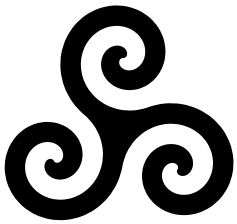
ROLLO - Rank-Ouroboros, LAKE & LOCKER



April 21, 2020

ROLLO is a compilation of three candidates to NIST's competition for post-quantum cryptography standardization. They are based on rank metric codes and they share the same decryption algorithm for LRPC codes (see Sec.1).

Rank-Ouroboros (formerly known as Ouroboros-R) and LAKE are IND-CPA KEM running in the category "post-quantum key exchange". LOCKER is an IND-CCA2 PKE running in the category "post-quantum public key encryption". Different sets of parameters for these three cryptosystems are proposed for security strength categories 1, 3, and 5.

For clarification reasons, we have chosen to uniformize the name of our protocols. In the following document,

- *LAKE is renamed ROLLO-I,*
- *LOCKER is renamed ROLLO-II,*
- *Rank-Ouroboros is renamed ROLLO-III.*
  (not considered anymore in April 2020 update)

Principal Submitters (by alphabetical order):

- Carlos Aguilar Melchor
- Nicolas Aragon
- Magali Bardet
- Slim Bettaieb
- Loïc Bidoux
- Olivier Blazy
- Jean-Christophe Deneuville
- Philippe Gaborit
- Adrien Hauteville
- Ayoub Otmani
- Olivier Ruatta
- Jean-Pierre Tillich
- Gilles Zémor

Inventors: Same as submitters

Developers: Same as submitters

Owners: Same as submitters

The PhD of Nicolas Aragon was partially funded by French DGA.

Main contact

Philippe Gaborit
philippe.gaborit@unilim.fr
+33-626-907-245
University of Limoges
123 avenue Albert Thomas
87 060 Limoges Cedex
France

Backup point of contact

Adrien Hauteville
adrien.hauteville@inria.fr
+33-642-709-282
INRIA Saclay
1 rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing
Campus de l'École Polytechnique
91120 Palaiseau
France

1 History of updates on ROLLO

1.1 Updates for April 22nd, 2020

- For sake of simplicity we now only consider the ROLLO-I and ROLLO-II schemes. These two schemes correspond to two versions of a unique protocol with adapted parameters depending on the expected DFR. ROLLO-I has small public keys when ROLLO-II has larger ones but a smaller DFR.

- New results on algebraic attacks for rank metric [4, 5] now provide a clear and better understanding of algebraic attacks against the rank syndrome decoding problem. As a consequence, we have updated our parameters in order to resist these attacks. The new parameters are presented in section 2.7. Roughly speaking the size of the key has increased by a factor of order 40% and our parameters remain attractive. For instance the size of the public key for ROLLO-I-128 is now 696 Bytes.

- In order to keep small parameters, we modified the decoding algorithm. We now consider in Section 2.4 the original LRPC decoder rather than its optimization. This simpler decoder permits to decode with smaller values of m (the degree of the field extension) than the optimized decoder, and having a smaller m permits to have a better resistance to algebraic attacks of [4, 5].

- We provide an implementation of the decoding algorithm implemented in a constant-time way whenever relevant and which should not leak any sensitive information with respect to timing attacks.

- We provide an optimized implementation leveraging AVX and CLMUL instructions.

- Our implementations no longer rely on third party libraries for finite field arithmetic.

- For 128 bits of security, we obtain the following sizes (in bytes) and performances (in kilocycles) for ROLLO-I:

Public Key	Ciphertext	KeyGen	Encaps	Decaps	$Log_2(DFR)$
696	696	939	113	686	-28

1.2 Updates between Round 1 and Round 2

- The authors of ROLLO are the authors of LAKE, LOCKER and Ouroboros-R, plus Magali Bardet and Ayoub Otmani.
- The names of the submissions have changed. LAKE becomes ROLLO-I, LOCKER becomes ROLLO-II and Ouroboros-R becomes ROLLO-III.
- ROLLO-III uses ideal codes instead of quasi-cyclic codes for Ouroboros-R.
- We have updated the parameters of the schemes such that the weight of the error, which is the most important parameter for the security, increases at each level of security. In practice it leads to a small increase of parameters. Concerning ROLLO-II, we have only kept the sets of parameters with a Decryption Failure Rate (DFR) inferior to 2^{-128}.
- We have reorganized the specifications for clarifications.
- We have added a description of the quantum speed-up in the section Known Attacks.

