much larger parameters to deal with our third contribution. The second way makes a different change to REDOG which improves the resistance to attacks while also fixing the decryption issue. It is shown that, using this strategy, not only REDOG parameters are sufficient to reach any claimed security level, but they provide security abundantly beyond each level, allowing room for an eventual optimization.

7.8 SageMath code

The following Sage script can be used for breaking the reference implementation for the 128-bit-security parameters. The file reads from the KAT file `rsp_128.rsp`. See also https://gitlab.tue.nl/tlange/kpqc-public/-/tree/master/redog for the code.

```
from sage.doctest.util import Timer

(n, k, l, q, m, r, t) = (44, 8, 37, 2, 83, 18, 6)

Fqm = GF(q^m)
ran_len = 30

def Hash_function(error_vec, length_n, length_k, length_l , seed_num = 0) : # copied from REDOG's implementation
    import random
    random.seed(seed_num)
    index_list = range(2*length_n - length_k)
    index = random.sample(index_list, length_l)
    Field = error_vec.base_ring()
    Hash_error = zero_matrix(Field, 1, length_l)
    for i in range(length_l):
        Hash_error[0,i] = error_vec[0, index[i]]
    return matrix(Hash_error)

def pis(G,y):
  M = copy(G)
  k,n = M.dimensions()
  p = list(Permutations(n).random_element())
  indexes = p[:k]
  indexes.sort()
  colsG = [M.columns()[i-1] for i in indexes]
  colsy = [y.columns()[i-1] for i in indexes]
  pisG = matrix(Fqm, colsG)
  pisy = matrix(Fqm, colsy)
  return pisG.transpose(), pisy.transpose()
```

```
def Prange(pubKey, y, t):
  M = pubKey[0].augment(pubKey[1])
  kpr,npr = M.dimensions()
  yleft = y[0][:n]
  yright = y[0][n:]
  while True:
    M1,y1 = pis(M,y)
    while not M1.is_invertible():
      M1,y1 = pis(M,y)
    U = M1.inverse()
    msg =y1*U
    y1left = msg*pubKey[0].matrix_from_columns(list(range(t+3)))
    wtleft = len([i for i in range(t+3) if y1left[0][i]!=yleft[i]])
    if wtleft <= t:
      x = msg*M
      wt = len([i for i in range(2*n-k) if x[0][i]!=y[0][i]])
      if wt <= t:
        e = y -x
        return msg, e

def string_to_mat(s,nrows,ncols):
  sbin = [list(reversed(Integer(ch,base=16).digits(base=2,padto=4))) for ch in s]
  sbin = flatten(sbin)
  meta = []
  for i in range(0,len(sbin),m):
    meta.append(Fqm(sbin[i:i+m]))
  return matrix(Fqm,nrows,ncols,meta[:ncols*nrows])

def el_to_string(polynomial): #copied from original submission
    p_coeff = matrix(ZZ(polynomial.integer_representation() ).digits(base=q, padto=m))
    p_bin = p_coeff[0]
    return p_bin

with open('rsp_128.rsp', 'r') as f:
  pk, cipher, pk0, pk1 = '', '', '', ''
  start = False # used to read pk which is on 2 different lines
  time = 0
  timer = Timer()
  for line in f:
    line = line.split()
    if line[:2] == ['msg', '=']:
      msg = string_to_mat(line[2],1,37)
    if line[:2] == ['pk', '=']:
      start = True
      pk = line[2]
      tmp = pk
      pk0 = string_to_mat(pk,37,44)
    elif start and line[0] == ',':
      pk= line[1]
      pk1 = string_to_mat(pk,37,36)
      start = False
    else: start=False
    if line[:2] == ['c','=']:
      cipher = line[2]
      cipher = string_to_mat(cipher,1,80)
      end=True
      timer.start()
      m_prange,e= Prange([pk0,pk1], cipher, 6)
      timer.stop()
      time += timer.walltime
      print(m_prange - Hash_function(e,44,8,37)==msg)
  print(time/ran_len)
```

The KAT files for the 192- and 256-bit-security parameters use a different encoding of the public key. This file is for `rsp_192.rsp` and can be used for `rsp_256.rsp` as well.

```
from sage.doctest.util import Timer

level = 192
(n, k, l, q, m, r, t) = (58, 10, 49, 2, 109, 24, 8)
ran_len = 15
Fqm.<z> = GF(q^m)

def Hash_function(error_vec, length_n, length_k, length_l , seed_num = 0) : # copied from REDOG's implementation
    import random
    random.seed(seed_num)
    index_list = range(2*length_n - length_k)
    index = random.sample(index_list, length_l)
    Field = error_vec.base_ring()
    Hash_error = zero_matrix(Field, 1, length_l)
    for i in range(length_l):
        Hash_error[0,i] = error_vec[0, index[i]]
    return matrix(Hash_error)

def pis(G,y):
  M = copy(G)
  k,n = M.dimensions()
  p = list(Permutations(n).random_element())
  indexes = p[:k]
  indexes.sort()
  colsG = [M.columns()[i-1] for i in indexes]
  colsy = [y.columns()[i-1] for i in indexes]
  pisG = matrix(Fqm, colsG)
  pisy = matrix(Fqm, colsy)
  return pisG.transpose(), pisy.transpose()


def Prange(pubKey, y, t):
  M = pubKey[0].augment(pubKey[1])
  kpr,npr = M.dimensions()
  yleft = y[0][:n]
  yright = y[0][n:]
  while True:
    M1,y1 = pis(M,y)
    while not M1.is_invertible():
      M1,y1 = pis(M,y)
    U = M1.inverse()
    msg =y1*U
    y1left = msg*pubKey[0].matrix_from_columns(list(range(t+3)))
    wtleft = len([i for i in range(t+3) if y1left[0][i]!=yleft[i]])
    if wtleft <= t:
      x = msg*M
      wt = len([i for i in range(2*n-k) if x[0][i]!=y[0][i]])
      if wt <= t:
        e = y -x
        return msg, e

def string_to_mat(s,nrows,ncols):
  sbin = [list(reversed(Integer(ch,base=16).digits(base=2,padto=4))) for ch in s]
  sbin = flatten(sbin)
  meta = []
  for i in range(0,len(sbin),m):
    meta.append(Fqm(sbin[i:i+m]))
```

```
  return matrix(Fqm,nrows,ncols,meta[:ncols*nrows])

def el_to_string(polynomial): #copied from original submission
    p_coeff = matrix(ZZ(polynomial.integer_representation() ).digits(base=q, padto=m))
    p_bin = p_coeff[0]
    return p_bin

def string_to_Fqm_tuple(s):
  s = s.split()
  tup = []
  el = ''
  j=1
  while j<len(s):
    if s[j]!='+':
      el += s[j-1]
      tup.append(Fqm(el))
      el = ''
      j +=1
    else :
      el += s[j-1]+s[j]
      j+=2
  if j ==len(s):tup.append(Fqm(el+s[j-1]))
  if j ==len(s)+1 : tup.append(Fqm(el+s[j-1]))
  return tup

with open(f'rsp_{level}.rsp', 'r') as f:
  cipher, pk0, pk1 = None, [], []
  startM = False # used to read pk which is on different lines
  startF = False
  i = 1 # used to read the public key 0 <= i < l
  time = 0
  timer = Timer()
  for line in f:
    if startM and i<l:
      i = i+1
      pkl = line[1:-2]
      pk0.append(string_to_Fqm_tuple(pkl))

    if startF and i < l:
      i += 1
      pkl = line[1:-2]
      pk1.append(string_to_Fqm_tuple(pkl))

    if line[:5] == 'msg =':
      msg = string_to_mat(line[6:-1],1,l)
    if line[:4] == 'pk =':
      startM = True
      pkl = line[6:-2]
      pk0.append(string_to_Fqm_tuple(pkl))
    if startM and line[0] == ',':
      pkl = line[3:-2]
      pk1.append(string_to_Fqm_tuple(pkl))
      startM = False
      startF = True
      i = 1
    if startF and i==l:
      startF=False
      i=1
    if line[:3] == 'c =':
      pk0 = matrix(Fqm,l,n,pk0)
      pk1 = matrix(Fqm,l,n-k,pk1)
      cipher = line[4:-1]
```

```
    cipher = string_to_mat(cipher,1,2*n-k)
    timer.start()
    m_prange,e= Prange([pk0,pk1], cipher, t)
    timer.stop()
    time += timer.walltime
    print(m_prange - Hash_function(e,n,k,l)==msg)
    pk0,pk1=[],[]
    i=1
    startM=False
    startF=False
print(time/ran_len)
```

Chapter 8

SMAUG and TIGER

SMAUG [CCH+22] and TiGER [PJP+22] are lattice-based key encapsulation algorithms (KEMs). Because they share several design elements we treat them jointly in this chapter. For more information on relating SMAUG to Kyber [SAB+22] see the thesis [Mer23] by Jorge Correa-Merlino. A thesis on TiGER is still in preparation.
In the following we will provide a description and discussion of key features of TiGER and SMAUG. It is instructive to compare both with the winner of NIST's PQC competition, Kyber. Naturally, the improvements employed in SMAUG and TiGER can be based on a more mature field of knowledge, incorporating more recent research results as compared to Kyber and thus we expect to see improvements. We then consider the security analysis under generic lattice attacks as well as combinatorial attacks. Finally we consider distinctive performance features of both submissions.

8.1 System description

Conceptually and on a high level, SMAUG and TiGER both closely follow the well-known template for building lattice-based KEMs as initiated by Regev [Reg05]. See Section 2.2 for more background on the Regev template. Regev-like cryptosystems mimic the well-known ElGamal cryptosystem, where each ciphertext consists of an ephemeral public key and a part involving the message. Decryption of the message is possible with a shared secret which is derived from the long-term key of the receiver $(\mathsf{pk}, \mathsf{sk})$ and the fresh ephemeral key $(\mathsf{epk}, \mathsf{esk})$ of the sender (so either from $(\mathsf{pk}, \mathsf{esk})$ by the sender or $(\mathsf{epk}, \mathsf{sk})$ by the receiver). All Regev-like encryption systems have comparatively simple security proofs linking them to some LWE-like assumption. These underlying security assumptions mainly differ in i) the type

of underlying lattice structure they are defined in, ii) the distribution that the secret key is chosen from, and iii) the error distribution that is used to bias (destroy) otherwise linear dependencies between public and secret variables. Applying these assumptions in constructions is simple. They guarantee that the long-term public key and the ephemeral public key of the ciphertext are indistinguishable from random to passive attackers. For random public keys, the decrypted message is essentially also random value to the attacker. In the security proof, this is enough to show that the scheme guarantees the relatively weak notion of IND-CPA security (indistinguishability under chosen plaintext attacks). Thus, even though there is a wide variety of LWE-like assumptions, each Regev-like cryptosystem chooses its algebraic setting and security assumption while following the same template.
In terms of provable security, the concrete choices influence the concrete correctness error and the achieved security level. Some schemes employ further mechanisms like error correcting codes to improve these key characteristics while others point to them as a source of implementation difficulties and issues with side-channel attacks. Essentially, each Regev-like cryptosystem proposes a new, unique trade-off between the key characteristics and efficiency.

8.1.1 Concrete parameters

TiGER and SMAUG both updated their parameters, here we comment on the most recent versions for both and also mention choices for Kyber.
Parameters should make the scheme i) have low bandwidth and high throughput, ii) have high concrete security supported by a proof to some security assumption, and iii) have adequate DFP.
Kyber uses MLWE with base ring $R_q = \mathbf{Z}_q[x]/(X^d + 1)$. The values of q and d are fixed to $q = 3329$ and $d = 256$ for all security levels; note that 256 divides $q - 1$ which permits using the NTT in R_q. The smallest security level uses two blocks of size 256 and the number increases with the security level, so $n \in \{2, 3, 4\}$.
SMAUG also uses MLWE and combines it with MLWR with power-of-2 cyclotomics for the ring at $d = 256$. Compared to Kyber it uses one more block for the highest security level, so $n \in \{2, 3, 5\}$. Like Saber [DKR+20] it wooks with a power-of-2 modulus $q \in \{1024, 2048\}$ and uses rounding.
TiGER uses uses RLWE and RLWR with $R_q = \mathbf{Z}_q[x]/(X^d + 1)$ with power-of-2-cyclotomic rings for $d = 512$ or 1024. The value of q is fixed to $q = 256$ for all security levels, so also using a power-of-2 modulus.

8.2 Security analysis

We first analyze the security against generic lattice attacks and then comment on key-search attacks and other issues.

8.2.1 Generic lattice attacks

Table 8.1 for SMAUG and Table 8.2 for TiGER contains the results of the estimator for the BKZ lattice attacks from Albrecht, Curtis, Deo, Davidson, Player, Postlethwaite, Virdia and Wunderer found in [ACD+18]. For more information see Section 2.1. The input of the code is given in Table 8.3 for SMAUG and Table 8.4 for TiGER.

Parameter n	512		768		1280	
Type	primal	dual	primal	dual	primal	dual
Q-Core-Sieve	109	134	155	179	259	297
Q-Core-Sieve + O(1)	125	146	171	192	275	292
Q-Core-Sieve (min space)	122	145	174	195	283	311
Q-β -Sieve	118	141	164	189	269	288
Q-8d-Sieve + O(1)	138	156	185	201	289	311
Core-Sieve	120	145	171	193	279	307
Core-Sieve + O(1)	136	156	187	208	295	315
Core-Sieve (min space)	151	170	215	231	326	336
β-Sieve	129	150	180	199	289	302
8d-Sieve + O(1)	149	166	200	216	309	323
Q-Core-Enum + O(1)	133	159	203	232	315	382
Lotus	137	164	209	241	321	353
Core-Enum + O(1)	240	243	308	340	420	514
8d-Enum (quadratic fit) + O(1)	260	266	328	378	440	431

Table 8.1: Estimations for security level for SMAUG found with code by [ACD+18]

8.2.2 Key search – the May attack on sparse secrets

As announced early in the competition by Bernstein [Ber22] and acknowledged by both submission teams, the combinatorial attack by May [May21] applies to the sparse distributions of the secret polynomials chosen (or obtained from rounding) in TiGER and SMAUG. Neither team had taken it into account.

Parameter n	512		1024		1024	
Type	primal	dual	primal	dual	primal	dual
Q-Core-Sieve	132	167	212	261	271	321
Q-Core-Sieve + O(1)	148	177	228	284	287	331
Q-Core-Sieve (min space)	149	180	227	244	305	354
Q-β -Sieve	141	173	221	246	281	327
Q-8d-Sieve + O(1)	162	188	241	277	301	349
Core-Sieve	146	178	225	286	299	349
Core-Sieve + O(1)	162	190	241	265	315	358
Core-Sieve (min space)	184	210	254	262	373	418
β-Sieve	155	185	233	262	309	355
8d-Sieve + O(1)	175	201	254	270	329	367
Q-Core-Enum + O(1)	173	211	231	252	395	472
Lotus	183	218	229	260	416	496
Core-Enum + O(1)	311	315	296	313	575	658
8d-Enum (quadratic fit) + O(1)	333	336	315	326	604	616

Table 8.2: Estimations for security level for TiGER found with code by [ACD+18]

For TiGER, Lee, Lee, and Kim [LLK23] showed that the 192-bit parameters in version 2.0 do not resist this attack and TiGER updated their parameters. In particular, TiGER changed the number of non-zero entries in the secret key from $(160, 84, 198)$ to $(142, 132, 196)$. SMAUG could not successfully be attacked by this attack, however, the SMAUG team changed the parameters as well to allow for a larger security margin.

8.2.3 Decryption failures

After v2.1 of TiGER was announced, Lee [Lee23b] showed that the DFP was calculated incorrectly and that it was larger by a factor of 2^8. The TiGER team acknowledged the issue and provided a new version (v3.0) in which they changed the parameters for the highest security level.

Both schemes seem to have adequate DFP now.

For TiGER we analyzed how one could mount a reaction attack if the CCA transform was not used. The challenge was to combine the basic attack [HGS99, Flu16] with the error correction algorithms in TiGER and we analyzed optimal choices of how to modify pairs of ciphertext entries in order to minimize the number of queries. This approach follows a similar analysis [BBLP18] on the NIST candidate HILA5 [BBLP18, Saa17], but in

n	$256 \cdot 2$	$256 \cdot 3$	$256 \cdot 5$
Standard deviation	1.0652	1.0652	1.0652
q	1024	2048	2048
k	2	3	5
Secret distribution	$(-1, 1), 140$	$(-1, 1), 140$	$(-1, 1), 140$
m	672	1024	1472
Claimed security	120	180	247
Category	1	3	5
Ring	$x^{n/k} + 1$	$x^{n/k} + 1$	$x^{n/k} + 1$

Table 8.3: Input for SMAUG in the code by [ACD+18]

n	512	1024	1024
Standard deviation	0.86	0.86	0.86
q	256	256	256
Secret distribution	$(-1, 1), 160$	$(-1, 1), 84$	$(-1, 1), 198$
Claimed security	120	192	246
Category	1	3	5
Ring	$x^n + 1$	$x^n + 1$	$x^n + 1$

Table 8.4: Input for TiGER in the code by [ACD+18]

contrast to HILA5, TiGER is proposed with a CCA transform so this attack does not apply on the proposed scheme. We tried but did not succeed to use it to attack the TiGER version with the larger DFP, different techniques would be needed to find a first decryption failure and then follow [GJY19]. These results will be included in the upcoming thesis by Casper von Berg.

8.3 Distinctive efficiency considerations

The original trade-off TiGER proposed focused on minimizing sizes to fit ciphertexts into the maximal transmission unit (MTU) of the highly important practical protocol IPsec which is 1244 bytes. The idea is that IPsec has a relatively small MTU and requires that the first message of each party, in the so-called IKA_SA_INIT phase, must not be fragmented. The mechanism IPsec employs requires fragmented messages to be protected by symmetric authenticated encryption under the current session key. However, since at the point of the first message there is no symmetric key available, the first message cannot be protected in this way. This is a pragmatic approach to ensure that the IPsec infrastructure does not have to change too much and in particular, stay competitive in terms of efficiency. However, the advantage will only materialize if future IPsec standardization a) does not change the MTU and b) maintains that ciphertexts and public keys should not be fragmented. Changing the MTU is theoretically possible within the limits set for Ethernet frames (minus headers) or by changing to jumbo frames, however the latter would break compatibility with older network infrastructure. Allowing for fragmentation of ciphertexts and public keys has the downside that efficiency (during connection establishment) is reduced. However, IPsec has a design that does not focus on reducing round complexity during startup. The handshake phase requires at least four protocol moves. While the design has several advantages like important privacy properties [SSL20], other designs of key exchange protocols like [Kra05, JKRS21] require fewer moves, which is important in particular on high-latency connections. In protocols designed for fewer moves, ephemeral keys and values designed for achieving authentication will be sent together in a single move, which makes even TiGER's parameters too large anyways.
Furthermore, due to a change in parameters in the most recent version v3.0 the goal of fitting into the IPsec MTU could not be upheld. Instead the TiGER team now focuses on meeting the MTU for Ethernet (1500 bytes minus headers). Meeting the MTU for Ethernet has the advantage that packets need not be fragmented on lower network levels. Both SMAUG and TiGER achieve that ciphertexts (and therefore also the shorter public

keys) fit into 1500 bytes for all security levels. However, if TCP/IP is used as a network protocol on top of Ethernet and thus headers are included, the remaining space is sufficient for all levels of TiGER but not for the highest level of SMAUG. We note that modern hardware equipment typically supports larger packets, so-called jumbo frames. These have a size of about 9000 bytes and Kyber, SMAUG, and TiGER all trivially meet this MTU and post-quantum versions of Internet protocols could move to requiring jumbo frames.

As a positive design feature, due to the modulus $q = 256$ having 8 bits only, TiGER is very SIMD friendly and operations on coefficients can easily be computed using native CPU commands. This is visible in the higher speeds achieved for TiGER.

SMAUG choose a trade-off that focuses on overall good efficiency but particularly small secret key sizes. This has the advantage that secret keys can more cheaply be stored in physically protected memory locations (trusted memory). Many server applications use such memory locations in the form of hardware security modules (HSMs). Trusted memory tends to be more expensive but features additional security guarantees that are typically not reflected in common security definitions but may matter in practice. Importantly, keys stored in trusted memory locations offer better protection against physical attacks like power analysis or fault injections. Trusted memory locations usually make sure that the secret key cannot leave the safe memory location in an unprotected form. Instead, the interface to the trusted memory only allows to apply the secret key to decrypt (or generate signatures on) input values sent into the HSM. At the same time, while security is higher, efficiency is often lower than computations on general CPUs that access the secret key from RAM. However, this is compensated by the fact that the trusted location computes on the secret keys in parallel to what the general CPU computes.

The main technique to obtain small secret keys is by using a sparse distribution, where most entries of the secret key vector are zero, and only storing the non-zero entries, together with the index of their component. However, drawing sparse secret keys has the downside of increasing the attack surface, particularly allowing combinatorial attacks, see section 8.2.2.

We note that the most recent variant of TiGER also proposes to draw secret keys from a sparse distribution and store them efficiently.

Both reference implementations compiled and produced the expected KAT vectors.

8.4 Considerations on provable security

SMAUG and TiGER provide security proofs in their documentation. Kyber, the NIST winner designed in 2017, uses the FO transform with implicit rejection and ciphertext contribution relying on [HHK17].
SMAUG uses the FO transform with implicit rejection and no ciphertext contribution relying on [BHH+19]. While relying on the additional non-standard assumption that the deterministic version of the PKE is sparse pseudo-random in the QROM, the SMAUG team uses [HHK17] to provide a tight reduction in the QROM.
TiGER uses the FO transform with implicit rejection and ciphertext contribution relying on [JZC+17] that seems to offer better security losses in the QROM (reducing security bounds of $q\sqrt{q^2\delta + q\sqrt{\epsilon}}$ to $q\sqrt{\delta} + q\sqrt{\epsilon}$).

We remark that we cannot see obstacles for SMAUG to incorporate ciphertext contribution. This could improve the security bounds by relying on [JZC+17]. Likewise we cannot see immediate obstacles to why the deterministic version of TiGER's IND-CPA scheme is not sparse pseudo-random. It is unclear whether the application of ECC violates the sparse pseudo-randomness property as required by [JZC+17].

8.5 General assessment

Both schemes TiGER and SMAUG seem solid and closely follow the Regev-template that has been very successful in the past. Both schemes try to improve on Kyber, the winner of the NIST competition and in general achieve higher efficiency. They achieve this mainly by incorporating recent research advances like more better FO transforms and exploiting sparse secret keys.
Similar to Kyber, the security proofs are non-tight in the quantum setting and incur a quadratic loss (unless relying on non-standard assumptions). This loss is not compensated for by the practical parameter choices. When proving security in more realistic multi-user models the loss is even larger. Using prefix hashing [DHK+21] the schemes could theoretically improve efficiency and avoid the naive loss in the number of users that comes from a hybrid argument. However, it is unclear if this still holds in the more realistic multi-user models with corruptions. Moreover it is unclear if the improvements in [DHK+21] synergize with those used in [JZC+18]. When trying to reduce to classical lattice assumptions, the loss is even greater. All in all, there is no indication of why the concrete choices are helpful to support that their instantiations are theoretically sound.

In terms of security, an important difference is that TiGER uses ECC to decrease the DFP, a feature that SMAUG avoids; the SMAUG designers point to side-channel attacks that could be facilitated with ECC.
The two schemes provide unique trade-offs between a subset of their parameter sizes. Both give convincing reasons for their choices. However, it is unclear whether in practice the reasons will indeed be of much importance. Small secret keys can be useful since they can be protected better physically. However, it is unclear what the additional costs are and how much weight this use case should have overall. Likewise, meeting the MTU of Ethernet is an interesting property even if it distinguishes SMAUG and TiGER only on the highest security level (NIST level V) from Kyber.
In terms of efficiency, SMAUG and TiGER have comparable numbers but TiGER seems to outperform SMAUG in virtually all categories. SMAUG has larger ciphertext size compared to TiGER on NIST level I (672 vs 640), smaller ciphertext on level III (1024 vs 1280 bytes) and larger ciphertext size on level V (1472 vs 1408 bytes). The public key sizes of TiGER on levels I,III, and V (480, 928, 928 bytes) are always smaller than those of SMAUG (672, 1088, 1792). Similarly, TiGER seems to outperform SMAUG in terms of computational efficiency on all levels, where the gap gets larger with higher levels up to a factor of two for key generation at level V. As for the size of the secret keys, TiGER has sizes of (134, 178, 263 bytes) on the three security levels whereas SMAUG offers (176, 236, 218 bytes). We note that the parameters for TiGER have been adapted considerably over the span of the competition.
For comparison the numbers for Kyber are: secret key (1632, 2400, 3168 bytes), public key (800, 1184, 1568), and ciphertext size (768, 1088, 1568). Both SMAUG and TiGER outperform Kyber computationally up to a factor of roughly 3.
We note that the recent results suggest that NTT can be applied to unfriendly moduli q [CHK+20]. This could be applied to improve the computational complexity of SMAUG and TiGER further.

8.5.1 Open questions

It would be interesting to hear the authors opinions on the following questions.

- It is interesting to know how large the parameters need to be in SMAUG and TiGER for a theoretically-sound instantiation in a strong security model (multi-user, corruptions).

- Moreover, can prefix hashing of pk be used to improve tightness and efficiency in the schemes?
- Can TiGER be proven tightly secure under the assumption of disjoint simulatability? Is this assumption plausible when using ECC?
- What are the computational costs for ciphertext contribution in the FO transform? -– Following the prefix hashing technique it seems considerable speedups can be made if the hash function is performed on less and smaller values (pk prefixes only). TiGER uses ciphertext contribution to apply the FO transform in contrast to SMAUG and so hashes the ciphertext as well. Nevertheless, it is still more efficient. How can this be best explained?
- Can the SMAUG team use ciphertext contribution to apply the [JZC+18] result and obtain better tightness. What would the disadvantage be in terms of efficiency?

Part II

Digital Signature Algorithms

Chapter 9

AIMer

AIMer [KHS+22a] is a signature scheme designed using the MPC-IN-THE-HEAD [IKOS09] paradigm to design an identification scheme which is then transformed into a signature scheme using the Fiat-Shamir transform.
The core idea of the MPC-IN-THE-HEAD paradigm in the context of signatures is to create an identification scheme that proves knowledge of a preimage for a one-way function in zero-knowledge. For this, the prover simulates a multi-party computation (MPC) that evaluates the one-way function on input of the secret-shared preimage. For this, every simulated party obtains a secret-share of the preimage. Then the MPC computation is simulated, the prover commits to all the internal state of each party, and sends the commitments together with all the communication between the parties to the verifier. The verifier then challenges the prover to open the full state of all but one party, which allows the verifier to probabilistically check that the prover did not cheat in the MPC simulation. At the same time, the security of the MPC protocol guarantees that even colluding parties cannot learn anything about the input of another party except what can be derived from the computation outcome. Thereby, it is guaranteed that the verifier does not learn anything about one of the secret shares and thereby the preimage remains hidden.
This identification scheme is then made non-interactive using the Fiat-Shamir transform. For this, the verifier gets replaced by the application of a hash function that takes all previous communications sent by the prover as well as the message to be signed, and the output is used as the challenge.
A famous example that followed this design is the Picnic [CDG+17] signature scheme which made it into the third round of the NIST competition. This design also has shown quite popular among submissions to the NIST signature on-ramp. The security of the resulting scheme is (provably) based on the security of several hash functions, use as pseudorandom generators,

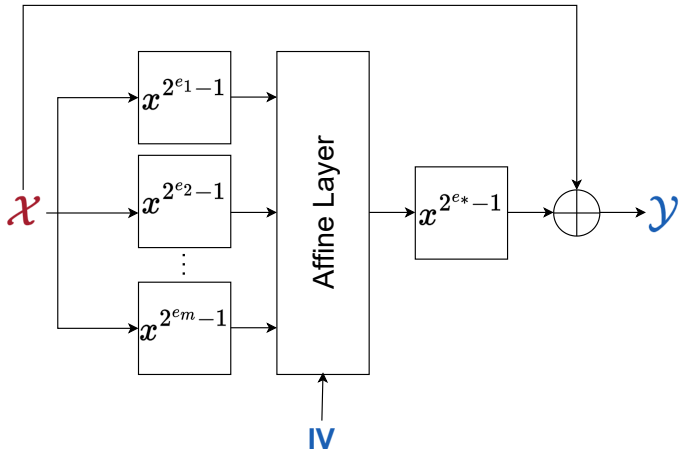


Figure 9.1: $\mathcal{X}, \mathcal{Y} \in \mathbb{F}_{2^n}$, S-box (non-linear) layer is Mersenne numbers power maps. affine layer is $n \times mn$ binary matrix and a binary vector randomized using IV

commitment scheme, and message hash, as well as the security of the one-way function for which knowledge of the preimage is proven. An important aspect is that while the hash functions used for the former applications can simply be instantiated with standardized hash functions / XOFs like SHA2, SHA3 or SHAKE, this does not hold for the one-way function. Given that the latter is evaluated in an MPC, its performance in this setting determines the performance of the signature scheme. Hence, proposals following the MPCitH design make use of special one-way functions that are optimized for the use in MPC[1]. For example, Picnic makes use of the third version of LowMC which was first proposed in [ARS+15] as its underlying one-way function. In the case of AIMer, the underlying one-way function is a new design called AIM [KHS+22b].
Below, we first discuss cryptanalysis results regarding AIM. Afterwards, we comment on the provable security claims made in the AIMer specification. Finally, we conclude.

9.1 Cryptanalysis results for AIM

The one-way function used in AIMer is AIM [KHS+22b] which is depicted in Figure 9.1.

[1]Specifically, these designs try to minimize the use of multiplication as it is the by far most costly operation in common MPC protocols.

Scheme	λ	Field	m	e_1	e_2	e_3	e_*
AIM-I	128	$\mathbb{F}_{2^{128}}$	2	3	27		5
AIM-III	192	$\mathbb{F}_{2^{192}}$	2	5	29		7
AIM-V	256	$\mathbb{F}_{2^{256}}$	3	3	53	7	5

Table 9.1: Parameters for different instances of AIM.

The parameters for different versions of AIM are described in Table 9.1. Let $\mathcal{X}, \mathcal{Y}$ be input and output of AIM respectively. Let $\mathcal{Z}_i$ be the output of the i^{th} S-box, $\mathcal{S}$ be the output of the affine layer, and $B_i(x) = \sum_{j=0}^{\lambda-1} a_{i,j} x^{2^j} + a_{i,\lambda}$ represent the affine layer applied to the state after the i^{th} Sbox. AIM can be described as:

$$\begin{aligned} \mathcal{Z}_i &= \mathcal{X}^{2^{e_i}-1} \text{ for } 1 \le i \le m, \\ \mathcal{S} &= \sum_{i=1}^{m} B_i(\mathcal{Z}_i), \\ \mathcal{Y} &= \mathcal{S}^{2^{e_*}-1} + \mathcal{X}. \end{aligned}$$

The security of AIM is analyzed against three categories of standard attacks, namely algebraic attacks, statistical attacks, and generic quantum attacks. In all the category of attacks, the security model of AIM restricts the attacker to the data complexity of $O(1)$, and the claim is that breaking any instance of AIM with λ bits of security requires $> 2^\lambda$ complexity.

9.1.1 Algebraic attacks

Algebraic attacks aim to exploit algebraic structures of the design to discover secret information. One approach in algebraic attacks is to model the design as a system of multivariate polynomials such that the set of solutions reveals secret information. The state of the art techniques for solving a system of multivariate polynomial includes:

- Fast exhaustive search that requires $d \cdot 2^n$ bit operations [BCC+10].
- Lokshtanov et al. probabilistic method [LPT+17] with asymptotic complexity of $2^{0.8765n}$.
- BooleanSolve algorithm [BFSS13] with complexity $\tilde{O}(2^{0.792n})$.
- Multiple Parity-Counting [Din21] with worst case complexity $O(2^{0.6943n})$.

- SAT solvers.
- Normal form methods.
- Hybrid approaches.

In the rest of this section, we study of the security claims of the designers of AIM against popular algebraic attacks.

Gröbner basis attacks

The idea behind the Gröbner basis attacks is to model the primitive as a system of multivariate polynomials with secrets as variables and solve the system to reveal the secret. The complexity of Gröbner basis attacks can be summarized as follow:

- Compute Gröbner basis in **degree reverse lexicographic order**:

$$O\left(\binom{\mathcal{V}+D}{D}^{\omega}\right),$$

  where $\mathcal{V}$ is the number of variables in the system and D is the solving degree of the system.

- Transform the basis into **lexicographic order**:

$$O\left(\sqrt{\mathcal{V}}D^{2+\frac{\mathcal{V}-1}{\mathcal{V}}}\right),$$

  where $\mathcal{V}$ is the number of variables and D is the degree of the corresponding ideal.

- Compute elimination ideals and expand partial solutions:

$$O\left(d\log(d)\log(q)\log(\log(d))\right),$$

  where d is the degree of the univariate polynomial in the Gröbner basis and q is the order of the field.

To analyze the security of AIM against Gröbner basis attacks, two possible approaches can be considered to model it as a system of polynomials. First approach is to model it as system of polynomials in $\mathbb{F}_{2^n}[x_1, \ldots, x_m]$ where the entire state is a variable and each operation defines a single polynomial. However, because of the design of the linear layer, which is an affine linearized polynomial in $\mathbb{F}_{2^n}$, the polynomial representation has large degree

and high density which makes the solving of the system computationally infeasible. Second approach is to consider each bit of the input as a variable in $\mathbb{F}_2[x_1, \ldots, x_m]$, and model each bit of the output as a polynomial in the input bits. In the case of AIM, the linear layer can be modeled as linear equations, and the S-box $y = x^{2^e-1}$ can be modeled as a system of quadratic equations by writing the Sbox relation as $xy = x^{2^e}$.

1. **basic:** **n** variables and **n** quadratic equations for each S-box.
2. **full:** **n** variables and **3n** quadratic equations for each S-box which results in an over-defined system.

In [KHS+22b, Appendix B.], different polynomial modelings are presented with their corresponding attack complexities. The results are summarized in Table 9.2. The complexity of computing the Gröbner basis is analyzed

scheme	n	#variables	#equations	degree of system	d_{reg}	complexity
AIM-I	128	$2n$	$6n$	4	22	214.9
AIM-III	192	$2n$	$6n$	6	31	310.6
AIM-V	256	$2n$	$6n$	8	40	406.2

Table 9.2: Complexity of finding the Gröbner basis.

using the Hilbert degree of regularity and the number of variables derived from different polynomial modelings.

XL

To Solve a system of multivariate polynomials using the XL family of algorithms, the system is extended to some degree D. Extending a quadratic system with n variables and m polynomials results in a system that has a Macualay matrix with $C = \sum_{i=1}^{D} \binom{n}{i}$ columns and $R = m\binom{n}{D-2}$ rows. As a result, techniques from linear algebra can be applied to solve the system if $R > C$. The complexity of XL attacks against AIM is summarized in Table 9.3

Scheme	n	#variables	#equations	degree of system	D	time
AIM-I	128	$3n$	$9n$	2	12	148
AIM-III	192	$3n$	$9n$	2	15	194.1
AIM-V	256	$3n$	$9n$	4	20	260.6

Table 9.3: Complexity of XL.

Multiple parity-counting

Multiple Parity-Counting approach is an effective approach to solve a system of polynomials over $\mathbb{F}_2$. The attack complexity against AIM is summarized in Table 9.4.

Scheme	n	#variables	#equations	degree of system	time
AIM-I	128	n	$3n$	10	137.3
AIM-III	192	n	$3n$	14	202.1
AIM-V	256	n	$3n$	12	264.1

Table 9.4: Complexity of [Din21].

9.1.2 Statistical attacks

The security model restricts the attacker to $O(1)$ data complexity.

Differential cryptanalysis

Having the data restriction of $O(1)$ makes the differential cryptanalysis irrelevant. However, the lower bounds on the probability of the best differential characteristics are summarized in Table 9.5.

n	128	192	256
$\log_2 \gamma$	-118.4	-178.0	-245.9

Table 9.5: Lower bounds of γ

Linear cryptanalysis

The security of AIM against linear cryptanalysis is ensured if:

$$\min_{1 \le i \le l} \left(2^{e_i} - 2\right)^2 \left(2^{e_*} - 2\right)^2 < 2^n$$

9.1.3 Quantum attacks

The authors analyzed the security of AIM against generic quantum attacks. The results of analysis against Grover's algorithm is summarized in Table 9.6.

Scheme	Offered Security Level
AIM-I	≥ 157
AIM-III	≥ 221
AIM-IV	≥ 285

Table 9.6: Lower bounds of γ

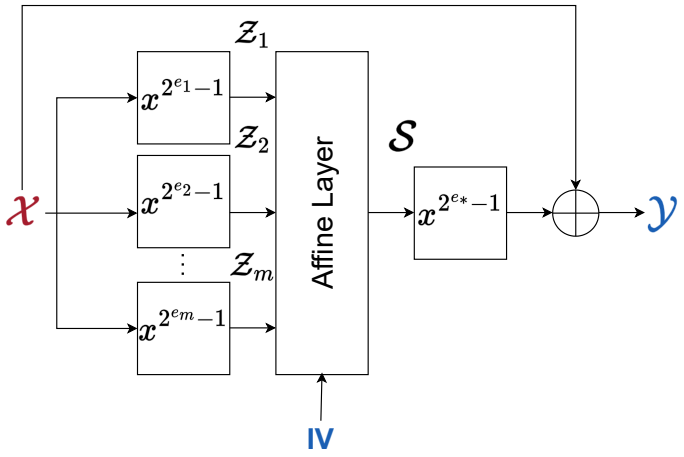


Figure 9.2: The variables used to model AIM are denoted with $\mathcal{X}$, $\mathcal{Z}_i$ for $1 \leq i \leq m$, $\mathcal{S}$.

9.1.4 Attacks on AIM

The design of AIM is published at ACM CCS 2023 and before that on ePrint [KHS+22b] and since then, it was the subject of cryptanalytic works. Two of the attacks that break the security claims of AIM are described in this chapter.

Fast exhaustive search

In [LMØM23], Fukang *et al.* used fast exhaustive search [BCC+10] to break AIM. The versions with 128/192/256-bit security are shown to be broken with complexity $2^{115}/2^{178}/2^{241}$. The attack exploits the low degree of the non-linear operations of AIM. Similar observation was made by Markku-Juhani O. Saarinen [2] which shows that AIM does not reach the claimed security level.
As shown in Figure 9.2, one can write:

$$\mathcal{X} = \mathcal{S}^{2^{e_*}-1} + \mathcal{Y}$$

[2] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BI2ilXblNy0

Scheme	λ	Field	$m+1$	Algebraic Degree	Time	Memory	Complexity
AIM-I	128	$\mathbb{F}_{2^{128}}$	3	10	$2^{136.2}$	$2^{61.7}$	2^{115}
AIM-III	192	$\mathbb{F}_{2^{192}}$	3	14	$2^{200.7}$	$2^{84.3}$	2^{178}
AIM-V	256	$\mathbb{F}_{2^{256}}$	4	15	$2^{265.0}$	$2^{95.1}$	2^{241}

Table 9.7: Complexity of breaking instances of AIM using fast exhaustive search.

And for each variable $\mathcal{Z}_i$, one can write:

$$\mathcal{Z}_i = \left(\mathcal{S}^{2^{e_*}-1} + \mathcal{Y}\right)^{2^{e_i}-1} \Rightarrow \mathcal{Z}_i = \sum_{j=0}^{2^{e_i}-1} \mathcal{Y}^j \mathcal{S}^{2^{e_*}-1(2^{e_j}-1-j)}$$

Then, $\mathcal{Z}_m$ can be written in two different ways as:

$$\mathcal{Z}_m = B_m^{-1}\left(c + \mathcal{S} + \sum_{i=0}^{m-1} B_i\left(\left(\mathcal{S}^{2^{e_*}-1} + \mathcal{Y}\right)^{2^{e_i}-1}\right)\right), \tag{9.1}$$

$$\mathcal{Z}_m = \left(\mathcal{S}^{2^{e_*}-1} + \mathcal{Y}\right)^{2^{e_m}-1}, \tag{9.2}$$

which gives an equation in $\mathcal{S}$, which can be solved using fast exhaustive search technique and the complexity is summarized in Table 9.7.

Linearization attack

It was shown in [ZWY+23] that the linearization method breaks AIM. The attack targets the design flaw in the first non-linear operation of AIM where all three Mersenne powers have the same input. Each S-box can be written as:

$$x^{2^{e_i}-1} = \left(x^d\right)^{s_i} \cdot x^{2^{t_i}},$$

and by guessing the value of x^d, one will have a linear system to solve. The summary of the attacks and their complexities are summarized in Table 9.8.

9.1.5 Mitigation of the proposed attacks

The designers of AIM, proposed AIM2 [KHSL23], which mitigates the attacks described earlier. In AIM2, the non-linear layers have larger exponents, and each input to a non-linear function is guaranteed to be different from other inputs in the same step. The design of AIM2 is depicted in Figure 9.3.

Scheme	λ	Field	$m+1$	d	t_1	t_2	degree of freedom	Complexity
AIM-I	128	$\mathbb{F}_{2^{128}}$	3	5	1	1	4	$2^{125.7}$
AIM-III	192	$\mathbb{F}_{2^{192}}$	3	45	8	8	12	$2^{186.5}$
AIM-V	256	$\mathbb{F}_{2^{256}}$	4	3	0	0	2	$2^{254.4}$

Table 9.8: Complexity of breaking instances of AIM using linearization method.

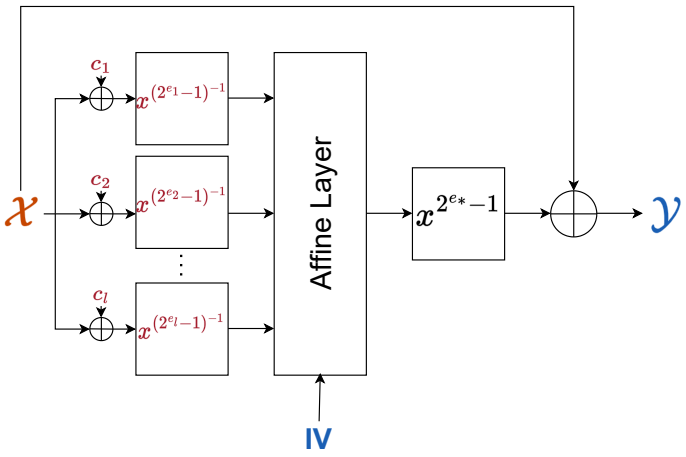


Figure 9.3: Design of AIM2

9.2 Provable security claims

The AIMer specification provides two provable security results. On the one hand, the authors present an analysis of the AIM design assuming that the S-boxes are random permutations. On the other hand, the authors present an analysis of the final signature scheme.

9.2.1 Security proof of AIM

The authors provide a security proof of one-wayness of the AIM construction assuming that the S-boxes are (public) random permutations. This is a common approach in arguing security of hash function designs. However, there is an important aspect which at least questions the meaningfulness of the presented result.
The security proof for AIM seems reasonable (although the argumentation is in parts too short to follow every detail). However, if we are willing to make the assumption that the S-boxes are public random permutations, then the construction is far more convoluted than necessary: Already the first layer is the "sum of permutations" construction which gives indifferentiability from a random oracle at least up to $2^{2n/3}$ queries (and previously conjectured to even reach full n-bit security, c.f. [GBJ+23], and thereby the same preimage bound. Indeed, for preimage finding, their argument for case one already demonstrates this, even with a better bound. This brings up two competing points. If the public permutation assumption is meaningful, it is unclear what benefit the additional S-box S3 and the feed-forward add. If they are necessary from a cryptanalysis perspective, this clearly demonstrates that the proof is meaningless as that invalidates the assumption that the S-boxes behave like public permutations.
As a minor note, we note that the used model is not the standard one and it is imprecise. If A is allowed to pick ct, it can simply pick ct = AIM(pt) for arbitrary pt. Hence, it is important to highlight that here the adversary is assumed to not have access to the permutations or AIM before committing to ct. However, this does not challenge the validity of the proof.

9.2.2 Security proof of signature

AIMer can be described as using AIM within a specific MPCitH based identification scheme called BN++ [KZ22] which generically describes how to turn the arithmetic circuit of a one-way function into an efficient identification scheme. The security proof of BN++ applies (and the proof in the AIMer

paper is a direct copy of the BN++ security proof with the one-way function being spelled out as AIM).

9.2.3 Interpretation of provable security results

While the formal security arguments seem sound, they have a significant shortcoming in common. Both proofs (security of AIM as well as security of the signature scheme) only consider classical adversaries. This also means that the bounds do not apply against quantum adversaries that at least can gain a polynomial advantage using Grover or the BHT-collision search algorithm on all the individual search bounds that make up the full bound. For the signature scheme, a result can likely be obtained using a recent work on a related scheme that used a different one-way function [AMHJ+23] but also follows the BN++ design on a high level. For the AIM bound, it will be significantly more complicated to extend the result to the setting of quantum adversaries as it is not yet known how to translate many arguments in the random permutation model to the quantum setting.

9.3 Overall assessment

There are several shortcomings with regard to the cryptanalytic security as well as the provable security that are described in this report. Some of them already led to attacks that break the underlying one-way function AIM. In general, the use of new dedicated symmetric primitives like LowMC [ARS+15] (used in Picnic), or Haraka[KLMR16] (used in one instance of SPHINCS+) has a troubled history (c.f., [LSW+22, Jea16] for the latest attacks). The security of both has been severely challenged, leading to refinement over refinement (LowMC is by now at version 3 and Haraka at version 2). The reason is that these designs are usually very aggressive to achieve performance, leaving only a minimal security margin. The same seemingly happened for AIMer. All the attacks are mitigated in the new design called AIM2. However, if the design of AIM2 is secure, it will be a revolutionary design in symmetric-key cryptography.
From a provable security point of view, it is not clear if the parameters actually achieve the necessary security against quantum adversaries or not. This needs further clarification.
As a minor remark, it would be good to also treat the instantiations of further symmetric primitives and their sizes as parameters of the scheme and mention them in the performance section.

Chapter 10

Enhanced pqsigRM: Code-Based Digital Signature Scheme with Short Signature and Fast Verification for Post-Quantum Cryptography

Enhanced pqsigRM is a code-based signature scheme submitted to the KpqC competition. In this report we describe the system in Enhanced pqsigRM support-doc. The scheme follows the structure introduced in [CFS01]. While the latter is a Goppa-code based signature scheme, pqsigRM and Enhanced pqsigRM are based on binary Reed-Muller codes or twists of those.
An earlier version of Enhanced pqsigRM has been submitted to the NIST PQC-competition under the name *pqsigRM*, and can be downloaded from pqsigRM NIST-submission. The scheme pqsigRM has been successfully attacked since its first submission to NIST PQC-competition and got modified to overcome the proposed attacks. The attacks and modifications to the original scheme can be found, in order of appearance, at pqsigRM NIST-comments. In this report we will describe the system and point out some inconsistencies between the specification and the supporting implementation. We then proceed to give some possible attack strategies.
Note that an updated version of Enhanced pqsigRM has been submitted to the NIST call for digital signature schemes. During the revision process two potential attacks (BBPS and DLV) have been pointed out. In this report we describe the DLV attack, which is at least supported by some SageMath code.

10.1 Enhanced pqsigRM

The main reason behind substituting Goppa codes with Reed-Muller codes [MS77, Chapter 13] in CFS is the efficient error correction provided by the closest coset decoding [Hem89], which is a modification of the soft decision decoding.

Remark 10.1.1. *All the codes involved are linear over* $\mathbb{F}_2$.

The supporting documentation describes a signature system that **does not** coincide with the C implementation provided in the KpqC submission.
We now describe the construction as in the specification. The signature system starts from the generator matrix Reed-Muller code. These codes have a $(U, U+V)$ structure [MS77, Chapter 2, Section 9], and can therefore be built in a recursive fashion.
Consider for example the Reed-Muller code $RM(6, 13)$, the visualization of the last two steps of the recursive construction of its generator matrix G is

$$G = \begin{pmatrix} RM(6,11) & RM(6,11) & RM(6,11) & RM(6,11) \\ 0 & RM(5,11) & 0 & RM(5,11) \\ 0 & 0 & RM(5,11) & RM(5,11) \\ 0 & 0 & 0 & RM(4,11) \end{pmatrix}. \tag{10.1}$$

The CFS signature system using plain RM codes is subject to structural attacks like Minder-Shokrollahi's attack [MS07], the Chizov-Borodin's attack [CB13] and the Square-code attack [OK15].
The authors propose therefore a set of modifications to G that allow the modified code to avoid these attacks. Here we list such theoretical modifications.

1. Code replacement

Unwrapping the recursive construction of the code $RM(r, m)$ one has 2^{m-r} copies of the code $RM(r, r) = \mathbb{F}_2^r$ which forces the dual code to have only codewords of even Hamming weight. Each of these copies of $RM(r, r)$ is replaced with a randomly chosen $[2^r, k_{rep}]$-code C_{rep} that allows efficient minimum-distance decoding and satisfies that its dual has at least one codeword of odd weight. Clearly, if $k_{rep} < 2^r$, the rank of the modified generator matrix drops.

2. Appending rows

In order to achieve the binomial distribution of the weights in the code a set of k_{app} random vectors of odd weight are appended to the matrix.

3. Appending a row of the dual to the hull

The hull of $RM(r, m)$ contains codewords of weight a multiple of four, while a random code's hull contains codewords of arbitrary even Hamming weight. In order to achieve the same in the considered code a random codeword c_{dual} from the dual code is appended to the generator matrix. If the the hull has still only codewords of weight a multiple of four, another codeword is chosen. **This check is not implemented**.

4. Partial permutation of the generator matrix

Submatrices of the matrix G in (10.1) are permuted in order to randomize the hull of the code and also prevent attacks like [MS07, CB13, OK15]. Two permutations σ, ρ of $1, \ldots, 2^{m-2}$ having support of size $|\mathsf{supp}(\sigma)| = |\mathsf{supp}(\rho)| < 2^{m-2}$ are chosen. Then the partial permutation of G happens as follows

$$G' = \begin{pmatrix} RM(6,11)^{\sigma} & RM(6,11)^{\sigma} & RM(6,11)^{\sigma} & RM(6,11)^{\sigma} \\ 0 & RM(5,11) & 0 & RM(5,11) \\ 0 & 0 & RM(5,11) & RM(5,11) \\ 0 & 0 & 0 & RM(4,11)^{\rho} \end{pmatrix}. \quad (10.2)$$

where $RM(6,11)^{\sigma}$ corresponds to the code $RM(6,11)$ whose columns are permuted according to σ, similarly for $RM(4,11)^{\rho}$.
The signature system described in the supporting documentation works as follows.

- key generation
  1. Apply modifications 1,2,3,4 to G to obtain G'.
  2. Compute parity check matrix H' and put in systematic form H_{sys}.
  3. Permute the columns of the non-identity part of H_{sys} by permutation Q obtaining T.
  4. Public key: T and correction capability w of the code generated by G'.
  5. Secret key: $Q, C_{rep}, \sigma, \rho, c_{dual}$ and k_{app} rows.
- sign
  1. Hash message M and random 32-bit integer i into $s = \mathsf{hash}(M|i)$
  2. Find error $e' = \mathrm{Decode}(s', H')$.

3. Set $e = Q^{-1}e'$.
4. Signature (M, e, i).

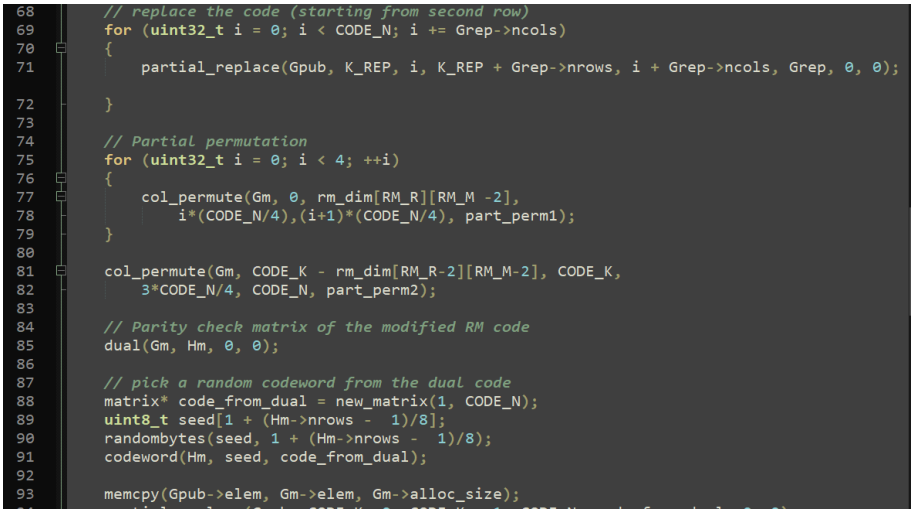
- open
  If $\mathsf{wt}(e) \leq w$ and $(IT)e^T = \mathsf{hash}(M, i)$ accept the signature as valid, else reject.

Decode$(\cdot)$ and $\mathsf{hash}(\cdot)$ are the decoding algorithm of the code and SHAKE-128/256, respectively.

Remark 10.1.2. *The documentation does not specify completely how the different modifications are involved in the decoding process. The recursiveness of an RM decoder is maintained and the decoder of C_{rep} is employed when the decoding indexes reach positions corresponding to codes in the first 2^r rows. Inverses of partial permutations σ and ρ are applied when this happens for the codes corresponding to C_{rep} or $RM(4, 11)$. Nevertheless, it is not clear how the appended rows are treated during decoding.*

10.2 Implementation

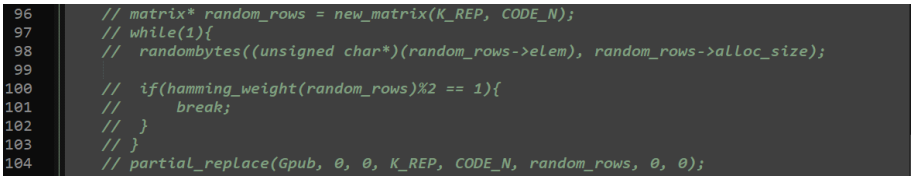
In this section we will show where things go wrong in the implementation. The following code snippet implements (at least it should) modifications 1,3, and 4.



```
68      // replace the code (starting from second row)
69      for (uint32_t i = 0; i < CODE_N; i += Grep->ncols)
70      {
71          partial_replace(Gpub, K_REP, i, K_REP + Grep->nrows, i + Grep->ncols, Grep, 0, 0);
72      }
73
74      // Partial permutation
75      for (uint32_t i = 0; i < 4; ++i)
76      {
77          col_permute(Gm, 0, rm_dim[RM_R][RM_M -2],
78              i*(CODE_N/4),(i+1)*(CODE_N/4), part_perm1);
79      }
80
81      col_permute(Gm, CODE_K - rm_dim[RM_R-2][RM_M-2], CODE_K,
82          3*CODE_N/4, CODE_N, part_perm2);
83
84      // Parity check matrix of the modified RM code
85      dual(Gm, Hm, 0, 0);
86
87      // pick a random codeword from the dual code
88      matrix* code_from_dual = new_matrix(1, CODE_N);
89      uint8_t seed[1 + (Hm->nrows -  1)/8];
90      randombytes(seed, 1 + (Hm->nrows -  1)/8);
91      codeword(Hm, seed, code_from_dual);
92
93      memcpy(Gpub->elem, Gm->elem, Gm->alloc_size);
```

Figure 10.1: Excerpt of keypair.c - Modification (1),(3) and (4)

The memcpy reverts completely the work done to replace the copies of $RM(6,6)$ with a random $[2^r, k_{rep}]$-code. The correct way to perform these operations would be to use `GM` instead of `Gpub` in line 71.



```
96      // matrix* random_rows = new_matrix(K_REP, CODE_N);
97      // while(1){
98      //  randombytes((unsigned char*)(random_rows->elem), random_rows->alloc_size);
99
100     //  if(hamming_weight(random_rows)%2 == 1){
101     //      break;
102     //  }
103     // }
104     // partial_replace(Gpub, 0, 0, K_REP, CODE_N, random_rows, 0, 0);
```

Figure 10.2: Another excerpt of keypair.c - Modification (2)

The modification by appending random odd-weight rows never happens as lines 96 – 104 are commented out.

Remark 10.2.1. *In the implementation provided in the KpqC submission of Enhanced pqsigRM only modifications* 3 *and* 4 *are correctly implemented. Indeed, the resulting code that is actually used in Enhanced-pqsigRM is generated by*

$$G'' = \begin{pmatrix} G' \\ v \end{pmatrix}^Q$$

where v is the appended random code word from the dual of the code generated by G'' and Q is the permutation matrix.

Furthermore, we noticed that Enhanced pqsigRM had KAT mismatches.

10.2.1 Indistinguishability issues

Note that the Reed-Muller code $RM(r, m)$ contains only even weight code words and σ_1 and σ_2 don't affect this. So G' and v have even weights only and thus G'' **has even weight distribution**.

Remark 10.2.2. *The weights of a random binary linear code are binomially distributed.*

The proportion of $[2^{13}, 2^{12}]$-codes over $\mathbb{F}_2$ having only even weights is given by the ratio of Gaussian coefficients

$$\begin{bmatrix} 2^{13}-1 \\ 2^{12} \end{bmatrix}_2 / \begin{bmatrix} 2^{13} \\ 2^{12} \end{bmatrix}_2 \sim 10^{-1234}.$$

Thus, betting on the code having even weight distribution will win the distinguishing game.
This invalidates the EUF-CMA proof given in an earlier version of the specification [LLKN19, Theorem 10].

10.3 Other analysis directions

Let us start with a lemma giving the shape of a parity check matrix of the code generated by G'.

Lemma 10.3.1. *A parity check matrix of the code spanned by G' is*

$$H' = \begin{pmatrix} RM(6,11)^{\sigma_2} & RM(6,11)^{\sigma_2} & RM(6,11)^{\sigma_2} & RM(6,11)^{\sigma_2} \\ RM(5,11) & 0 & RM(5,11) & 0 \\ RM(5,11) & RM(5,11) & 0 & 0 \\ RM(4,11)^{\sigma_1} & 0 & 0 & 0 \end{pmatrix}$$

Proof. Recall that $RM(r, m)^\perp = RM(m - r - 1, m)$, so

$$RM(6,11)^\perp = RM(4,11) \quad \text{and} \quad RM(5,11)^\perp = RM(5,11).$$

Also, $(RM(r, m)^\sigma)^\perp = \left(RM(r, m)^\perp\right)^\sigma$. Finally, $H'G'^T = \mathbf{0}$ and $\mathsf{rank}(H') = n - \mathsf{rank}(G')$. □

10.3.1 The strategy

The idea given in this paragraph could lay the foundations for a possible structural attack. This assumes one can spot the appended vector v. Let

H'' be a parity check matrix of the code $\mathsf{span}(G'')$ and define the hull of the code $\mathsf{span}(H'')$ as $\mathsf{hull}'' = \mathsf{span}(G'') \cap \mathsf{span}(H'')$. Similarly do for hull'.
Let C_0 be the code obtained from $RM(6,13)^Q$ by puncturing on $supp(\sigma_1)$ and $supp(\sigma_2)$ and inserting 0's in those positions.

Remark 10.3.2. *C_0 is invariant under permutations σ_1, σ_2. Also, $C_0 \subseteq \mathsf{hull}''$.*

Therefore hull'' contains as a subcode a punctured $RM(6,13)^Q$. There exist attacks [1] that can recover (at least partially) Q from such a code. Once Q is found, apply plain Minder-Shokrollahi [MS07] on the first 2^{m-2} columns of $G'^{Q^{-1}}$ to recover σ_1. Similarly do with the last 2^{m-2} cols of $H'^{Q^{-1}}$ to find σ_2. Moreover $\dim \mathsf{hull}' \cap RM(6,13)$ is almost as big as $\dim \mathsf{hull}'$. This gives hope to an attacker.

10.4 Unreal scenario - recovering Q, σ and ρ from H'

In the ideal case where the Gaussian elimination in step (2) of the key generation never happens and that we can revert modification 3, it is easy to check that that the following matrix

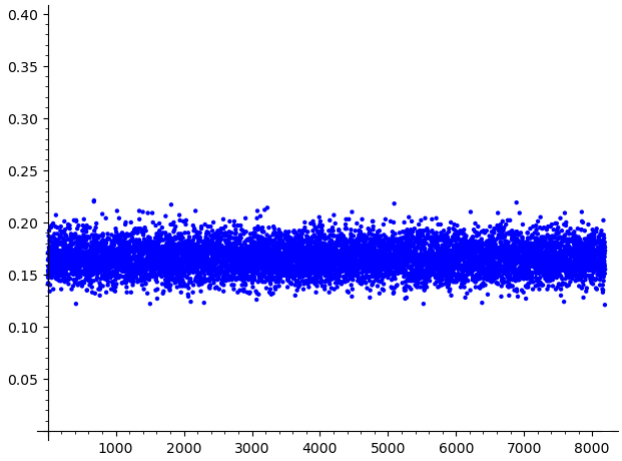
$$H'' = Q \begin{pmatrix} RM(6,11)^\rho & RM(6,11)^\rho & RM(6,11)^\rho & RM(6,11)^\rho \\ RM(5,11) & 0 & RM(5,11) & 0 \\ RM(5,11) & RM(5,11) & 0 & 0 \\ RM(4,11)^\sigma & 0 & 0 & 0 \end{pmatrix} \tag{10.3}$$

has full rank and annihilates G'^Q. Therefore it is a valid parity check matrix of G' so we can assume that $H' = H''$. Knowing the exact structure of G when $RM(r,m)$ is constructed recursively allows to devise an algorithm that recovers the permutations Q, σ and ρ at a computational cost in $O(2^{52})$. The SageMath code provided in Appendix 10.8 could be used for the computations in attack the parity check matrix H' for the proposed parameters $RM(6,13)$.

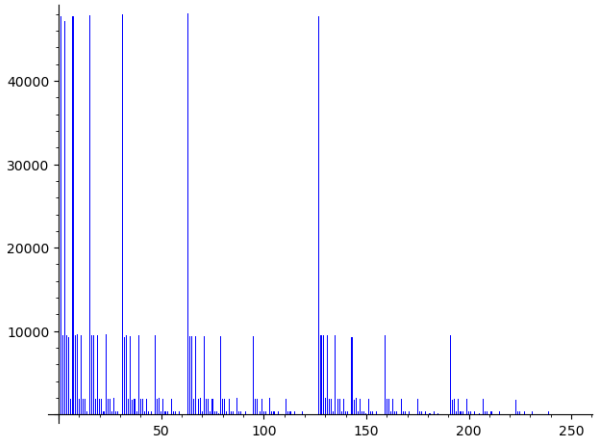
[1] https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/pqsigRM-official-comment.pdf

10.5 Some statistical analysis of the signatures

In this section we report the plots of the distributions of 1 bits over a set of 1000 signatures. The first plot shows the probability of each bit to be set. Except for slight outliers, where we can see that some bits have double



probabilities of being set than some other, this analysis didn't lead me to any conclusion about the signature or key structure.
We tried to plot the distribution of the bytes of a batch of hundreds of signatures. we represent the bytes as integers in $\{0, \ldots, 256\}$. The result is as in Figure 10.5.



After some discussions we agree that the behavior depicted in 10.5 is a natural consequence of the fact that, for example, different sequences of 8 bits with

just one set bit occur equally likely. With less probability a sequence of 8 bits will contain two 1 bits and so on. Moreover the number of different 8 bits sequences having two set bits are more than the sequences with one set bit ($\binom{8}{2} > 8$).

10.6 The Debris-Loisel-Vasseur attack

An updated version of Enhanced pqsigRM has been submitted to NIST's Round 1 call for signature schemes on 17-07-2023. On the 29-07-2023 an heuristic method to recover the $(U \mid U + V)$ structure of the code has been announced on the pqc-forum by Thomas Debris-Alazard, Pierre Loisel and Valentin Vasseur. In this section we will describe the attack, which, in short, we will call *DLV-attack*.

Remark 10.6.1. *Due to the absence of a write up of the attack, the description in this section has been extracted from the announcement text and the attack implementation. This means that it might not coincide with the description that the authors of the attack would give.*

10.6.1 Correlation analysis of signatures

As a starting point, the DLV-attack collects statistical data from a bunch of signatures of Enhanced pqsigRM generated under the same key. The correlation analysis consists in finding the highest similarity between the bits of the error part of the signatures. Let $S \subset \mathbb{F}_2^n$ be a set of signatures of Enhanced pqsigRM of cardinality $N := \#S$. Consider the map $\mathsf{count} : \{1, \dots, n\} \times \{1, \dots, n\} \to \mathbb{N}$ defined as

$$\mathsf{count}_S(i, j) = \#\{s \in S \mid s[i] = s[j]\}.$$

Define a *matched pair* as the pair of $(i, j)_S \in \{1, \dots, n\} \times \{1, \dots, n\}$ to be the couple of integers i, j where i and j is such that $\mathsf{count}_S(i, j) \geq \mathsf{count}_S(i, j')$ for any $j' \in \{1, \dots, n\}$. The authors of the attack noticed that matched pairs very likely correspond to couples of coordinates in the signatures that were placed $n/2 = 4096$ indices apart before permutation Q^{-1} is applied during the sign process.

Remark 10.6.2. *In other words if $(i, j)_S$ is a matched pair, then either $Q(j) = Q(i) + 4096$ or $Q(i) = Q(j) + 4096$ with high probability.*

Choosing S such that $N \sim 50000$ it is possible to maximize the probability of this event and experimentally determine all the matched pairs correctly.

10.6.2 Exploiting correlations

Recall that the secret matrix G', i.e. the generator matrix $RM(6,13)$ code after modifications 1,2,3, and 4 generates a code of the form:

$$C = \begin{pmatrix} C_1 & C_2 \\ U & U \\ 0 & V \end{pmatrix}, \tag{10.4}$$

where (C_1, C_2) is the code of dimension k_{app} spanned by the appended rows. The columns of G are finally permuted by Q, the dual matrix of GQ in systematic form, i.e. $H = (I_k, T)$ is computed and T is published. The DLV attack takes advantage of the precomputed correlations in order to reveal the $(U \mid U+V)$ structure of the secret code, neutralizing the presence of C_1 and C_2, whose aim is to randomize the parity of the public code.

Remark 10.6.3. *Let M be any $n_M \times m_M$ matrix, we denote $M[a:b, c:d]$ the submatrix of M consisting of coordinates in $\{c, \dots, d\}$ of the rows from $a, \dots, b$. More in general, if $A \subset \{1, \dots, n_M\}$ and $B \subset \{1, \dots, m_M\}$ then we denote $M[A, B]$ the submatrix of M consisting of rows indexed by A and columns indexed by B. In the case $A = \{a\}$ (resp. $B = \{B\}$) then we will simply write $M[a, B]$ (resp. $M[A, b]$).*

Let MP_S be the set of all the 4096 matched pairs extracted from S. Reconstruct the entire public matrix by prepending the identity chunk to T, i.e. $H = (I_{\dim T} \mid T)$. Construct two $k \times n/2$ matrices $\hat{H}_l$ and $\hat{H}_r$ by collecting in $\hat{H}_l$ all the columns of H indexed by i and in $\hat{H}_r$ all the columns of H indexed by j, where $(i,j)_S \in MP_S$. Finally construct $\hat{H} = \left(\hat{H}_l \mid \hat{H}_r\right)$. Notice that due to the 4096 offset of the column indexed by a matched pair, $\hat{H}$ generates a code of the form

$$\begin{pmatrix} C_1' & C_2' \\ \pi(U) & \pi(U) \\ \pi(V') & \pi(V'') \end{pmatrix}$$

where the union of the columns of V' and those of V'' gives V. The reduced row echelon form of the product $\bar{H} := \mathsf{rref}\left(\hat{H} \begin{pmatrix} I_{4096} & I_{4096} \\ I_{4096} & 0 \end{pmatrix}\right)$ generates a code of the form

$$\begin{pmatrix} C_1'' & C_2'' \\ \pi(V) & \pi(V') \\ 0 & \pi(U) \end{pmatrix}.$$

The presence of V' is due to the fact that matched pairs can be of the form described in Remark 10.6.2. And the goal is now to find a permutation that

solves the issue introduced by the remark and therefore place columns of $\hat{H}$ in the correct half of the matrix (i.e. a permutation that swaps columns between $\hat{H}_l$ and $\hat{H}_r$). The problem is modeled as $k' := k - \dim U$ number of equation systems corresponding to k' number of $(k - \dim U) \times (n/2 - \dim U)$ matrices $M_1, \ldots, M_{k'}$.
Let $(i, j)_S \in MP_S$, recall that the columns of of H indexed by i and j are placed in $\hat{H}$ with an offset of 4096 positions, i.e. $\hat{H}[1 : k, h] = H[1 : k, i]$ and $\hat{H}[1 : k, h + 4096] = H[1 : k, j]$. The swap of the columns of $\hat{H}$ corresponding to a matched pair is equivalent to swapping $\hat{H}[1 : k, h]$ and $\hat{H}[1 : k, h + 4096]$ for each $h = 1, \ldots, 4096$. This can be done by working on the matrix $\bar{H}$ where the DLV attack simulates this swap by temporarily adding $\bar{H}[1 : k, h]$ to $\bar{H}[1 : k, h + 4096]$ instead of actually swapping. Denote by $\bar{H}_h$ the matrix obtained from $\bar{H}$ by adding the h-th column to the $(h + 4096)$-th column, for $h = 1, \ldots, n$. Let B be the set of pivots of $\bar{H}[1 : k, n/2 : n]$. The idea is to consider the effect of each column swap on the submatrix $\bar{H}[1 : k', B]$. Compute the difference of submatrices $\left(\bar{H} - \mathsf{rref}(\bar{H}_h)\right)[1 : k', B]$ and set $M_r[h, 1 : (n/2 - \dim U)] = \left(\bar{H} - \mathsf{rref}(\bar{H}_h)\right)[r, B]$ for $r = 1, \ldots, k'$ and $h = 1, \ldots, 4096$.

Remark 10.6.4. *Note that all the pivot elements in B appear in the $\pi(U)$ submatrix of $\bar{H}$, the submatrix $\pi(V')$ has zero entries in the columns indexed by B. Moreover the sum of columns and the* rref *transformation do not affect columns of $\bar{H}$ indexed by B. Also $\#B = n/2 - \dim U$.*

Recall that the goal is to find a permutation of the columns that makes $\pi(V')$ disappear in $\bar{H}$. The equation system $xM_r = \bar{H}[r, B]$ has a solution a linear combination of swaps annihilates the right hand side. We also need to handle the presence of the first k_app rows in each M_r which correspond to the code C''. Indeed, if a solution has a nonzero among the first k_{app} coordinates then $\bar{H}[r, B]$ gets appended to M_r hoping that it is a basis element of C''. Otherwise, if $x_h \neq 0$ then $\bar{H}[1 : k, h]$ will be added to $\bar{H}[1 : k, h + 4096]$ and the swap is recorded. This process of solving equation systems is performed and the recorded swaps are adjusted until the rank of $\bar{H}[1 : k', B] = k_{app}$ meaning that we reduced the rank of $\pi(V')$ to 0. This means that we have a combination of swaps, i.e. a permutation Q' that put all the columns indexed by each matched pair in MP_S in the right half of $\bar{H}$ and thus also of $\hat{H}$. Thus $\hat{H}Q'$ has the shape

$$\begin{pmatrix} C'_1 & C'_2 \\ \pi(U) & \pi(U) \\ 0 & \pi(V) \end{pmatrix}$$

. Now, due to the fact that U and V are also permuted $(U' \mid U' + V)$-codes the aim is to apply the same heuristic method to $\pi(U)$to reorder columns on the

right halves in a recursive way until the original secret code is recovered. It is worth noticing that appended rows are not present up in the next iterations, improving on the complexity of the attack.

10.7 Overall assessment

We pointed out inconsistencies between the documentation and the implemented version. Moreover as per Remark 10.1.2, it is not clear as the decoder for the modified code should deal with the k_{app} rows and and c_{dual}. The implementation issues were fixed in the subsequent submission to NIST, but an attack (DLV attack) exploiting correlations in signature bits has been devised. The attack recursively recovers the secret key by reconstructing the $(U \mid U + V)$ structure of the code. We gave an interpretation of the DLV attack. Furthermore, another attack has been announced by the NIST team but the details are unclear.

10.8 SageMath code

The SageMath code that follows implements the content of section 5.

```
reset()

r=6
m=13

dims = [
[1,1,1,1,1,1,1,1,1,1,1,1,1],
[0,2,3,4,5,6,7,8,9,10,11,12,13],
[0,0,4,7,11,16,22,29,37,46,56,67,79],
[0,0,0,8,15,26,42,64,93,130,176,232,299],
[0,0,0,0,16,31,57,99,163,256,386,562,794],
[0,0,0,0,0,32,63,120,219,382,638,1024,1586],
[0,0,0,0,0,0,64,127,247,466,848,1486,2510]
]

Q = Permutations(2^m).random_element()

#G = zero_matrix(dims[r][m], 2^m)

def genRM(Gen,r1,m1, rowf, rowl, colf, coll):
```

```
if r1 == 0:
for i in range(2^m1):
Gen[rowf][colf+i] = 1
elif r1 == m1:
for i in range(2^m1):
Gen[rowf+i][colf+i] = 1
else:
colm = int((colf+coll)/2)
genRM(Gen,r1,m1-1, rowf, rowf + dims[r1][m1-1], colf, colm)
genRM(Gen,r1,m1-1, rowf, rowf + dims[r1][m1-1], colm, coll)
genRM(Gen,r1-1,m1-1, rowf + dims[r1][m1-1], rowl, colm, coll)

def find_permutation(G,r,m):
origtmp = [[0 for i in range(2^m)] for j in range(dims[r][m])]
genRM(origtmp,r,m,0,2^(m-1),0, 2^m)
orig = matrix(GF(2),origtmp)
p = []
permcols = G.columns()
origcols = orig.columns()
for col in permcols:
p.append(origcols.index(col)+1)
return Permutations(2^m)(p)

def find_Q(G,r,m):
# reorder the columns of G depending on the zeroes
G1,G2,G3,G4 = [],[],[],[]
ind1,ind2,ind3,ind4 = [],[],[],[]
i=1
cols = G.columns()
zero1 = zero_vector(4096-dims[r][m-2])
zero2 = zero_vector(4096-dims[r][m-2]-dims[r-1][m-2])
zero3 = zero_vector(dims[r-2][m-2])
for col in cols:
if col[dims[r][m-2]:]==zero1:
G1.append(col)
ind1.append(i)
elif col[dims[r][m-2]+dims[r-1][m-2]:]==zero2:
G2.append(col)
ind2.append(i)
elif col[-dims[r-2][m-2]:]==zero3:
G3.append(col)
```

```
ind3.append(i)
else:
G4.append(col)
ind4.append(i)
i += 1
print(len(G1))
print(len(G2))
print(len(G3))
print(len(G4))
# extract permuted RM(5,11) code from G4
G4 = matrix(G4).transpose()
rhoC = G4[dims[r][m-2]:dims[r][m-2]+dims[r-1][m-2],0:2^(m-2)]
rho4 = find_permutation(rhoC,r-1,m-2)
G4.permute_columns(rho4.inverse())
rho4 = rho4.inverse()
ind4 = [ind4[rho4(i+1)-1] for i in range(2^(m-2))]

G2 = matrix(G2).transpose()
rhoC = G2[dims[r][m-2]:dims[r][m-2]+dims[r-1][m-2],0:2^(m-2)]
rho2 = find_permutation(rhoC,r-1,m-2)
G2.permute_columns(rho2.inverse())
rho2 = rho2.inverse()
ind2 = [ind2[rho2(i+1)-1] for i in range(2^(m-2))]

G3 = matrix(G3).transpose()
rowf = dims[r][m-2] + dims[r-1][m-2]
rowl = rowf + dims[r-1][m-2]
rhoC = G3[rowf:rowl,0:2^(m-2)]
rho3 = find_permutation(rhoC,r-1,m-2)
G3.permute_columns(rho3.inverse())
rho3 = rho3.inverse()
ind3 = [ind3[rho3(i+1)-1] for i in range(2^(m-2))]

G1 = matrix(G1).transpose()
rhoC = G1[0:dims[r][m-2],0:2^(m-2)]
rho1 = find_permutation(rhoC,r,m-2)
G1.permute_columns(rho1.inverse())
rho1 = rho1.inverse()
ind1 = [ind1[rho1(i+1)-1] for i in range(2^(m-2))]

return Permutations(32)(ind1+ind2+ind3+ind4)
```

```
Gtmp = [[0 for i in range(2^m)] for j in range(4096)]
genRM(Gtmp,r,m,0,2^(m-1),0, 2^m)
G = matrix(GF(2),Gtmp)

G.permute_columns(Q)
```

Chapter 11

FIBS: Fast Isogeny Based Digital Signature

FIBS [KLY22] is an isogeny-based signature scheme. It is constructed by instantiating the NIST selected standard SPHINCS+ [HBD+22] using an isogeny-based hash function based on the Charles-Goren–Lauter (CGL) hash function [CLG09] instead of SHA2 or SHAKE. Given that SPHINCS+ is a framework that can take any secure cryptographic hash function to instantiate the different functions used by SPHINCS+ this creates a secure scheme if CGL fulfills the security properties required by the SPHINCS+ security proof.
The sizes of keys and signatures of SPHINCS+ only depend on the hash function output length which remains the same for SPHINCS+, independent of the used hash function. However, the performance of SPHINCS+ is determined by the performance of the used hash function. Given that signing requires in the order of a few million evaluations of the hash function, already a slightly worse performance of the hash function results in a massive performance penalty. At the same security level, FIBS takes a more than 60 000-fold penalty in speed. With a signing time of 2 837 seconds for FIBS compared to 46 milliseconds for SPHINCS+ this reaches a runtime that is simply not practical for any relevant use-case.
For a more extensive discussion of how SPHINCS+ and the CGL hash function work and a comparison between FIBS and SPHINCS+ see the bachelor thesis of Arend Verbeek [Ver23].

11.1 System description

We start with a very, very brief introduction on isogenies and then introduce the CGL hash function that the authors have used to replace SHA2.

Isogenies

First we introduce elliptic curves which are smooth projective curves of genus one over a field k with a specified point ∞. The points on a specific curve over a finite field $\mathbb{F}_p$ form a group. A curve is supersingular over $\mathbb{F}_p$, for $p > 3$ if the number of points on the curve is equal to $p + 1$. Over $\mathbb{F}_{p^2}$, a supersingular curve has $(p-1)^2, p^2+1$, or $(p+1)^2$ points. Most elliptic curves can be written in the short Weierstrass form $E : y^2 = x^3 + ax + b$, for some $a, b \in k$ for some field k.
Isogenies then are rational maps between these curves, which are group homomorphisms. Usually isogenies are denoted $\varphi : E \to E'$ with E, E' elliptic curves. We will be looking at separable isogenies, which means the degree of the isogeny is the size of its kernel (where the kernel is defined as the points that are mapped to ∞ on E'). We denote an isogeny with a kernel of size ℓ as an ℓ-isogeny. Isogenies are uniquely defined by their kernel. Given a kernel we can use the Vélu formulas to find the codomain of the isogeny and find the images of specific points. For each isogeny $\varphi : E_1 \to E_2$ there exists a dual isogeny $\hat{\varphi} : E_2 \to E_1$ which has the same degree ℓ and for which it holds that the composition $\hat{\varphi} \circ \varphi = [\ell]_{E_1}$ is the multiplication-by-ℓ map on E_1 and likewise $\varphi \circ \hat{\varphi} = [\ell]_{E_2}$. Isogenies from a curve to itself, say $\varphi : E \to E$, are called endomorphisms. All the endomorphisms of a certain curve form a ring, called the endomorphism ring.
Curves are isomorphic to each other over the algebraic closure of the field, i.e, there exists a 1-isogeny between them, if and only if they have the same j-invariant, which is defined as $j(E) = 1728\frac{4a^3}{4a^3+27b^2}$. Thankfully, any supersingular elliptic curve in the algebraic closure $\mathbb{F}_p$ is isomorphic to a curve in $\mathbb{F}_{p^2}$, so we do not have to look at larger extension fields. Using this j-invariant we can easily find isomorphism classes, and define them in $\mathbb{F}_{p^2}$. Another important result is that for a prime ℓ there exist exactly $\ell + 1$ isomorphism classes of curves that are ℓ-isogeneous to E. This can be used to create the "ℓ-isogeny graph", where each node is an isomorphism class of elliptic curves, and the edges connect the nodes that are ℓ-isogeneous to each other.
This ℓ-isogeny graph has some really nice properties since it is almost ℓ-regular and it is very connected. Dual isogenies also make sure it is an undirected graph. Figure 11.1 shows the 2-isogeny graph created over $\mathbb{F}_{419^2}$.

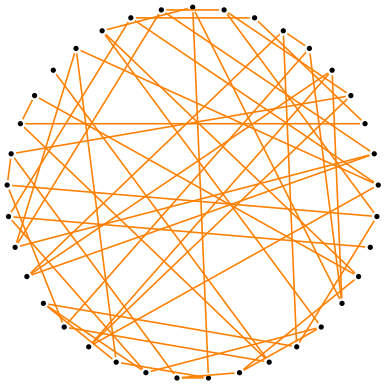


Figure 11.1: Nodes: Supersingular elliptic curves $\mathbb{F}_{419^2}$, edges are 2 isogenies. Image credit: Lorenz Panny.

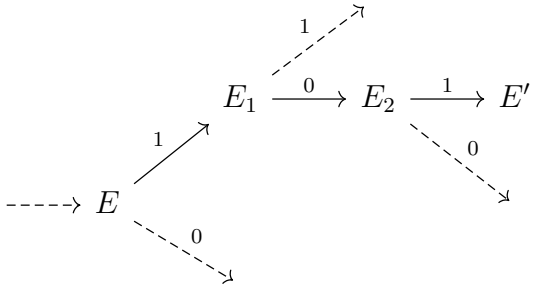


Figure 11.2: An example of a hashing of the message 101, the output is E'.

CGL hash function

The CGL hash function was introduced by Charles, Goren and Lauter [CLG09]. The original idea of the CGL hash function uses a 2-isogeny graph. This thus is an almost-3-regular graph. Given a starting point E on the graph, there are always 3 outgoing edges. But as soon as the first step is made, there are only 2 since we disregard the edge through which we entered the node. Simply denoting one of the paths as 0 and the other as 1, we can hash a message by walking on the graph. The output of the hash is then the node E' where we end up after taking a step for each of the bits of the message to be hashed. Note that for each step, a 2-isogeny needs to be computed, which is not very expensive, but still is not as fast as hashing by SHA2, for example. In Figure 11.2 we give an example of the hashing of the message 101.
An important thing to note here is that this hash function is only secure if

the starting curve E has an unknown endomorphism ring. The idea is that if we can find a collision in our hash function, then there are two distinct paths of length n between starting point E and output E'. These paths correspond to two isogenies of degree 2^n, and combining the dual of one with the other we create an endomorphism of degree 2^{2n}. So if the endomorphism ring is known, the Lauter-Petit attack [PL17] can easily find a cycle. Thus we need a trusted setup to create a curve E with unknown endomorphism ring, since finding an endomorphism is still considered a hard problem (though a new attack by Page and Wesolowski [PW23] has reduced the complexity from $O(p)$ to $O(p^{1/2})$).
Panny [Pan19] has introduced a change to the original version of the CGL hash function which leaves the prime ℓ to be variable and only uses a fraction r of the available edges at each step. This version fixes the problem sketched above and this version of the CGL hash function is what is used in FIBS. This ensures collision resistance.

11.2 Security and implementation considerations

The security of the CGL hash function is not too well understood and, as noted below, the specific properties that would be required for usage in a SPHINCS+ construction are not studied. However, the main concern with FIBS is efficiency. Table 11.1 shows the differences in runtime between FIBS and SPHINCS+.

	Key generation	Signature generation	Verification
FIBS-128	121 660 ms	2 837 040 ms	172,370 ms
SPHINCS+-128f (SHA2)	1.86 ms	46.20 ms	2.59 ms

Table 11.1: Comparisons of runtime between FIBS-128 and SPHINCS+-128f (SHA2)

Furthermore, we noticed that FIBS had KAT mismatches.

11.3 Provable security

The FIBS authors simply repeat the proof stated in the first-round submission of SPHINCS+ [HBD+17]. This proof was later found to be flawed and withdrawn in the round 2 documents [HBD+19].

A fixed proof appeared at Asiacrypt 2022 [HK22] and is referenced in the latest SPHINCS+ documentation. This proof has some additional requirements on the used hash functions (among others undetectability and decisional-second-preimage-resistance). Further research is needed to understand if the used isogeny-based hash function CGL guarantees these properties.
The version of the CGL function used in FIBS has not been proven to be either. There is a newer version of the CGL hash function by Doliskani, Pereira and Barreto [DPB17] which gives a better runtime and is proven to be preimage resistant. Even if FIBS adopts the use of this hash function, we expect it still can not compete with SPHINCS+ on speed.

11.4 Overall assessment

FIBS replaces the well established assumption that SHA2 or SHAKE256 behave like random functions with far less analyzed hardness assumptions regarding isogenies. By this, the authors create an isogeny-based signature scheme for which key generation, signing and verification can all be run within a day – something not necessarily the case for all isogeny-based signature schemes. However, it is questionable if there is any security benefit to this change given the amount of cryptanalysis spent on SHA2 and SHAKE compared to the amount spent on isogenies. In addition, the current submission still would have to fill the gap of arguing the exact security of the CGL hash function for undetectability and decisional-second preimage resistance. Even if all these aspects were handled with a positive outcome, it remains to note that the speed of FIBS is entirely impractical.

Chapter 12

GCKSign: Simple and Efficient Signatures from GCK

12.1 Introduction

GCKSign [WLP22] is a lattice-based signature scheme based on the Generalized Compact Knapsack problem, following Lyubashensky [Lyu09], with a Fiat-Shamir transformation leading to Schnorr-type signature scheme, in particular also based on Dilithium [LDK+22], a NIST PQCrypto selected algorithm. It is different from both in that it replaces the underlying hard problem of GCK one-wayness by a newly designed hardness assumption, Target Modified One-wayness (TMO). It is also simpler that Dilithium in that it does not rely on shortened public keys with hints. Its design rationale is mainly a performance improvement on Dilithium, both in key and signature sizes and in algorithm speed.

12.2 GCK vs. TMO one-wayness

Let $R_q = \mathbb{Z}_q[X]/(X^n+1)$, and $R_{[-\beta,\beta]}$ its subset of elements with coefficients in the interval $[-\beta, \beta]$. Choose a subset $S \subset R_{[-\beta,\beta]}$. A knapsack function for a randomly chosen $\mathbf{a} = (a_1, a_2, \ldots, a_m) \in R_q^m$ is a function $F_\mathbf{a} : S^m \to R_q$ given by $F_\mathbf{a}(\mathbf{x}) = \sum_i a_i x_i$, where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$.

GCK one-wayness is the problem of, given a pair $\mathbf{a} \in R_q^m, t \in R_q$, to find $\mathbf{x} \in S^m$ such that $F_\mathbf{a}(\mathbf{x}) = t$. Micciancio showed that this problem is as hard as SIVP. A relaxation of this hard problem is TMO one-wayness: given $\alpha, \beta \in \mathbb{R}$ and $\mathbf{a} \in R_q^m, t \in R_q$, it is hard to find $\mathbf{x} \in R_q^m$ and $c \in R_q$ such that $\|c\|_\infty \leq \alpha$ and $\|\mathbf{x}\|_\infty \leq \beta$ and $F_\mathbf{a}(\mathbf{x}) = ct$. It is trivial that GCK one-wayness

is at least as hard as TMO one-wayness for $\alpha \geq 1$.
The submission claims that the parameters α, β can be chosen such that the TMO one-wayness problem is at least as hard as the GCK one-wayness problem, while at the same time these parameters allow for smaller key and signature sizes. Part of the argument is also that a worst-case analysis is replaced by a different analysis leading to a more tight bound.

12.3 System description

For completeness we provide an overview of the key generation and signature generation and verification methods, to show that the scheme indeed resembles Schnorr signatures.
Parameters: η, B, h, L.
Key generation: sample $\mathbf{a} \in R_q^m$ from a seed, and $\mathbf{s} \in R_{[-\eta,\eta]}^m$, then the public key consists of $t = F_{\mathbf{a}}(\mathbf{s})$ and the seed, and the private key is $\mathbf{s}$ and the seed.
Signature generation: recover $\mathbf{a}$ from the seed, sample $\mathbf{y} \in R_{[-B,B]}^m$ and compute $v = F_{\mathbf{a}}(\mathbf{y})$, for the message m compute $\hat{c} = H(v, m) \in \{0,1\}^{256}$ where H is a hash function, compute $c = \text{encode}(\hat{c}) \in R_q$ for an encoding function leading to h coefficients ± 1 and all others 0, compute $\mathbf{z} = \mathbf{y} + c\mathbf{s}$ and reject if it is not in $R_{[-B+L,B-L]}^m$, in that case restart with fresh $\mathbf{y}$, then the signature is $(\mathbf{z}, \hat{c})$.
Signature verification: recover $\mathbf{a}$ from the seed, compute $c = \text{encode}(\hat{c})$, compute $w = F_{\mathbf{a}}(\mathbf{z}) - tc$, and verify if $\mathbf{z} \in R_{[-B+L,B-L]}^m$ and $\hat{c} = H(w, m)$. Verification makes use of the linearity of $F_{\mathbf{a}}(\mathbf{z}) = F_{\mathbf{a}}(\mathbf{y}) + cF_{\mathbf{a}}(\mathbf{s}) - tc = v$.

12.4 Problems

The first problem with GCKSign was noted by Kim, Ryu and Lee [KRL23] who point out a mistake in the cost of a key recovery attack. The mistake is that the (Module) SIS-problem to which the one-wayness can be reduced to is not a low-density problem but a high-density problem. The claimed hardness of GCKSign in the code SVP-model in the submission and as computed in [KRL23] are as follows:

NIST security level	II	III	IV
claimed in [WLP22]	125	183	268
claimed in [KRL23]	65.12	55.77	120.60

Thus it is clear that the submission does not deliver the security levels it promises. The submitters (see their reply in [KRL23]) acknowledged this

mistake, and promised to update their parameters. To our best knowledge they have not yet announced their update.
The second problem was noted by Kim [Kim23], who points out a mistake in the analysis of the hardness of the TMO problem. This is about the probability of c being invertible, which is claimed by the original submission to be 'overwhelming', while Kim asserts it is, in security level 2, only 2^{-46}, which is too low. The submitters acknowledge the issue and promise an update, indicating the computational efficiency problems this issue will lead to, but again they have not yet announced their update, to the best of our knowledge.

12.5 Overall assessment

TMO one-wayness is a new security assumption that has not received a lot of scrutiny from the cryptanalytic community yet.
The submission reasonably argues side channel resistance.
In the KpqC benchmark GCKSign scores among the best for key generation, and in the middle for signature generation and verification. Regarding key and signature sizes it promises a decent improvement compared to Dilithium. However, it remains to be seen what the effect of any forthcoming fixes and parameter updates will be.

Chapter 13

HAETAE Hyperball bimodAl modulE rejecTion signAture schemE

HAETAE [CCD+22] is a module lattice-based signature scheme based on the Fiat-Shamir with Aborts paradigm [Lyu09, Lyu12]. In that sense, it resembles NIST finalist Dilithium [LDK+22]. The main difference is that the design of HAETAE is aimed at improving the sizes of keys and signatures – the proposal claims 40% shorter signatures when comparing with Dilithium – and 20% smaller public keys. The main changes are using a bimodal distribution as in BLISS [DDLL13] and a new sampler using hyperballs, where Dilithium uses uniform distribution and BLISS used discrete Gaussians.
Below, we discuss the provable security claims made in the HAETAE specification. There are two versions of HAETAE, the October-2022 submission [CCD+22], retroactively labeled v0.9, and a May-2023 version, labeled v1.0.

13.1 System description

Please see Chapter 15 for the general description of Dilithium-like signatures. These systems use rejecting sampling on the signatures to ensure that their distribution is independent of the secret key.
Concretely, HAETAE works in $R = \mathbf{Z}[x]/(x^n + 1)$ for $n = 2^r$ (here $n = 256$), and also in $R_q = (\mathbf{Z}/q)[x]/(x^n+1)$ for q a prime with $2n|(q-1)$. All HAETAE parameters use $q = 64513$. Unlike NCC-Sign but like Dilithium, HAETAE uses module lattices in which the lattice elements are vectors of elements from R. The public key is an $k \times (\ell + k)$ matrix over R. The maximum-security

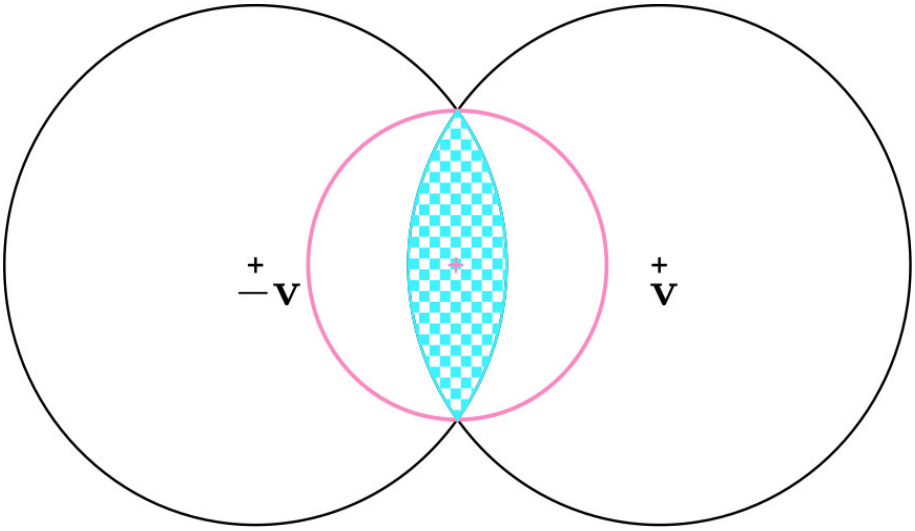


Figure 13.1: The HAETAE eyes, picture taken from [CCD+22].

version has $k = 4, \ell = 7$. This matrix can be interpreted as a $kn \times (\ell + k)n$ lattice system and the generic attacks use this interpretation. So far no better attacks are known for module structure than generic lattices.

The benefits of a bimodal distribution are that the signatures are distributed over two different centers, one linked to the secret $\mathbf{v}$ and one to $-\mathbf{v}$. Rejection sampling samples from the overlay of these two distributions and can stay in a narrower region around the middle, meaning that fewer rejections are encountered and that parameters can be chosen smaller for the same security level as the narrower size makes forgeries less likely.

In HAETAE the authors choose a hyperball distribution around the secrets and do rejection sampling to a hyperball around 0. In Figure 13.1, taken from the HAETAE submission [CCD+22] the secrets are the pupils of the eyes of the Haetae; the originally sampled signatures are the blue circles and the signatures that pass rejection are in the pink circle. The checkered region would be sampled twice as frequently and is thus rejected with 50% probability.

The May 2023 version of HAETAE computes bounds on how many times retry is needed which is ≤ 6.

13.2 Security

All changes from BLISS and Dilithium are covered in the security proofs (for issues see the next section). However, we were unable to evaluate their code `HAETAE.zip`, advertised for checking parameters. The code included required downloading the estimators for Dilithium and Kyber for module lattices, which are readily found on github, https://github.com/pq-crystals/security-estimates/tree/master but also needed a file `/home/julien/Documents/bliss-security/security-estimates/entropy\_coordinate\_hyperball.py` The directory `__pycache__` includes compiled versions and we managed to decompile to the missing files from executibles, however, running the code never terminated. The script does many searches to find best parameters for estimated attacks. In principle this uses https://estimate-all-the-lwe-ntru-schemes.github.io/docs/ which we used in all other chapters to evaluate the parameters, however for this one the matches were not clear.

13.3 Implementation considerations

The motivation stated for moving to the hyperball distribution, away from discrete Gaussians used in BLISS, is that sampling discrete Gaussians in constant time is hard. Indeed, implementations of BLISS have been attacked [BHLY16, PBY17, EFGT17] with cache-timing attacks. The implementation submitted in October 2023 does not avoid conditional branches and does not seem to be concerned with timing attacks, the May-2023 version has a new implementation with fewer obvious issues.
The original version shows how to sample in hyperballs using continuous Gaussians. No mention is made how that would be done but [HLS18] showed an easy and fast way to sample continuous Gaussians if the CPU supports floating-point arithmetic. Surprisingly, the May-2023 version changes the description and implementation to sampling discrete Gaussians to approximate continuous Gaussians to sample from the hyperball distribution. They now use the discrete Gaussian sampler from [BBE+19], which is advertised as running in constant time, but this raises the question of why they do not use this sampler as the main sampler, avoiding the detour via continuous Gaussians and hyperballs.
HAETAE advertises short signatures, where part comes from the bimodal distribution and part comes from a space-efficient encoding. This rANS encoding is missing in the implementation of the KpqC submission (v0.9) but was added in May (v1.0). However, the tables for the rANS encoding

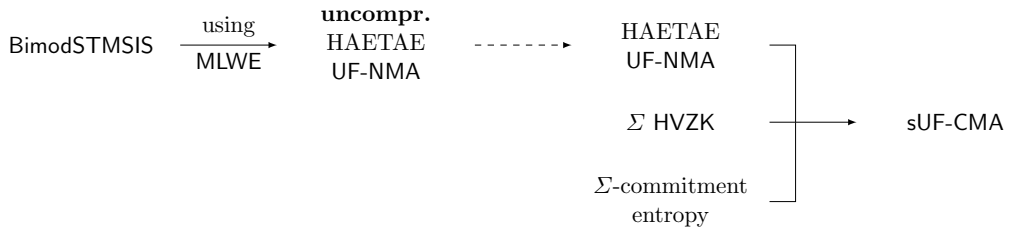
are very large. The designers announced that they would be able to work with smaller tables.
We noticed that HAETAE had KAT mismatches. Several implementation issues were pointed out in a NIST PQC Forum post by Markku Saarinen.
There were many changes between v0.9 and v1.0 incl. in security estimates and sizes and not all were explained. There are still lots of typos (Fig ??, missing increment of counter, ...) even in the v1.0 version.

13.4 Provable security claims

On a high level, HAETAE shares with Dilithium that it first builds a lattice-based identification scheme Σ, and then turns Σ into a signature scheme using the Fiat-Shamir paradigm. Consequently, the security reasoning for HAETAE is similar to the security reasoning for Dilithium, with appropriately adapted assumptions.
The core idea of the Fiat-Shamir paradigm is as follows: signatures consist of a prover commitment, together with a prover response. To tie the prover response to the to-be-signed-message, it is built by using as its challenge the hash value of the message and the prover commitment. The Fiat-Shamir with aborts paradigm additionally introduces rejection sampling. This is done to render the distribution of signatures sufficiently independent of sensitive information (the secret key), so that observing exchanged signatures will not help an attacker with forging a signature.

13.4.1 Security proof of HAETAE

Rejection sampling slightly complicates security proofs, in particular when concerning quantum attackers: two recent papers [BBD+23, DFPS23] pointed out a flaw in the security proof of Dilithium that was directly tied to how the proof addressed rejection sampling. HAETAE v1.0 took this into account by adapting the proof to the fix provided in [DFPS23].
To argue Strong Unforgeability under Chosen Message Attacks (sUF-CMA) of (deterministic) HAETAE, the submission follows the approach of Dilithium: it goes through an "implication chain" that relates computational hardness of (appropriately adapted) problems to security properties. We summarize the chain in the figure below.
Gap in UF-NMA reasoning. The specification claims that hardness of the Bimodal Self-Target MSIS problem (BimodSTMSIS) directly translates into the fact that HAETAE satisfies the intermediate security notion UF-NMA, which is then used to argue the aimed-at security property (sUF-CMA). The



definitions of BimodSTMSIS and UF-NMA, however, are only equivalent for the **uncompressed** version of HAETAE. The specification does not address whether/how compression affects UF-NMA security, and it is not obvious that compression does not decrease security.

Gap in HVZK reasoning. The specification aims to argue HVZK without presenting the underlying identification scheme Σ, thus forcing the reader to essentially re-do the proof. The proof sketch for HVZK also seems to introduce a LWR-like assumption that differs from more well-studied variants in the distribution of matrix and secret: it is assumed that $w = A\lfloor y \rceil$ is indistinguishable from a uniform element (both modulo q), where A is a public key (a randomly chosen compressed matrix). The specification does not make explicit how y is defined in this assumption. The likeliest interpretation is that y is computed like in the deterministic signing algorithm (Figure 7 in the specification), i.e., by using a – not further specified – expansion algorithm expandYbb.

Reading through the implementation package for v0,9, expandYbb is mentioned only in a comment but the steps match what is described in the main part of the specification, except for that b is not made dependent on the seed but sampled uniformly random, so this should fail for recomputing the KATs. In the specification the Gaussian sampler is assumed to be continuous. Looking at the code for `sampler_gaussian`, this uses a big table and is sure not safe against cache timing attacks (nor did it claim to be).

In v1.0 the code changed from calling `polydblveclk_uniform_hyperball` to calling a new function `polyfixveclk_sample_hyperball` using fixed-point arithmetic and the discrete Gaussian sampler. The latter is implemented with a very small CDT which might be small enough to fit in cache and to avoid cache-timing attacks. In general, there are far fewer branches in the code, but there might still be timing dependencies on the secret.

Asymptotic reasoning not necessarily applicable to parameter choices. The specification does not specify how closely the security notions are related to one another as it is missing concrete security bounds that quantify the relation. This leaves the reader with asymptotic reasoning,

meaning it is only shown that the implication chain will begin to be satisfied at the point where appropriately large parameters are chosen. This makes it hard to verify security for the concrete parameter choices made in the specification. Depending on how close (or distant) the relations are, the proof might not apply.

13.4.2 Interpretation of provable security results

The overall security reasoning looks sound up to the missing details in the security proof, which need to be added. Due to the asymptotic reasoning, it is hard to verify security for the concrete parameter choices in the specification. If a vulnerability were to be found, it would likely stem from

- a cryptanalytical break of one of the underlying computational problems (MSIS, MLWE, BimodSTMSIS and the unnamed LWR-like assumption needed for HVZK), or
- parameter choices that do not match the intended level of security.

13.5 General assessment

HAETAE is built from known good ingredients, starting from Dilithium and BLISS. It is also submitted to NIST in the current new call for signatures. The recent changes seem all like improvements but more explanations would be good to have, in particular on the use of discrete Gaussians in the sampler.

Chapter 14

MQ-Sign: A New Post-Quantum Signature Scheme based on Multivariate Quadratic Equations: Shorter and Faster

The digital signature scheme MQ-Sign [KASA22] is based on the trapdoor paradigm. In this setting, the secret key is composed of a central map

$$\mathcal{F} : (x_1, \ldots, x_n) \in \mathbb{F}_q^n \to \left(\mathcal{F}^{(1)}(x_1, \ldots, x_n), \ldots, \mathcal{F}^{(m)}(x_1, \ldots, x_n)\right) \in \mathbb{F}_q^m,$$

for which it is computationally easy to find a solution, aka a tuple $\mathbf{x} = (x_1, \ldots, x_n)$ such that $\mathcal{F}(\mathbf{x}) = 0$, and two bijective affine mappings $\mathcal{S} \in \mathrm{AGL}_n(q)(\mathbb{F}_q), \mathcal{T} \in \mathrm{AGL}_m(q)(\mathbb{F}_q)$. These mappings are used to hide the special structure of $\mathcal{F}$ that allows us to compute solutions easily. The public key is obtained by mapping $\mathcal{F}$ to another quadratic map $\mathcal{P}$, such that $\mathcal{P} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S}$. This results in a multivariate system that is hard to solve and indistinguishable from a randomly generated multivariate quadratic system unless the some structure of $\mathcal{F}$ can be retrieved from $\mathcal{P}$,
Define a hash function to map to $\mathbb{F}_q^m$ and let $\mathbf{y} = \mathsf{hash}(M)$, then a signature on M is a preimage $\mathbf{x}$ of $\mathbf{y}$ under the public map. To verify the signature, compute $\mathbf{y}$ and accept if this matches $\mathcal{P}(\mathbf{x})$. To find such an $\mathbf{x}$, the signer uses the knowledge of the affine maps and the ability to compute preimages of the central map to compute $\mathbf{y}' = \mathcal{T}^{-1}(\mathbf{y}), \mathbf{x}' = \mathcal{F}^{-1}(\mathbf{y}')$, and $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{x}')$, where $\mathcal{F}^{-1}(\mathbf{y}')$ means the computation of a (typically not unique) preimage of $\mathbf{y}'$. For a secure scheme it should be hard to find $\mathbf{x}$ given $\mathbf{y}$ and the public key.

14.1 Unbalanced Oil and Vinegar

One of the oldest trapdoor constructions is the Unbalanced Oil and Vinegar signature scheme, proposed by Kipnis, Patarin, and Goubin [KPG99] as a modification of the oil and vinegar scheme of Patarin [Pat97] that was broken by Kipnis and Shamir in 1998 [KS98].
In the oil and vinegar construction, the variables in the central map are divided in two distinct sets, called vinegar variables and oil variables. The vinegar variables are combined quadratically with all of the variables, while the oil variables are only combined quadratically with vinegar variables and not with other oil variables. This special structure serves as a trapdoor that allows to find preimages of the central map. Formally, the central map is defined as $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, with polynomials

$$\mathcal{F}^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in V} \gamma_{ij}^{(k)} x_i x_j + \sum_{i \in V, j \in O} \gamma_{ij}^{(k)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} x_i + \alpha^{(k)} \quad (14.1)$$

where $n = v + m$, and $V = \{1, \ldots, v\}$ and $O = \{v + 1, \ldots, n\}$ denote the index sets of the vinegar and oil variables, respectively.
The affine mapping $\mathcal{T}$ is omitted, as it can be shown that if an oil and vinegar central map is used in the standard MQ construction, $\mathcal{T}$ does not add to the security of the scheme. Hence the secret key consists of a linear transformation $\mathcal{S}$ and central map $\mathcal{F}$, while the public key is defined as $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$. As with any trapdoor-based multivariate signature scheme, to sign a message, we need to find a preimage of $\mathcal{F}$. This can be done by simply fixing the vinegar variables to some random values. The resulting system is a linear system of m equations in m variables, and thus, it has a solution with probability around $1 - 1/q$. If the obtained system does not have a solution, we repeat the procedure with different values for the vinegar variables, otherwise we apply the inverse affine transformation and obtain a signature.

14.2 MQ-Sign description

UOV signature schemes are attractive because they have very small signatures and fast verification. On the downside, they have large public and secret keys. As a result, variations of the traditional UOV scheme are usually developed with the goal to reduce the size of the public key. These variations have additional structure that might compromise the security of the scheme. A notable example of such a scheme is Rainbow [DS05], which

was a finalist in the NIST PQC standardization process as [DCP+20], before Beullens showed that it does not meet the security requirements [Beu22]. It is a great challenge to develop UOV variations with additional structure that does not compromise the security of the scheme or where the trade-off results in smaller key sizes.
MQ-Sign is a UOV-based signature scheme, where the main focus is to reduce the size of the secret key compared to traditional UOV. This is achieved by using sparse polynomials for the quadratic part of the central map. The scheme uses inhomogenous polynomials and each polynomial of the central map can be written as

$$\mathcal{F}^{(k)} = \mathcal{F}_V^{(k)} + \mathcal{F}_{OV}^{(k)} + \mathcal{F}_{L,C}^{(k)}$$

where $\mathcal{F}_V^{(k)} = \sum_{i\in V, j\in V} \gamma_{ij}^{(k)} x_i x_j$ and $\mathcal{F}_{OV}^{(k)} = \sum_{i\in V, j\in O} \gamma_{ij}^{(k)} x_i x_j$. These can alternatively be referred to as the vinegar-vinegar quadratic part and the vinegar-oil quadratic part. Finally, $\mathcal{F}_{L,C}^{(k)}$ refers to the linear and constant part of the polynomials. In the following, we ignore the linear and constant parts, since our attack does not use them. The quadratic homogenous part of the sparse polynomials is defined as $\mathcal{F}_V^{(k)} + \mathcal{F}_{OV}^{(k)}$ such that

$$\begin{aligned} \mathcal{F}_V^{(k)} &= \sum_{i=1}^{v} \alpha_i^{(k)} x_i x_{(i+k-1(\bmod v))+1} \\ \mathcal{F}_{OV}^{(k)} &= \sum_{i=1}^{v} \beta_i^{(k)} x_i x_{(i+k-2(\bmod m))+v+1}. \end{aligned} \tag{14.2}$$

The size of the secret key is thus reduced to $2vm$ field elements.
The MQ-Sign proposal provides a parameter selection for four variations of the scheme: MQ-Sign-SS, MQ-Sign-RS, MQ-Sign-SR and MQ-Sign-RR. The first S/R in the suffix specifies whether $\mathcal{F}_V$ is defined with sparse or random polynomials. The second S/R refers to the same property, but for $\mathcal{F}_{OV}$. Note that the variation MQ-Sign-RR corresponds to the standard UOV scheme defined with inhomogenous polynomials.
Discussions about UOV systems typically present the maps $\mathcal{F}^{(k)}$ as $n \times n$ matrices $F^{(k)}$ giving the coefficients of the polynomials so that $(x_1, x_2, \ldots, x_n)^\top F(x_1, x_2, \ldots, x_n) = \mathcal{F}^{(k)}$ and $F^{(k)}$ is an upper triangular matrix so that the coefficient of $x_i x_j$ for $i \leq j$ appears at position (i, j) and not at position (j, i). Given that there are no terms involving two oil variables the matrix has the form

$$F^{(k)} = \begin{pmatrix} F_1^{(k)} & F_2^{(k)} \\ 0 & 0 \end{pmatrix}$$

where F_1 is an upper triangular matrix. In this representation, the map S is given as an $n \times n$ matrix S and the k-th polynomial in the public key becomes $P^{(k)} = S^\top F^{(k)} S$.

14.3 First algebraic attack

In March 2023, a first algebraic attack on MQ-Sign was proposed by Aulbach, Samardjiska, and Trimoska [AST23] that exploits the sparseness of the vinegar-oil part of the secret key. The attack also relies on the fact that the map $\mathcal{S}$ is chosen to be given by a matrix of the following form

$$S = \begin{pmatrix} I_{v\times v} & S_1 \\ 0_{m\times v} & I_{m\times m} \end{pmatrix}. \tag{14.3}$$

This typically does not reduce the security of a UOV-based scheme because it was shown in [Pet13] that for any instance of a UOV secret key $(\mathcal{F}', S')$, there is an equivalent key $(\mathcal{F}, S)$ where S has the form as in (14.3). The advantage of using the *equivalent keys* form is a reduction of the size of the secret key, as only the submatrix S_1 needs to be stored. This optimization is used in most modern UOV-based schemes, including the original MQ-Sign submission.
With the equivalent keys structure of the map S, the computation of the public key for UOV-like signatures schemes can be written as

$$\begin{pmatrix} I & 0 \\ S_1^\top & I \end{pmatrix} \begin{pmatrix} F_1^{(k)} & F_2^{(k)} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix},$$

which in upper-triangular matrix form simplifies to

$$\begin{pmatrix} P_1^{(k)} & P_2^{(k)} \\ 0 & P_4^{(k)} \end{pmatrix} = \begin{pmatrix} F_1^{(k)} & (F_1^{(k)} + F_1^{(k)\top})S_1 + F_2^{(k)} \\ 0 & \mathsf{Upper}(S_1^\top F_1^{(k)} S_1 + S_1^\top F_2^{(k)}) \end{pmatrix} \tag{14.4}$$

adding the entry at (j, i) to the entry of (i, j) for $j > i$.
From this representation, we can easily see that $P_1^{(k)} = F_1^{(k)}$ and we have the following relation

$$P_2^{(k)} = (P_1^{(k)} + P_1^{(k)\top})S_1 + F_2^{(k)}. \tag{14.5}$$

From (14.5), we can derive a polynomial system containing variables from the matrix S and the matrices $F_2^{(k)}$, and notably, the system is linear. Furthermore, $F_2^{(k)}$ is sparse and the MQ-Sign specification defines exactly which entries are zero. The main idea in [AST23] is to build this system and remove

Variant	Security Category					
	I		III		V	
	Before	After	Before	After	Before	After
MQ-Sign-SS	15561	26173	37729	63521	66421	111749
MQ-Sign-RS	133137	143749	485281	511073	1110709	1156037

Table 14.1: Size (in Bytes) of the secret key of MQ-Sign before and after the update of the implementation.

all of the equations where the $F_2^{(k)}$ entry is not zero. The resulting subsystem contains only the vm variables from the matrix S. The number of equations that we obtain is $mv(m-1)$, so we expect to have vm linearly independent equations and solve the system easily through Gaussian Elimination.
The attack was implemented in both SageMath [The22] and MAGMA [BCP97], and it was reported to run in 0.6 seconds for the proposed parameters for security level I, 2.3 seconds for security level III and 6.9 seconds for security level V.
The authors of MQ-Sign acknowledged that the attack works and specified that MQ-Sign should be implemented using a random affine map S instead of the equivalent keys form. They added that they would update the implementation accordingly. Table 14.1 shows the impact of this modification on the secret key sizes, compared to the sizes reported in the MQ-Sign specification.

14.4 Second algebraic attack

Less than a month after the first attack, Ikematsu, Jo, and Yasuda proposed another algebraic attack that also targets the MQ-Sign-{S/R}S variants but is not dependent on S having the equivalent keys structure [IJY23]. This is achieved by taking an approach that aims at recovering the trapdoor space $\mathbf{O}$ from the public key. For all UOV-based schemes, it is known that $\mathcal{P}$ vanishes on a secret linear subspace $\mathbf{O} \subset \mathbb{F}_q^n$ and that if this subspace is recovered by an attacker, any signature can be forged easily. Concretely, the task is to find m linearly independent vectors $\mathbf{o}_1, \ldots, \mathbf{o}_m \in \mathbb{F}_q^n$ such that

$$\mathbf{o}_i^\top P^{(k)} \mathbf{o}_j = 0, \quad \text{for } i, j, k \in \{1, \ldots, m\}.$$

This algebraic attack starts by rewriting the relations $P^{(i)} = S^\top F^{(i)} S$ as

$$P^{(i)} S^{-1} = S^\top F^{(i)}$$

Since in MQ-Sign-{S/R}S the matrices $F^{(i)}$ are sparse, separating these relations by column results in

$$P^{(i)}\mathbf{s}'_{(i+m-1(\bmod m))} = \mathbf{s}_m \beta_m^{(i)}, \qquad \text{for } i \in \{1, \ldots, m\},$$

where $\beta_m^{(i)}$ are single entries from $F^{(i)}$, instead of column vectors in the general case. This idea starts a chain of reasoning that, with some guessing of variable assignments that remains practical, results in recovering the first two vectors. When two vectors of the $\mathbf{O}$ space are recovered, it is easy to recover the entire space $\mathbf{O}$. Recent work shows how the entire space can be practically recovered with even a single vector [ACK+23].
Similarly as in [AST23], this attack was verified with an implementation in MAGMA that was reported by the authors to run in no more that 30 minutes for all security levels.

14.5 Vulnerabilities in third variant

As a result of these two attacks, the authors of MQ-Sign announced that the MQ-Sign-SS and MQ-Sign-RS variants are removed and a binding technique is added in the implementation of MQ-Sign so that a signature is identified with a unique public key and message to prevent potential attacks. The updated implementation still uses a map S with the equivalent keys structure, but there are no attacks announced that exploit this structure for the remaining variants.
The first variant that is not impacted by the algebraic attacks on MQ-Sign is MQ-Sign-SR. In this variant, only the vinegar-vinegar quadratic part is sparse and the vinegar-oil quadratic part is random. Intuitively, it seems that there is no danger in having a specific structure in the vinegar-vinegar part of the secret key, as this part corresponds exactly to the vinegar-vinegar part of the public key. Indeed, it can be seen from (14.4) that $P_1^{(k)} = F_1^{(k)}$ for all $k \in \{1, \ldots, m\}$, so the vinegar-vinegar part of the secret key can always be derived from the public key. This is inherent to the general UOV construction, because of the equivalent keys attacks and the extensive analysis of UOV suggests that there is currently no known way that this can be exploited without using side-channel information. However, further analysis is needed to see whether attacks that exploit the fact that the vinegar-vinegar part of the secret key is sparse (and not only that it is known) can be developed, as this structure is specific to the recently proposed MQ-Sign candidate.
We outline here a first idea of such an attack, and point out where the key vulnerabilities in this variant come from. Consider again that our goal is to

find a vector $\mathbf{o}$ such that $\mathbf{o}^\top P^{(k)}\mathbf{o} = 0$, for all k. Rewriting this in block matrix form

$$\begin{pmatrix}\mathbf{o}_V^\top & \mathbf{o}_O^\top\end{pmatrix}\begin{pmatrix}P_1^{(k)} & P_2^{(k)}\\ 0 & P_4^{(k)}\end{pmatrix}\begin{pmatrix}\mathbf{o}_V\\ \mathbf{o}_O\end{pmatrix} = 0$$

gives us the following algebraic constraint

$$\mathbf{o}_V^\top P_1^{(k)}\mathbf{o}_V + \mathbf{o}_V^\top P_2^{(k)}\mathbf{o}_O + \mathbf{o}_O^\top P_4^{(k)}\mathbf{o}_O = 0. \tag{14.6}$$

From (14.6) we can see that fixing all variables in $\mathbf{o}_O$ gives us a system of m equations in v variables where the quadratic part depends only on the sparse $P_1^{(k)}$. Since v is greater than m, we can still fix another $(v-m)$ variables and expect to have a solution. We notice also that because of the sparseness in $P_1^{(k)}$, we can extract a subset of equations that yields a bilinear system. For instance, for m even, there is a subset of $\frac{m}{2}$ equations that is bilinear in the sets of variables $\{\mathbf{x}_1, \mathbf{x}_3, \ldots, \mathbf{x}_{m-1}\}$ and $\{\mathbf{x}_2, \mathbf{x}_4, \ldots, \mathbf{x}_m\}$, where we denote by $\mathbf{x}_i$ the variables in vector $\mathbf{o}_V$. Hence, fixing all variables in the set of odd variables yields a linear system in the even variables. Specifically, we can fix $(v-m)$ variables and enumerate the rest with the usual cost of enumeration. Then, from the linear system of $\frac{m}{2}$ equations in $\frac{v}{2}$ variables, we obtain a solutions space of dimension $\frac{v-m}{2}$. Substituting these findings in the remaining quadratic equations from the initial system, we are faced with a quadratic system of $\frac{m}{2}$ equations in $\frac{v-m}{2}$ variables, which we can solve using Gröbner basis techniques. The overall cost of this attack is the cost of the enumeration times the cost of the Gröbner algorithm, which amounts approximately to 2^{111} for the security level I, 2^{170} for security level III and 2^{228} for security level V MQ-Sign parameters. The attack was developed in collaboration with Aulbach and Samardjiska, and the complexity analysis and the methodology for finding bilinear subsystems will be detailed in an extended version of [AST23].

This simple attack shows that MQ-Sign-SR falls below the security requirements by a small margin. We do not think that simply increasing the parameters would be sufficient to meet the security levels, as there can be more elaborate attacks that exploit this or other bilinear subsystems that arise from the sparseness in $P_1^{(k)}$. However, a straightforward way to mitigate this attack would be to remove the use of equivalent keys structure of S. The impact of this countermeasure on the secret key sizes is summarized in Table 14.2. If this countermeasure is adopted, the attack outlined here would be avoided, but further research is needed to see whether the sparseness of $P_1^{(k)}$ can still be exploited in a similar manner.

Variant	Security Category					
	I		III		V	
	Before	After	Before	After	Before	After
MQ-Sign-SR	164601	175213	610273	636065	1416181	1461509

Table 14.2: Size (in Bytes) of the secret key of MQ-Sign-SR before and after adopting the potential countermeasure.

Security Category	MQ-Sign-SR with random S	MQ-Sign-RR with equivalent keys S	UOV in [BCH+23]
I	175213	282177	237896
III	636065	1057825	1044320
V	1461509	2460469	2436704

Table 14.3: Size (in Bytes) of the secret key of MQ-Sign and another UOV variant.

14.6 Secure MQ-Sign variant

The second remaining variant is MQ-Sign-RR. There is strong confidence that this variant is secure, especially because it is equivalent to the traditional UOV scheme. The confidence in its security comes with the disadvantage of not having any reduction in the secret key size. Other techniques for storing the secret key can be used that are common in the literature, such as for instance, storing a basis of the oil space instead of the central map [BCH+23]. UOV with the implementation and parameter choices outlined in [BCH+23] was submitted to the additional call for signatures by NIST in summer 2023. It differs from MQ-Sign-RR mainly in the representation of the secret keys and, as a result, the signing algorithm. Another difference is that MQ-Sign uses the Block Matrix Inversion (BMI) method proposed in [SLK22] and offline precomputation to improve the signing runtime. Even though the secret keys of UOV in [BCH+23] and MQ-Sign-RR consist of entirely different data structures, they have comparable sizes. Table 14.3 shows this comparison, including also MQ-Sign-SR with a random S, as this variant is not affected by current attacks.

14.7 General assessment

For security proofs, MQ-Sign refers to [SSH11], an article studying the provable security of UOV schemes. These arguments should be included in a future version of the scheme.

The analysis of generic attacks against MQ systems in the MQ-Sign submission is adequate and we are not aware of any attack against the systems with sizes as described in Table 14.3.
There has been a lot of research over the past year on UOV schemes due to the NIST call for additional signatures which closed in June 2023. For round 2, the MQ-Sign authors may want to adjust parameters and implementation in order to adopt some of the new improvements found during the past year since they submitted to KpqC.

Chapter 15

NCC-Sign: A New Lattice-based Signature Scheme using Non-Cyclotomic Polynomials

The main idea of NCC-Sign is to take the Dilithium design and replace the mathematical structure used. While Dilithium uses a module lattice, NCC-Sign moves to ideal lattices.

15.1 System description

NCC-Sign [SKA22] is based on Lyubashevky's signature scheme [Lyu12] using Fiat–Shamir with aborts, a signature compression technique by Bai and Galbraith [BG14], and the public-key compression technique from Dilithium [LDK+20] on the signature side and on NTRU Prime [BCLv17, BBC+20b] and NTTRU [LS19] on the ideal lattice side.
The main part of the submission, as also reflected in the title, uses ideal lattices over the NTRU Prime field $R = \mathbf{Z}[x]/(x^p - x - 1)$ modulo a prime q, where q is chosen such that $x^p - x - 1$ is irreducible modulo q and that q is inert in $\mathbf{Q}[x]/(x^p - x - 1)$. The authors follow NTRU Prime in pointing to security concerns related to the cyclotomic structure and the many subfields present in the typical choice of $x^n + 1$ for $n = 2^d$.
The submission also considers a version using cyclotomic polynomials of the form $x^{2n} - x^n + 1$ for $n = 2^a 3^b$. These cyclotomic polynomials were proposed in [LS19] to add flexibility in the dimension beyond $n = 2^d$ while keeping the speed benefits of NTT-friendly rings. In the original NCC-Sign submission

from October 2022 this version is presented in Section 3.5 and Table 6 while all implementation considerations cover only the non-cyclotomic case. In the updated version [SKA23], labeled v1.0 in that document, two sets of parameters are proposed for this case and implementation results indeed show better speeds, however, most of the text still focuses on the non-cyclotomic case.

KeyGen, Sign, and Verify as well as the supporting algorithms match those of Dilithium. The difference is that where Dilithium uses module lattices, NCC-Sign uses ideal lattices. The supporting algorithms are defined for the coefficients and thus match 1-to-1, for the other functions matrices of polynomials are replaced by polynomials.

KeyGen generates a public polynomial $\mathbf{a} \in R_q$ from some seed ζ, this seed forms the first part of the public key. The second part part is an RLWE sample using $\mathbf{a}$: Pick small $(\mathbf{s}_1, \mathbf{s}_2)$ and compute $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$, where "small" means that the coefficients are in $\{0, \pm 1, \pm 2\}$; the July-2023 version includes also an option for using $\in \{0, \pm 1\}$. The public key includes only the top part of $\mathbf{t}$ while the bottom part is included in the secret key along with the small polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$. The secret key additionally includes the seed ζ, the hash $ph = \mathrm{H}(\zeta, \mathbf{t}_1)$ of the public key, and a string dK to generate pseudo-random numbers in signing.

Top and bottom parts of $\mathbf{t}$ are defined as follows: Assume that the coefficients of $\mathbf{t}$ are in $[0, q-1]$. Let $\mathbf{t}_0$ be the polynomial whose coefficients are computed from the coefficients of $\mathbf{t}$ taking the remainder under division by 2^d using representatives inside $(-2^{d-1}, 2^{d-1}]$. Then $\mathbf{t}_1 = (\mathbf{t} - \mathbf{t}_0)/(2^d)$. Since the division here is by a power of 2, the top part basically means the top bits of each coefficient, apart from the detail that the remainder can be negative which then adds 1 to the top part. Other functions use top and bottom parts for more general moduli γ, using the same approach of computing the remainder centered around 0 and then taking the quotient of division by γ after subtracting the remainder.

The signature should show that the signer indeed knows $\mathbf{s}_1, \mathbf{s}_2$ matching $\mathbf{t}_1$. To sign message M, first a random commitment $\mathbf{y} \in R_q$ with restricted coefficients is sampled; this process may need to be repeated (the aborts part), hence the sampling includes a counter κ. The randomized version version picks a 512-bit ρ at random while for the deterministic version ρ depends on M, ph, and dK. Here, "restricted" is not as small as in the key generation but coefficients of $\mathbf{y}$ are in $(-2^{17}, 2^{17}]$ for the 128-bit security level and the range doubles for each level. Then $\mathbf{w} = \mathbf{a}\mathbf{y}$ is computed and only the quotient after division by some γ_2 is taken, the remainder is centered around 0 and γ_2 is co-prime to q and has $7 - 5$ bits less than q. (The full details include one corner case, see the Decompose function.)

The challenge is then given by $\tilde{c} = \mathrm{H}(\mu, \mathbf{w}_1) \in \{0,1\}^{256}$, where $\mu = \mathrm{H}(ph, M)$. This $\tilde{c}$ is then used to deterministically sample a fixed-weight polynomial $\mathbf{c}$, having τ coefficients in $\{-1, 1\}$ and the rest being 0, where $\tau = 25$ for the smallest parameters and 32 for the largest.
The polynomial $\mathbf{z} = \mathbf{y} + \mathbf{cs}_1$ then uses the secret key. However, the public key is computed using also $\mathbf{s}_2$ and the public key only includes the top part of $\mathbf{t}$, hence the next steps in signature generation ensure that verification can proceed. First it is checked $\mathbf{z}$ does not leak information on $\mathbf{s}$, for that it is checked that none of the coefficients are larger than some bound $\gamma_1 - \beta$, where γ_1 was the bound on the coefficients of $\mathbf{y}$ and $\beta = 4\tau$. This means that $\mathbf{z}$ does not depend on the secret. Signature verification for Bai–Galbraith signature compression reconstructs the top part of $\mathbf{w}$ as the top part of $\mathbf{az} - \mathbf{ct} = \mathbf{ay} + \mathbf{acs}_1 - \mathbf{cas}_1 - \mathbf{cs}_2 = \mathbf{w} - \mathbf{c} - \mathbf{s}_2$ which matches the top part of $\mathbf{w}$ if $\mathbf{s}_2$ is sufficiently small. This is checked by checking that the centered remainder of $\mathbf{w} - \mathbf{cs}_2$ after division by $2\gamma_2$ has no coefficient larger than $\gamma_2 - \beta$, because $\beta = 2\eta\tau$ is the maximum size a coefficient of $\mathbf{cs}_2$ can have (note that reductions modulo $x^p - x - 1$ cause the extra factor of 2 here. If either of these are violated, the counter κ is incremented and a new $\mathbf{y}$ is sampled. There is an indentation error in the signing function in both versions of NCC-Sign [SKA22, SKA23] as Step 20 needs to be indented less (be at the same level as **if** and **else**).
NCC-Sign follows Dilithium in additionally compressing the public key (including only $\mathbf{t}_1$ instead of $\mathbf{t}$), which means that only $\mathbf{ct}_1$ is available, leading to $\mathbf{w} - \mathbf{cs}_2 + \mathbf{ct}_0$. The signature includes a vector $\mathbf{h}$ of hints, which are 1 in the positions in which the high parts of $\mathbf{w} - \mathbf{cs}_2$ and of $\mathbf{w} - \mathbf{cs}_2 + \mathbf{ct}_0$ differ. Valid signatures are limited in how large the Hamming weight of $\mathbf{h}$ is permitted to be as these hints give extra flexibility to a forger. If the calculated $\mathbf{h}$ has too large weight κ is incremented and a new $\mathbf{c}$ is sampled.
Eventually all checks succeed and the signature is $(\tilde{c}, \mathbf{z}, \mathbf{h})$.
To verify signature $(\tilde{c}, \mathbf{z}, \mathbf{h})$ on M compute $\mathbf{c}$ from $\tilde{c}$ and compute the high part $\mathbf{w}'_1$ of $\mathbf{az} - \mathbf{ct}_1 \cdot 2^d$ using the hint vector $\mathbf{h}$. If this computation worked correctly, $\mathbf{w}_1 = \mathbf{w}'_1$ and $\tilde{c} = \mathrm{H}(\mu, \mathbf{w}'_1)$, hence this forms the verification check along with checking the weight of $\mathbf{h}$ and the coefficient sizes of $\mathbf{z}$.
By the above, an honestly generated signature passes verification.

15.2 Security

The suitability of the underlying lattice problem has been argued in NTRU Prime and NTTRU respectively. While NTRU Prime cautions of using cyclotomic lattices no actual attacks on RLWE or RLWR are known. The general

strategy of NCC-Sign equals that of Dilithium and is thus well studied. As we comment in the section on security proofs, the differences are not fully explored and it is not clear that the proofs hold, however, we have not been able to turn the differences into attacks. The most visible difference, caused by the asymmetry in how reductions modulo $x^p - x - 1$ affect the coefficients, has been taken into account by the designers. there is no matching counterpart for the cyclotomic polynomial. This leaves generic attacks as the main attack avenue and guidance on choosing parameters.

15.2.1 Generic lattice attacks

Tables 15.1 and 15.2 contains the results of the estimator for the BKZ lattice attacks from Albrecht, Curtis, Deo, Davidson, Player, Postlethwaite, Virdia and Wunderer found in [ACD+18] for the concrete and conservative parameters respectively. For more information see Section 2.1. The input of the code is given in Table 15.3 for the concrete parameters and Table 15.4 for conservative parameters.

Parameter n	1021		1429		1913	
Type	primal	dual	primal	dual	primal	dual
Q-Core-Sieve	112	127	173	194	249	274
Q-Core-Sieve + O(1)	128	141	189	207	265	287
Q-Core-Sieve (min space)	126	141	194	212	279	303
Q-β-Sieve	121	135	182	201	258	283
Q-8d-Sieve + O(1)	143	155	204	220	280	302
Core-Sieve	124	138	191	210	274	298
Core-Sieve + O(1)	140	152	207	224	290	314
Core-Sieve (min space)	156	169	240	259	345	367
β-Sieve	132	146	200	218	284	308
8d-Sieve + O(1)	154	166	221	239	305	326
Q-Core-Enum + O(1)	138	158	247	273	397	420
Lotus	143	166	270	290	448	472
Core-Enum + O(1)	276	279	493	489	793	777
8d-Enum (quadratic fit) + O(1)	310	316	587	561	1046	931

Table 15.1: Estimations for security level for the concrete parameters found with code by [ACD+18]

Parameter n	1201		1607		2093	
Type	primal	dual	primal	dual	primal	dual
Q-Core-Sieve	131	146	190	211	263	288
Q-Core-Sieve + O(1)	147	160	206	224	279	304
Q-Core-Sieve (min space)	147	162	213	233	295	319
Q-β -Sieve	140	155	199	218	273	298
Q-8d-Sieve + O(1)	161	174	221	237	294	317
Core-Sieve	144	159	209	229	290	313
Core-Sieve + O(1)	160	175	225	243	306	329
Core-Sieve (min space)	182	196	263	283	365	387
β-Sieve	153	168	218	237	300	323
8d-Sieve + O(1)	175	186	240	258	321	343
Q-Core-Enum + O(1)	170	191	279	301	427	454
Lotus	180	202	308	329	485	502
Core-Enum + O(1)	339	346	557	552	854	834
8d-Enum (quadratic fit) + O(1)	386	379	677	646	1149	1042

Table 15.2: Estimations for security level for the conservative parameters found with code by [ACD+18]

15.3 Implementation considerations

NCC-Sign required fixing `CRYPTO ALGNAME` but then still had issues with signing large messages and sometimes on short messages, too.
The designers observe that reducing $\mathbf{cs}_i$ modulo $x^p - x - 1$ leads to higher weight in the bottom half of the result. They modify the sampler to split $\mathbf{c}$ into top and bottom parts $\mathbf{c} = \mathbf{c}_2 + x^{p_2}\mathbf{c}_1$ with weight τ_i in $\mathbf{c}_i$. Taking $\mathbf{s}_i$ with extremal coefficients $\pm\eta$ leads to a polynomial with constant term and coefficients of x^j with $j \geq p_2$ bounded by $\beta_2 = (2\tau_1 + \tau_2)\eta$ and coefficients of x^j with $1 \leq j < p_2$ bounded by $\beta_1 = 2(\tau_1 + \tau_2)\eta$. They thus suggest to change the distribution of $\mathbf{c}$ to have $\tau_1 < \tau_2$ and to change the size constraints to using β_1 and β_2 instead. There is an error in Table 7 of [SKA22] in that the columns labeled τ_1, τ_2 should be labeled β_1, β_2. The choice of variable name κ for the challenge entropy is bad as κ in the signing algorithm is a counter. While it is confusing that $\tau_1 + \tau_2 \neq \tau$ and no comment is made to this regard, the examples have $\tau_1 + \tau_2 > \tau$ which is good as the sampler loses some entropy by fixing a split of the weight, e.g., the first parameter set has $\tau_2 = 14, \tau_1 = 12$ for $\tau = 25$.
NCC-Sign gains flexibility over power-of-two cyclotomics by permitting any prime p as length. NTRU Prime has shows that arithmetic in these rings can

n	1021	1429	1913
Standard deviation	1.414	1.414	1.414
q	8339581	8376649	8343469
Secret distribution	$(-2, 2)$	$(-2, 2)$	$(-2, 2)$
m	2458	3605	5055
Claimed security	147	211	291
Category	1	3	5
Ring	$x^n - x - 1$	$x^n - x - 1$	$x^n - x - 1$

Table 15.3: Input for the concrete parameters in the code by [ACD+18]

n	1021	1429	1913
Standard deviation	1.414	1.414	1.414
q	17279291	17305741	17287423
Secret distribution	$(-2, 2)$	$(-2, 2)$	$(-2, 2)$
m	3186	4251	5385
Claimed security	167	229	298
Category	1	3	5
Ring	$x^n - x - 1$	$x^n - x - 1$	$x^n - x - 1$

Table 15.4: Input for the conservative parameters in the code by [ACD+18]

be competitive with implementations using NTT but NCC-Sign needs larger parameters than the KEM. It is likely that further speedups are possible, using the tooling for code generation in NTRU Prime, but at this moment the speed is much slower. The cyclotomic rings can use the implementation from NTTRU and do obtain better speed in the experiments reported in [SKA23]. In general, there is room for improvement.

15.4 Provable security of the signature scheme

The following is based on the submitted version of NCC-Sign [SKA22]; the more recent version [SKA23] from July 2023 comments on the flaw in the Dilithium security proof but the proof remained a sketch.
One way to prove that NCC-Sign is secure, is showing that the Dilithium

proof [KLS18] applies with appropriately adapted assumptions. This is also the approach taken by the NCC-Sign authors. In more detail, this requires to show security of the signature scheme under No-Message Attacks (UF-NMA) as well as non-aborting Honest-Verifier Zero-Knowledge (naHVZK) of the underlying identification scheme.
The presented proof is not convincing as it is neither giving a full proof, nor working out exactly which parts of the Dilithium proof have to be adopted, why they have to be adopted, and how.
For UF-NMA security, the given proof sketch is so much compressed, that it is not possible to recover a meaningful proof without redoing the proof. With regard to HVZK, the specification tries to argue HVZK without even presenting the underlying identification scheme. Hence, also here the reader is left to essentially redo the proof.
We did not redo the proofs. While it is believable that the proofs go through with the changes in assumptions made by the authors, we do not think one can rely on this without someone carefully doing the proofs and writing them out to enable public scrutiny.
Even if the proofs were well done, it has to be noted that a recent paper [BBD+23] (published after the KpqC deadline) pointed out a flaw in the security proof of Dilithium. While [BBD+23] also presents a fix, this fix comes with a loss in tightness that has to be analyzed for the specific scheme. Although it is likely that the analysis for Dilithium carries over to NCC-Sign with minimal changes due to their similarity, this should be done with care.

15.5 General assessment

NCC-Sign is built on a solid design history and we did not identify any weakness. The gap in Dilithium's security proof (2 papers at Crypto 2023) applies to whole family of signature systems using Fiat–Shamir with aborts. But the changes in how the rings are chosen mean that the proofs need to be done carefully to deal with differences in the distributions. The authors seem aware of these differences for the implementation parts and developed a sampling strategy suitable for reductions modulo $x^p - x - 1$ but more work is needed to show that the proofs still apply.

Chapter 16

Peregrine: Toward Fastest FALCON Based on GPV Framework

16.1 Introduction

Peregrine [SKLN22a, SKLN22b] is a signature scheme based on Falcon [PFH+22]. It is similar to Falcon in that it is based on the GPV framework due to Gentry, Peikert and Vaikunthanathan [GPV08], using NTRU lattices, and in those aspects Peregrine is actually more or less identical to Falcon. It is based on the hardness of the SIS problem, and security arguments are given in the quantum random oracle model. The main difference is in the trapdoor sampler. The main design rationale here is performance, and indeed it seems that for signature generation Peregrine outperforms Falcon by a factor of about 3.[1] Other design rationales are a simpler implementation resulting in better side channel resistance than Falcon offers, but the submission already itself notices that this is achieved at the price of a loss of security assurance.

16.2 Overview

The GPV framework has the following structure.
Key generation: the public key is a full rank matrix $A \in \mathbb{Z}_q^{m \times n}$ (with $m > n$), of which the rows are seen as the basis of a q-ary lattice Λ. The private key

[1] This is reflected in the choice of name: the peregrine bird is a subspecies of falcon that is known to be the fastest of all animals, with measured speeds of well over 300 km/h.

is a matrix $B \in \mathbb{Z}_q^{m \times m}$ of which the rows form a basis of the q-ary lattice $\Lambda^\perp$, so that $B \cdot A^\top = 0$.
Signature generation: for a message m with hash $H(m)$, find a $\mathbf{c} \in \mathbb{Z}_q^m \notin \Lambda^\perp$, satisfying $\mathbf{c} \cdot A^\top = H(m)$, then find a $\mathbf{v} = \mathbf{v}_0 \cdot B \in \Lambda^\perp$ close to $\mathbf{c}$ (so $\mathbf{v}_0 \in \mathbb{Z}^m$), the signature is $\mathbf{s} = \mathbf{c} - \mathbf{v}$, which is short by construction.
Signature verification: $\mathbf{s} \cdot A^\top = \mathbf{c} \cdot A^\top - \mathbf{v}_0 \cdot B \cdot A^\top = H(m)$.
For NTRU details we refer to the submission [SKLN22a] and Chapters 5 and 17.

16.3 The trapdoor sampler

The trapdoor sampler in the signature generation is the method to find $\mathbf{v} \in \Lambda^\perp$ close to the given $\mathbf{c} \notin \Lambda^\perp$. Two basic methods are known. Babai's round-off algorithm computes $\mathbf{c}_0 \in \mathbb{R}^m$ such that $\mathbf{c} = \mathbf{c}_0 B$ (i.e. computes the $\mathbb{R}^m$-coordinates of $\mathbf{c}$ relative to the basis B), and then rounds off: $\mathbf{v}_0 = \lfloor \mathbf{c}_0 \rceil$. Babai's nearest plane algorithms recursively finds coordinates in nearest hyperplanes. Both methods are deterministic, and find $\mathbf{s} = \mathbf{c} - \mathbf{v}$ in respectively $[-\frac{1}{2}, \frac{1}{2}]^m \cdot B$ and $[-\frac{1}{2}, \frac{1}{2}]^m \cdot B^*$, where B^* is the Gram-Schmidt matrix of B. Both algorithms compromise security when they are applied just like that, as then the 'Learning a parallelepiped' attack by Nguyen and Regev [NR06] applies, leaking the private key after sufficiently many signatures have been generated.
To solve this issue, Falcon uses the Fast Fourier nearest plane algorithm by Ducas and Prest [DP15], which interleaves Babai's nearest plane algorithm with adding random lattice points at each recursion (computed with FFT) and apples rejection sampling, which makes the signature distribution independent of the private basis B, with security proof. This is shown to be efficient, secure and usable with NTRU lattices, but is susceptible to side channel attacks as it uses rejection sampling.
This is the main point where Peregrine deviates from Falcon. It is argued that this application of the Fast Fourier sampler is a major source for the computational complexity of the signature generation. That is why the Peregrine authors propose to replace this by a much simpler trapdoor sampler, that works as follows. One simply takes Babai's round-off algorithm to find v, and then randomizes this by adding a random $\mathbf{a} \in \Lambda^\perp$, so $\mathbf{v}' = \mathbf{v} + \mathbf{a}$ is used instead of $\mathbf{v}$. This $\mathbf{a}$ is sampled from a centered binomial distribution. This design choice makes the rejection sampling disappear so that the implementation becomes a lot less complex and it removes the side channel vulnerability, but also the security proof of Falcon is lost. Nevertheless the submission claims to achieve sufficient (experimentally observed) indepen-

dence for the signature distribution from the private basis B.

16.4 Security of the trapdoor sampler

The Peregrine authors make only two comments on the security of their trapdoor sampler.

> [SKLN22b, Section 2, p. 6, just above Figure 4]: Using the numerical simulation, we can show the uniform distribution of signature enough to hide the information on the secret key B. Thus, Peregrine can be securely used like Crystal-Kyber, an algorithm based on the centered binomial distribution.
> [SKLN22b, Section 5, p. 17, top]: Peregrine currently does not provide the same security proof level as FALCON. However, the centered binomial distribution is widely used in many lattice-based cryptosystems instead of the discrete Gaussian distribution.

This argumentation has been shown to be quite weak by Lin, Suzuki, Zhang, Espitau, Yu, Tibouchi and Abe [LSZ+23], who describe a practical statistical learning key recovery attack, with experimental validation. The point is that the randomization of Peregrine is still dependent on the private basis B. This attack is a generalization of the method of Nguyen and Regev [NR06]. The latter applies to a signature distribution inside one fundamental parallelepiped of the private basis B, which now is replaced by a set of adjacent similar parallelepipeds labeled by α, where the distribution of the α's is independent of B. This leads to studying the Hidden Transformation Problem (HTP), which is the problem of finding good approximations of the rows (up to sign) of the hidden matrix B from a number of independent samples $y = B \cdot x$ where x is sampled from a public distribution.
This Learning a Hidden Transformation is optimized for the case of Peregrine, and is applied to a Peregrine reference implementation for $n = 512$. Moreover, it is shown that only about half of the coefficients of a basis vector are needed, since the other half can then be recovered using a recent technique by Prest using the NTRU equations. This allows for recovering a private key with high probability from only about $25,000$ signatures when parameters are chosen according to the reference implementation, and $11,000,000$ signatures when parameters are chosen according to the specification.
As a result the key recovery parameters in [SKLN22b, Table 3] are untrustworthy, and the submissions' claim that their method will not significantly

damage Falcon's security level is unjustified. It seems that Peregrine's security cannot be fixed without losing its performance advantages over Falcon. It would be of interest to study whether adding rejection sampling to the randomization procedure of Peregrine has better security and performance properties.

16.5 Miscellaneous

Indeed in the KpqC benchmark Peregrine is among the fastest schemes for signature generation and verification, for the vulnerable parameter choices. As remarked in [LSZ+23], the reference implementation has a parameter choice deviating from the specification.
In the implementation we noticed that Peregrine has a heap overflow but was matching KAT files before.

Chapter 17

SOLMAE: quantum-Secure algorithm for Long-term Message Authentication and Encryption

SOLMAE [KTW+22] is a lattice-based signature scheme and may be seen as the result of a sequence of follow-up works to the NIST-selected algorithm FALCON [PFH+22]. More specifically, it follows the ideas presented in MITAKA [EFG+22], to remove the requirement of floating-point arithmetic, as well as new compression techniques presented in [ETWY22].

17.1 System description

The system is based on the hash-then-sign approach in which the message is hashed to some point in the n-dimensional space and the signature is a lattice point close to the hash. Early proposals based on this approach were broken in [NR06] because of the way that the difference of the lattice points from the hashed points reveals the secret basis of the lattice. See also Chapter 16 on Peregrine, a candidate in the KpqC competition falling to the same attack.
FALCON [PFH+22], one of the signature systems selected by NIST for standardization, is based on the same ideas but hides the secret basis using the approach from Gentry, Peikert, and Vaikuntanathan [GPV08]. It uses NTRU lattices and the public key is the quotient of two small polynomials, the latter forming the secret key:
Let $d = 2^n$. KeyGen picks small $f, g \in R = \mathbf{Z}[x]/(x^d + 1)$ and computes $h \equiv g/f \bmod q$ for some prime q, typically chosen so that $2n$ divides $q - 1$.

The public key h defines a lattice with matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix}$$

which defines the same lattice as

$$\mathbf{B} = \begin{pmatrix} f & F \\ g & G \end{pmatrix}$$

for some $F, G \in R$ with $fG - gF = q$. Denote by $L(\mathbf{A}) = L(\mathbf{B})$ the lattice defined by $\mathbf{A}$ and $\mathbf{B}$. Turning the polynomials into $d \times d$ matrices corresponding to the multiplication by that polynomial modulo $x^d + 1$ mean that this is a lattice of dimension $2d$, but for most of this explanation we work with polynomials.
Let $\mathbf{c} = (0, H(r||m))^\top \in R^2$ for some hash function H. Sign uses $\mathbf{B}$ to compute $\mathbf{z} \in L(\mathbf{B})$ and $\mathbf{s}$ with $\mathbf{s} = \mathbf{c} - \mathbf{z}$. Pick new r until $\mathbf{s}$ is short enough so that it does not leak information on $\mathbf{B}$.
The signature is then $(r, \mathbf{s})$.
Verification then checks that $\mathbf{s}$ is small and that $\mathbf{z} = \mathbf{c} - \mathbf{s} \in L(\mathbf{A})$.
The overall scheme is easy to understand, however, FALCON is the only of the NIST-selected schemes for which NIST has not yet posted a draft. One reason is that FALCON is notoriously hard to implement securely and that it is generally hard to implement on small devices that do not feature floating-point units. This is because the sampling of lattice points $\mathbf{z}$ in the signing procedure is defined using floating-point arithmetic. It is possible to emulate floating-point arithmetic using integer arithmetic via so-called fixed-point arithmetic, but high-precision computations get very costly.
MITAKA [EFG+22] is a signature scheme following the same ideas but changing the sampler to the hybrid sampler in Ducas–Prest [DP16]. This has the benefit that the scheme has less system requirements but the downside that the sampled lattice points end up further away from the hash value, which decreases security. The paper puts much effort into improving the sampler and is analysis.
SOLMAE [KTW+22] builds on the same sampler but changes how the secret key is generated, achieving better quality. The main idea in SOLMAE is in KeyGen, defining how f and g are chosen to ensure that $\mathbf{s}$ is short enough and that $\mathbf{z} = \mathbf{c} - \mathbf{s}$ is a lattice point. Computations in $R = \mathbf{Z}[x]/(x^{2^n} + 1)$ use the FFT and the Fast Fourier Sampler from [DP16], used in Falcon, samples in the Fourier domain using complex embeddings and needing floating point numbers. The main idea in SOLMAE is to generate the key as small elements *in the Fourier domain*, ensuring that the sampler would have very good

quality (corresponding to a small value of α, but we do not want to speak of "low" here). KeyGen then checks that also the corresponding polynomials in R (obtained after the inverse FFT and rounding) and their canonical embeddings are small and repeats the choice otherwise. While in general smallness in both domains is related, it is not a strict correspondence, so it is relevant to ensure that both are small. At the expense of more sampling during key generation, SOLMAE achieves as good quality as FALCON at the 128-bit security level and slightly worse at higher levels.

To give more precise definitions we need the concept of canonical embeddings. Let ζ be a primitive $2d$-th root of unity. All odd powers of ζ give different primitive $2d$-th roots of unity. Let φ_j be the embedding $R \to \mathbf{C}, x \mapsto \zeta^{2j-;1}$. Note that $\varphi_{d-j}(f) = \overline{\varphi_j(f)}$, where $^-$ denotes complex conjugation. The canonical embedding of $f \in R$ is $\varphi(f) = (\varphi_1(f), \varphi_2(f), \ldots, \varphi_d(f))$. The lengths of f and $\varphi(f)$ are linked with $||\varphi(f)|| = \sqrt{d}||f||$ (taking the Euclidean norm of the coefficients of f.

The definition of quality depends on the sampling algorithm, for the hybrid sampler

$$Q(f,g)^2 = \max_{1 \le i \le d/2} \max\left(\frac{\varphi_i(f)^2 + \varphi_i(g)^2}{q}, \frac{q}{\varphi_i(f)^2 + \varphi_i(g)^2}\right).$$

The sampling during key generation picks pairs of Fourier terms in polar coordinates by uniformly picking their absolute values within a segment of a 2-dimensional torus and then uniformly selecting the angle in $[0, 2\pi)$. The inverse FFT is applied to get back to a polynomial where the coefficients are rounded to the nearest integers to determine f and g. Of those rounded polynomials the Fourier coefficients are computed anew and the key is accepted if the absolute value of each part is within $[q/\alpha^2, d\alpha^2]$. The target quality for Falcon is $\alpha_{512} = \alpha_{1024} = 1.17$, for SOLMAE it is $\alpha_{512} = 1.17$ and $\alpha_{1024} = 1.64$, A later scheme, Antrag [ENS+23], which is a development of SOLMAE involving many of the same authors, uses $\alpha_{512} = 1.15$ and $\alpha_{1024} = 1.23$.

The smaller Fourier coefficients of f and g then allow to find smaller $\mathbf{s}$ efficiently.

A further improvement in SOLMAE, using [ETWY22], is to shorten the signatures. Let $\mathbf{s} = (s_1, s_2)^\top$ and change the signature to (r, s_1).

Verification then computes $s_2 = H(r||m) + hs_1 \bmod q$ and checks that (s_1, s_2) is short. This uses that $(a, b)^\top \in L(\mathbf{A})$ if $ah - b \equiv 0 \bmod q$, so it reconstructs s_2 so that the vector $\mathbf{z}$ is in the lattice, but unless s_1 was properly constructed, s_2 will not be short.

17.2 Security

The system uses power-of-two cyclotomics and quotient NTRU for the ideal lattices. The scheme is close to FALCON and we have found no security concerns in the changed sampling of the secret key. This leaves generic attacks to determine the dimensions and sizes of the keys.

17.3 Generic lattice attacks

Table 17.1 contains the results of the estimator for the BKZ lattice attacks from Albrecht, Curtis, Deo, Davidson, Player, Postlethwaite, Virdia and Wunderer found in [ACD+18]. For more information see Section 2.1. The input of the code is given in Table 17.2.

Parameter n	512	1024
Type	primal	primal
Q-Core-Sieve	122	248
Q-Core-Sieve + O(1)	138	264
Q-Core-Sieve (min space)	137	278
Q-β-Sieve	131	258
Q-8d-Sieve + O(1)	151	278
Core-Sieve	134	273
Core-Sieve + O(1)	150	289
Core-Sieve (min space)	169	344
β-Sieve	143	283
8d-Sieve + O(1)	164	304
Q-Core-Enum + O(1)	154	396
Lotus	162	447
Core-Enum + O(1)	308	791
8d-Enum (quadratic fit) + O(1)	347	1041

Table 17.1: Estimations for security level found with code by [ACD+18]

17.4 Provable security aspects

The authors of SOLMAE are clear and open about the scheme not being covered by a formal proof. To be precise, SOLMAE follows the GPV framework [GPV08] of hash-then-sign using a preimage-sampleable trapdoor function. While the GPV framework itself has a formal proof of security, the

n	512	1024
Standard deviation	$\sqrt{(12889/2)/512}$	$\sqrt{(12889/2)/1024}$
q	12289	12289
Secret distribution	normal	normal
m	512	1024
Norm f	$\sqrt{(12889/2)}$	$\sqrt{(12889/2)}$
Norm g	$\sqrt{(12889/2)}$	$\sqrt{(12889/2)}$
Claimed security	127	256
Category	1	5
Ring	$x^n - +1$	$x^n + 1$

Table 17.2: Input for the code by [ACD+18]

parameters used by SOLMAE do not meet the requirements of the proof; this is the same situation as for FALCON.

17.5 Overall assessment

Overall, SOLMAE has a solid design history with GPV and FALCON and is easier to implement than FALCON. Compared to MITAKA it offers better security relative to the dimension. The later design Antrag shares this benefit and achieves slightly better quality. In the next round SOLMAE should consider adopting these benefits.

For FALCON, a core problem is that GPV requires a certain product to be uniformly distributed. This is commonly proven via the leftover hash lemma and its variants. However, for the mathematical structure underlying FALCON (i.e., module lattices), no strong enough result is known. Given that SOLMAE is using the same structure, the same issue applies.

In conclusion, it is possible that the additional structure used in FALCON or SOLMAE enables attacks that are not possible when using GPV with unstructured lattices. On the positive side, no such vulnerability has been found for FALCON to this moment, although it was selected as a NIST finalist in 2020, and for standardization in 2022.

Bibliography

[AAB+17a] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Adrien Hauteville, and Gilles Zémor. Ouroboros-R. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[AAB+17b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, and Gilles Zémor. RQC. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[AASA+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the second round of the NIST post-quantum cryptography standardization process. NIST IR 8309, 2020. https://doi.org/10.6028/NIST.IR.8309.

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technol-

ogy, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.

[ABD+17a] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LAKE. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[ABD+17b] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LOCKER. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[ABD+19] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. ROLLO. 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.

[ABD+20] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. Rollo specification. Available at https://pqc-rollo.org/documentation.html, 2020.

[ACD+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18:*

11th International Conference on Security in Communication Networks, volume 11035 of *Lecture Notes in Computer Science*, pages 351–367. Springer, Heidelberg, September 2018.

[ACK+23] Thomas Aulbach, Fabio Campos, Juliane Krämer, Simona Samardjiska, and Marc Stöttinger. Separating oil and vinegar with a single trace side-channel assisted Kipnis-Shamir attack on UOV. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):221–245, 2023.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, Heidelberg, August 2009.

[AD17] Martin R. Albrecht and Amit Deo. Large modulus ring-LWE $\geq$ module-LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 267–296. Springer, Heidelberg, December 2017.

[AGHT18] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A new algorithm for solving the rank syndrome decoding problem. In *ISIT*, pages 2421–2425. IEEE, 2018.

[AMHJ+23] Carlos Aguilar-Melchor, Andreas Hülsing, David Joseph, Christian Majenz, Eyal Ronen, and Dongze Yue. SDitH in the QROM. Cryptology ePrint Archive, Paper 2023/756, 2023. https://eprint.iacr.org/2023/756.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, Heidelberg, April 2015.

[AST23] Thomas Aulbach, Simona Samardjiska, and Monika Trimoska. Practical key-recovery attack on MQ-Sign. Cryptology ePrint Archive, Paper 2023/432, 2023. https://eprint.iacr.org/2023/432.

[BBB+20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 64–93. Springer, Heidelberg, May 2020.

[BBB+23] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. Revisiting algebraic attacks on minrank and on the rank decoding problem. *Designs, Codes and Cryptography*, pages 1–37, 07 2023.

[BBC+20a] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 507–536. Springer, Heidelberg, December 2020.

[BBC+20b] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, Chitchanok Chuengsatiansup, Tanja Lange, Adrian Marotzke, Bo-Yuan Peng, Nicola Tuveri, Christine van Vredendaal, and Bo-Yin Yang. NTRU Prime. 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.

[BBD+19] Shi Bai, Katharina Boudgoust, Dipayan Das, Adeline Roux-Langlois, Weiqiang Wen, and Zhenfei Zhang. Middle-product learning with rounding problem and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 55–81. Springer, Heidelberg, December 2019.

[BBD+23] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and mechanizing the

security proof of fiat-shamir with aborts and dilithium. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 358–389. Springer, Heidelberg, August 2023.

[BBE+19] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2147–2164. ACM Press, November 2019.

[BBLP18] Daniel J. Bernstein, Leon Groot Bruinderink, Tanja Lange, and Lorenz Panny. HILA5 Pindakaas: On the CCA security of lattice-based encryption with error correction. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18: 10th International Conference on Cryptology in Africa*, volume 10831 of *Lecture Notes in Computer Science*, pages 203–216. Springer, Heidelberg, May 2018.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, Heidelberg, May 2000.

[BCC+10] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, Heidelberg, August 2010.

[BCH+23] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):321–365, 2023.

[BCL22] Daniel J. Bernstein, Jolijn Cottaar, and Tanja Lange. Analysis of IPCC. Email to KpqC Bulletin, 2022.

[BCLv17] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime: Reducing attack surface at low cost. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017: 24th Annual International Workshop on Selected Areas in Cryptography*, volume 10719 of *Lecture Notes in Computer Science*, pages 235–260. Springer, Heidelberg, August 2017.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma Algebra System. I. The User Language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[Ber03] Thierry P. Berger. Isometries for rank distance and permutation group of Gabidulin codes. *IEEE Trans. Inf. Theory*, 49(11):3016–3019, 2003.

[Ber22] Daniel J. Bernstein. combinatorial lattice security? Email to KpqC bulletin, 2022.

[Beu22] Ward Beullens. Breaking rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479. Springer, Heidelberg, August 2022.

[BFSS13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic Boolean systems. *Journal of Complexity*, 29(1):53–75, 2013.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47. Springer, Heidelberg, February 2014.

[BHH+19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in*

Computer Science, pages 61–90. Springer, Heidelberg, December 2019.

[BHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 323–345. Springer, Heidelberg, August 2016.

[BP18] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018. https://eprint.iacr.org/2018/526.

[BS20] Nina Bindel and John M. Schanck. Decryption Failure Is More Likely After Success. In *PQCrypto'20*, pages 206–225. Springer, 2020.

[CB13] I. V. Chizhov and M. A. Borodin. The failure of McEliece PKC based on Reed-Muller codes. Cryptology ePrint Archive, Report 2013/287, 2013. https://eprint.iacr.org/2013/287.

[CCD+22] Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Junbum Shin, and Damien Stehlé. HAETAE. Submission to KpqC Round 1, 2022.

[CCH+22] Jung Hee Cheon, Hyeongmin Choe, Dongyeon Hong, Jeongdae Hong, Hyoeun Seong, Junbum Shin, and MinJune Yi. SMAUG: the Key Exchange Algorithm based on Module-LWE and Module-LWR. Submission to KpqC Round 1, 2022.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1825–1842. ACM Press, October / November 2017.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin

Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, Heidelberg, December 2001.

[CHK+20] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. NTT multiplication for NTT-unfriendly rings. Cryptology ePrint Archive, Report 2020/1397, 2020. https://eprint.iacr.org/2020/1397.

[CJLR23] Seongtaek Chee, Kyung Chul Jeong, Nari Lee, and Hansol Ryu. Analysis of Layered ROLLO-I. Email to KpqC Bulletin, 2023.

[CLG09] Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, January 2009.

[CMT23] Alain Couvreur, Rocco Mora, and Jean-Pierre Tillich. A new approach based on quadratic forms to attack the McEliece cryptosystem. Cryptology ePrint Archive, Paper 2023/950, 2023. https://eprint.iacr.org/2023/950.

[COT14] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. Polynomial time attack on wild McEliece over quadratic extensions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 17–39. Springer, Heidelberg, May 2014.

[DB22] Jan-Pieter D'Anvers and Senne Batsleer. Multitarget decryption failure attacks and their application to saber and kyber. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, March 2022.

[DCP+20] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias J. Kannwischer, and Jacques Patarin. Rainbow. 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.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, Heidelberg, August 2013.

[Del78] Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3):226–241, 1978.

[Den03] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, Heidelberg, December 2003.

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of Fiat-Shamir with aborts. Cryptology ePrint Archive, Paper 2023/245, 2023. https://eprint.iacr.org/2023/245.

[DGZ17] Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 18–34. Springer, Heidelberg, 2017.

[DHK+21] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. Faster lattice-based KEMs via a generic fujisaki-okamoto transform using prefix hashing. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 2722–2737. ACM Press, November 2021.

[Din21] Itai Dinur. Improved algorithms for solving polynomial systems over GF(2) by multiple parity-counting. In Dániel Marx, editor, *32nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2550–2564. ACM-SIAM, January 2021.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo

Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. 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.

[DP15] Léo Ducas and Thomas Prest. Fast Fourier orthogonalization. Cryptology ePrint Archive, Report 2015/1014, 2015. https://eprint.iacr.org/2015/1014.

[DP16] Leo Ducas and Thomas Prest. Fast Fourier orthogonalization. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016*, pages 191—-198, 2016.

[DPB17] Javad Doliskani, Geovandro C. C. F. Pereira, and Paulo S. L. M. Barreto. Faster cryptographic hash function from supersingular isogeny graphs. Cryptology ePrint Archive, Report 2017/1202, 2017. https://eprint.iacr.org/2017/1202.

[DRV20] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, May 2020.

[DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer, Heidelberg, June 2005.

[EFG+22] 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, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 222–253. Springer, Heidelberg, May / June 2022.

[EFGT17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1857–1874. ACM Press, October / November 2017.

[EM22] Freja Elbro and Christian Majenz. An algebraic attack against McEliece-like cryptosystems based on BCH codes. Cryptology ePrint Archive, Report 2022/1715, 2022. https://eprint.iacr.org/2022/1715.

[ENS+23] Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. Antrag: Annular NTRU trapdoor generation. IACR Cryptology ePrint Archive 2023/1335, 2023.

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Shorter hash-and-sign lattice-based signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 245–275. Springer, Heidelberg, August 2022.

[FK93] Michael R. Fellows and Neal Koblitz. Kid krypto. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 371–389. Springer, Heidelberg, August 1993.

[Flu16] Scott Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085, 2016. https://eprint.iacr.org/2016/085.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, Heidelberg, March 1999.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[FOPT10] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, Heidelberg, May / June 2010.

[For18] Classic McEliece Comparison Task Force. Classic McEliece vs. NTS-KEM. Posted as https://classic.mceliece.org/nist/vsntskem-20180629.pdf, 2018.

[Gab85] Ernst M. Gabidulin. Theory of codes with maximum rank distance. *Problems of Information Transmission*, 21(1):1–12, 1985.

[Gär23] Joel Gärtner. Concrete security from worst-case to average-case lattice reductions. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023 - 14th International Conference on Cryptology in Africa, Sousse, Tunisia, July 19-21, 2023, Proceedings*, volume 14064 of *Lecture Notes in Computer Science*, pages 344–369. Springer, 2023.

[GBJ+23] Aldo Gunsing, Ritam Bhaumik, Ashwin Jha, Bart Mennink, and Yaobin Shen. Revisiting the indifferentiability of the sum of permutations. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 628–660. Springer, Heidelberg, August 2023.

[GJY19] Qian Guo, Thomas Johansson, and Jing Yang. A novel CCA attack using decryption errors against LAC. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 82–111. Springer, Heidelberg, December 2019.

[GKK+17] Lucky Galvez, Jon-Lark Kim, Myeong Jae Kim, Young-Sik Kim, and Nari Lee. McNie. Technical report, National Institute of Standards and Technology, 2017. available at https:

//csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[GPT91] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and thier applications in cryptology. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 482–489. Springer, Heidelberg, April 1991.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008.

[GRS16] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Transactions on Information Theory*, 62(2):1006–1019, 2016.

[HBD+17] 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, and Peter Schwabe. SPHINCS$^+$. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[HBD+19] 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, and Jean-Philippe Aumasson. SPHINCS$^+$. 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.

[HBD+22] 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.

[Hem89] Farhad Hemmati. Closest coset decoding of |u|u+v| codes. *IEEE J. Sel. Areas Commun.*, 7(6):982–988, 1989.

[HGS99] Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99: 2nd International Conference on Information and Communication Security*, volume 1726 of *Lecture Notes in Computer Science*, pages 2–12. Springer, Heidelberg, November 1999.

[HGSW03] Nick Howgrave-Graham, Joseph Silverman, and William Whyte. A Meet-In-The-Middle Attack on an NTRU Private Key, July 2003. https://ntru.org/f/tr/tr004v2.pdf.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, Heidelberg, November 2017.

[HHM22] Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Failing gracefully: Decryption failures and the fujisaki-okamoto transform. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 414–443. Springer, Heidelberg, December 2022.

[HK22] Andreas Hülsing and Mikhail A. Kudinov. Recovering the tight security proof of SPHINCS+. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, December 2022.

[HLS18] Andreas Hülsing, Tanja Lange, and Kit Smeets. Rounded Gaussians - fast and secure constant-time sampling for lattice-based crypto. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 728–757. Springer, Heidelberg, March 2018.

[How07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, Heidelberg, August 2007.

[HPS98] 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 *Lecture Notes in Computer Science*, pages 267–288. Springer, Heidelberg, June 1998.

[HS01] Jeffrey Hoffstein and Joseph H. Silverman. Optimizations for NTRU. In *Public-key cryptography and computational number theory. Proceedings of the international conference organized by the Stefan Banach International Mathematical Center, Warsaw, Poland, September 11–15, 2000*, pages 77–88. de Gruyter, 2001.

[HTMR15] Anna-Lena Horlemann-Trautmann, Kyle Marshall, and Joachim Rosenthal. Extension of Overbeck's attack for Gabidulin-based cryptosystems. *Designs, Codes and Cryptography*, 86:319–340, 2015.

[IJY23] Yasuhiko Ikematsu, Hyungrok Jo, and Takanori Yasuda. A security analysis on MQ-Sign. Cryptology ePrint Archive, Paper 2023/581, 2023. https://eprint.iacr.org/2023/581.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[Jea16] Jérémy Jean. Cryptanalysis of haraka. Cryptology ePrint Archive, Report 2016/396, 2016. https://eprint.iacr.org/2016/396.

[JKRS21] Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 117–146. Springer, Heidelberg, October 2021.

[JZC+17] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. Cryptology ePrint Archive, Report 2017/1096, 2017. https://eprint.iacr.org/2017/1096.

[JZC+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 96–125. Springer, Heidelberg, August 2018.

[JZM21] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. On the non-tightness of measurement-based reductions for key encapsulation mechanism in the quantum random oracle model. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 487–517. Springer, Heidelberg, December 2021.

[KASA22] Jeongsu Kim Kyung-Ah Shim and Youngjoo An. MQ-Sign: A New Post-Quantum Signature Scheme based on Multivariate Quadratic Equations: Shorter and Faster . Submission to KpqC Round 1, 2022.

[KHL+22a] Jon-Lark Kim, Jihoon Hong, Terry Shue Chien Lau, YounJae Lim, Chik How Tan, Theo Fanuela Prabowo, and Byung-Sun Won. REDOG. Submission to KpqC Round 1, 2022.

[KHL+22b] Jon-Lark Kim, Jihoon Hong, Terry Shue Chien Lau, YounJae Lim, Chik How Tan, Theo Fanuela Prabowo, and Byung-Sun Won. REDOG and its performance analysis. Cryptology ePrint Archive, Report 2022/1663, 2022. https://eprint.iacr.org/2022/1663.

[KHS+22a] Seongkwang Kim, Jincheol Ha, Mincheol Son, Byeonghak Lee, Dukjae Moon, Joohee Lee, Sangyub Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. The AIMer Signature Scheme. Submission to KpqC Round 1, 2022.

[KHS+22b] Seongkwang Kim, Jincheol Ha, Mincheol Son, Byeonghak Lee, Dukjae Moon, Joohee Lee, Sangyup Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. AIM: Symmetric primitive for shorter signatures with stronger security. Cryptology ePrint Archive, Report 2022/1387, 2022. https://eprint.iacr.org/2022/1387.

[KHSL23] Seongkwang Kim, Jincheol Ha, Mincheol Son, and Byeonghak Lee. Mitigation on the AIM cryptanalysis. Cryptology ePrint Archive, Paper 2023/1474, 2023. https://eprint.iacr.org/2023/1474.

[Kim23] Seongkwang Kim. A question for the hardness of the TMO problem (GCKSign). KpqC Bulletin, 23 Feb. 2023, 2023. https://groups.google.com/g/kpqc-bulletin/c/kFNAPdxg_QQ.

[KJKK22] Dong-Chan Kim, Chang-Yeol Jeon, Yeonghyo Kim, and Minji Kim. PALOMA: Binary Separable Goppa-based KEM. Submission to KpqC Round 1, 2022. https://www.kpqc.or.kr/images/pdf/PALOMA.pdf.

[KKGK21] Jon-Lark Kim, Young-Sik Kim, Lucky Erap Galvez, and Myeong Jae Kim. A modified Dual-Ouroboros public-key encryption using Gabidulin codes. *Appl. Algebra Eng. Commun. Comput.*, 32(2):147–156, 2021.

[KKN22] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes. Submission to KpqC Round 1, 2022.

[KKN23a] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Comments and modification on Layered ROLLO on kPQC-forum. Slides attached to reply on KpqC bulletin 19 May 2023, 2023.

[KKN23b] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Comments and modification on Layered ROLLO on kPQC-forum. Slides attached to reply on KpqC Bulletin 22 Septmber 2023, 2023.

[KKN23c] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Comments and modification on Layered ROLLO on kPQC-forum. Slides attached to reply on KpqC bulletin 20 Oct 2023, 2023.

[KKN23d] Chanki Kim, Young-Sik Kim, and Jong-Seon No. New design of blockwise interleaved ideal low-rank parity-check codes for fast post-quantum cryptography. *IEEE Commun. Lett.*, 27(5):1277–1281, 2023.

[KLMR16] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka - efficient short-input hashing for post-quantum applications. Cryptology ePrint Archive, Report 2016/098, 2016. https://eprint.iacr.org/2016/098.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, Heidelberg, April / May 2018.

[KLY22] Suhri Kim, Youngdo Lee, and Kisoon Yoon. FIBS: Fast Isogeny Based Digital Signature. Submission to KpqC Round 1, 2022.

[KP22] Jonghyun Kim and Jong Hwan Park. NTRU+: Compact Construction of NTRU Using Simple Encoding Method. Submission to KpqC Round 1, 2022.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, Heidelberg, May 1999.

[Kra05] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, Heidelberg, August 2005.

[KRL23] Minkyu Kim, Han Sol Ryu, and Ho Chang Lee. Analysis of GCKSign. KpqC Bulletin, 26 Dec. 2022, 2023. https://groups.google.com/g/kpqc-bulletin/c/LnxqwSbyljg.

[KS98] Aviad Kipnis and Adi Shamir. Cryptanalysis of the Oil & Vinegar signature scheme. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Springer, Heidelberg, August 1998.

[KTW+22] Kwangjo Kim, Mehdi Tibouchi, Alexandre Wallet, Thomas Espitau, Akira Takahashi, Yang Yu, and Sylvain Guilley. SOLMAE: quantum-Secure algorithm for Long-term Message Authentication and Encryption. Submission to KpqC Round 1, 2022.

[KZ22] Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Report 2022/588, 2022. https://eprint.iacr.org/2022/588.

[LDK+20] 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, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[LDK+22] 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.

[Lee23a] Joohee Lee. Analysis of NTRU+. Email to KpqC bulletin, 2023.

[Lee23b] Seungwoo Lee. Analysis of TiGER. Email to KpqC Bulletin, 2023.

[LLK23] Joohee Lee, Eunmin Lee, and Jiseung Kim. Combinatorial security estimation reports on TiGER KEM. Email to KpqC Bulletin, 2023.

[LLKN19] Yongwoo Lee, Wijik Lee, Young-Sik Kim, and Jong-Seon No. A modified pqsigRM: RM code-based signature scheme. Cryptol-

ogy ePrint Archive, Report 2019/678, 2019. https://eprint.iacr.org/2019/678.

[LMØM23] Fukang Liu, Mohammad Mahzoun, Morten Øygarden, and Willi Meier. Algebraic attacks on RAIN and AIM using equivalent representations. Cryptology ePrint Archive, Paper 2023/1133, 2023. https://eprint.iacr.org/2023/1133.

[Loi17] Pierre Loidreau. A new rank metric codes based encryption scheme. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 3–17. Springer, Heidelberg, 2017.

[LP23] Tanja Lange and Alex Pellegrii. Email and attachment posted on KpqC bulletin 22 Sep 2023, 2023.

[LPR23] Tanja Lange, Alex Pellegrini, and Alberto Ravagnani. On the security of REDOG. Cryptology ePrint Archive, Paper 2023/1205, 2023. https://eprint.iacr.org/2023/1205.

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In Philip N. Klein, editor, *28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2190–2202. ACM-SIAM, January 2017.

[LS19] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly fast NTRU using NTT. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):180–201, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8293.

[LSW+22] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-the-middle attack on LowMC. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 225–255. Springer, Heidelberg, December 2022.

[LSZ+23] Xiuhan Lin, Moeto Suzuki, Shiduo Zhang, Thomas Espitau, Yang Yu, Mehdi Tibouchi, and Masayuki Abe. Cryptanalysis of the peregrine lattice-based signature scheme. Cryptology ePrint Archive, Paper 2023/1628, 2023. https://eprint.iacr.org/2023/1628.

[LT18] Terry Shue Chien Lau and Chik How Tan. Key recovery attack on McNie based on low rank parity check codes and its reparation. In Atsuo Inomata and Kan Yasuda, editors, *IWSEC 18: 13th International Workshop on Security, Advances in Information and Computer Security*, volume 11049 of *Lecture Notes in Computer Science*, pages 19–34. Springer, Heidelberg, September 2018.

[LTP21] Terry Shue Chien Lau, Chik How Tan, and Theo Fanuela Prabowo. On the security of the modified Dual-Ouroboros PKE using Gabidulin codes. *Appl. Algebra Eng. Commun. Comput.*, 32(6):681–699, 2021.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, Heidelberg, December 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, Heidelberg, April 2012.

[May21] Alexander May. How to meet ternary LWE keys. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 701–731, Virtual Event, August 2021. Springer, Heidelberg.

[Mer23] Jorge Correa Merlino. From Middle-Earth to the Galaxy: SMAUG vs. Kyber. Bachelor's thesis, Eindhoven University of Technology, 2023. https://research.tue.nl/en/studentTheses/from-middle-earth-to-the-galaxy-smaug-vs-kyber.

[MS77] F. Jessie MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Pub. Co., 1977.

[MS07] Lorenz Minder and Amin Shokrollahi. Cryptanalysis of the sidelnikov cryptosystem. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes*

in Computer Science, pages 347–360. Springer, Heidelberg, May 2007.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer, Heidelberg, May / June 2006.

[OK15] Ayoub Otmani and Hervé Talé Kalachi. Square code attack on a modified sidelnikov cryptosystem. In *C2SI*, volume 9084 of *Lecture Notes in Computer Science*, pages 173–183. Springer, 2015.

[Ove05] Raphael Overbeck. A new structural attack for GPT and variants. In *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 2005.

[Ove08] R. Overbeck. Structural attacks for public key cryptosystems based on Gabidulin codes. *Journal of Cryptology*, 21(2):280–301, April 2008.

[Pan19] Lorenz Panny. Isogeny-based hashing despite known endomorphisms. Cryptology ePrint Archive, Report 2019/927, 2019. https://eprint.iacr.org/2019/927.

[Pat75] Nicholas J. Patterson. The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, 21:203–207, 1975.

[Pat97] Jacques Patarin. The oil and vinegar signature scheme. Dagstuhl Workshop on Cryptography, 1997.

[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 Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1843–1855. ACM Press, October / November 2017.

[Pel23a] Alex Pellegrini. Email and attachment posted on KpqC Bulletin 3 Oct 2023, 2023.

[Pel23b] Alex Pellegrini. Email and attachment posted on KpqC Bulletin 22 Oct 2023, 2023.

[Pet13] Albrecht Petzoldt. *Selecting and reducing key sizes for multivariate cryptography*. PhD thesis, Darmstadt University of Technology, Germany, 2013.

[PFH+22] 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.

[PJP+22] Seunghwan Park, Chi-Gon Jung, Aesun Park, Joongeun Choi, and Honggoo Kang. TiGER: Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE(R). Submission to KpqC Round 1, 2022.

[PL17] Christophe Petit and Kristin Lauter. Hard and easy problems for supersingular isogeny graphs. Cryptology ePrint Archive, Report 2017/962, 2017. https://eprint.iacr.org/2017/962.

[PP19] Chris Peikert and Zachary Pepin. Algebraically structured LWE, revisited. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 1–23. Springer, Heidelberg, December 2019.

[Pra62] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[PW23] Aurel Page and Benjamin Wesolowski. The supersingular endomorphism ring and one endomorphism problems are equivalent. Cryptology ePrint Archive, Paper 2023/1399, 2023. https://eprint.iacr.org/2023/1399.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[RKY+22] Jieun Ryu, Yongbhin Kim, Seungtai Yoon, Ju-Sung Kang, and Yongjin Yeom. IPCC – Improved Perfect Code Cryptosystems. Submission to KpqC Round 1, 2022.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):307–335, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8592.

[S+21] W. A. Stein et al. *Sage Mathematics Software (Version (9.3))*. The Sage Development Team, 2021. http://www.sagemath.org.

[Saa17] Markku-Juhani O. Saarinen. HILA5. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[SAB+22] 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.

[SE91] Claus-Peter Schnorr and Michael Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In Lothar Budach, editor, *FCT'91*, volume 529 of *LNCS*, pages 68–85. Springer, 1991.

[SKA22] Kyung-Ah Shim, Jeongsu Kim, and Youngjoo An. NCC-Sign: A New Lattice-based Signature Scheme using Non-Cyclotomic Polynomials. Submission to KpqC Round 1, 2022.

[SKA23] Kyung-Ah Shim, Jeongsu Kim, and Youngjoo An. NCC-Sign: A new lattice-based signature scheme using non-cyclotomic polynomials. https://github.com/jsk-NIMS/NCC-MQ-Sign/blob/main/NCC-Sign-v1.0.pdf, 2023.

[SKLN22a] Eun-Young Seo, Young-Sik Kim, Joon-Woo Lee, and Jong-Seon No. Peregrine. Submission to KpqC Round 1, 2022.

[SKLN22b] Eun-Young Seo, Young-Sik Kim, Joon-Woo Lee, and Jong-Seon No. Peregrine: Toward fastest FALCON based on GPV framework. Cryptology ePrint Archive, Report 2022/1495, 2022. https://eprint.iacr.org/2022/1495.

[SLK22] Kyung-Ah Shim, Sangyub Lee, and Namhun Koo. Efficient implementations of rainbow and UOV using AVX2. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):245–269, 2022.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. On provable security of UOV and HFE signature schemes against chosen-message attack. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 68–82. Springer, Heidelberg, November / December 2011.

[SSL20] Sven Schäge, Jörg Schwenk, and Sebastian Lauer. Privacy-preserving authenticated key exchange and the case of IKEv2. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 567–596. Springer, Heidelberg, May 2020.

[Sta11] Richard P. Stanley. *Enumerative Combinatorics*, volume 1 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2 edition, 2011.

[Ste23] Luc G.A.M. Steenbakkers. NTRU+: Analyzing variants, attacks and security of a lattice-based post-quantum scheme. Bachelor's thesis, Eindhoven University of Technology, 2023. https://research.tue.nl/en/studentTheses/ntru-2.

[STHY23] Kyohei Sudo, Masayuki Tezuka, Keisuke Hara, and Yusuke Yoshida. Quantum search-to-decision reduction for the LWE problem. Cryptology ePrint Archive, Report 2023/344, 2023. https://eprint.iacr.org/2023/344.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum

random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551. Springer, Heidelberg, April / May 2018.

[The22] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5)*, 2022. `https://www.sagemath.org`.

[van16] Christine van Vredendaal. Reduced memory meet-in-the-middle attack against the NTRU private key. Cryptology ePrint Archive, Report 2016/177, 2016. https://eprint.iacr.org/2016/177.

[Ver23] Arend Verbeek. Comparing FIBS to SPHINCS$^+$: An Analysis of Isogeny-Based Signatures. Bachelor's thesis, Eindhoven University of Technology, 2023. https://research.tue.nl/en/studentTheses.

[WLP22] Joo Woo, Kwangsu Lee, and Jong Hwan Park. GCKSign: Simple and efficient signatures from generalized compact knapsacks. Cryptology ePrint Archive, Report 2022/1665, 2022. https://eprint.iacr.org/2022/1665.

[ZWY+23] Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic attacks on round-reduced RAIN and full AIM;-III. Cryptology ePrint Archive, Paper 2023/1397, 2023. https://eprint.iacr.org/2023/1397.