Contents

Prologue

The public key encryption protocol NTRU [14] was introduced in 1998, the main idea behind the protocol is that the secret key consists in the knowledge of a small Euclidean weight vector, which is used to derive a double circulant matrix. This matrix is then seen as a dual matrix of an associated lattice and a specific decoding algorithm based on the knowledge of this small weight dual matrix is used for decryption.

This idea of having as a trapdoor a small weight dual matrix (with a specific associated decoding algorithm) can naturally be generalized to other metrics. It was done in 2013 with MDPC [17] for Hamming metric and also in 2013 for Rank metric with LRPC codes [8]. These three protocols derive from the same basic main idea, adapted for different metrics, which have different properties in terms of efficiency, size of parameters and security reduction.

The previous schemes have many nice features in terms of size of keys, size of exchanged data and efficiency but suffer from the same weakness: their security do not reduce to a well known problem but rather to a specific problem where a special structure is hidden in the public matrix. Indeed the public matrix is generated by small weight vectors. Although this problem is less specific than hidden structure in the McEliece setting, it remains a potential weakness for these schemes (even if practically, one does not really know how to use this type of structure for strongly more efficient attacks in the more general cases).

Recently a new approach called Ouroboros was presented in [1], this approach permits to benefit from the nice features of the previous schemes, but at the same time has a reduction to decoding random quasi-cyclic codes, rather than a more specific code. Of course this comes at a cost: doubling the size of the ciphertext. ROLLO-I follows the idea of [1] for rank metric. The resulting scheme benefits from the nice features of NTRU-like schemes but has also a reduction to a generic problem, at the cost of doubling the size of the ciphertext, also as all associated decoding algorithm for the NTRU-like family of schemes, there is a decryption failure, but in the case of rank metric this decryption failure is low and perfectly estimated.

This proposal is the fusion of three propositions to standardization for the post-quantum cryptography NIST competition: LAKE, LOCKER and Rank Ouroboros (formerly Ouroboros-R). For uniformity reasons, they have been renamed ROLLO-I, ROLLO-II and ROLLO-III respectively.

In the present for sake of simplicity we chose to concentrate on ROLLO-I and ROLLO-II.

2 Specifications

In the following document, q denotes a power of a prime p. The finite field with q elements is denoted by $\mathbb{F}_q$ and more generally for any positive integer m the finite field with q^m elements is denoted by $\mathbb{F}_{q^m}$. We will frequently view $\mathbb{F}_{q^m}$ as an m-dimensional vector space over $\mathbb{F}_q$.

We use bold lowercase (resp. uppercase) letters to denote vectors (resp. matrices).

Let $P \in \mathbb{F}_q[X]$ a polynomial of degree n. We can identify the vector space $\mathbb{F}_{q^m}^n$ with the ring $\mathbb{F}_{q^m}[X]/\langle P\rangle$, where $\langle P\rangle$ denotes the ideal of $\mathbb{F}_{q^m}[X]$ generated by P.

$$\begin{aligned} \Psi : \quad \mathbb{F}_{q^m}^n \quad &\simeq \mathbb{F}_{q^m}[X]/\langle P\rangle \\ (v_0, \ldots, v_{n-1}) &\mapsto \sum_{i=0}^{n-1} v_i X^i \end{aligned}$$

For $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_{q^m}^n$, we define their product similarly as in $\mathbb{F}_{q^m}[X]/\langle P\rangle$: $\boldsymbol{w} = \boldsymbol{u}\boldsymbol{v} \in \mathbb{F}_{q^m}^n$ is the only vector such that $\Psi(\boldsymbol{w}) = \Psi(\boldsymbol{u})\Psi(\boldsymbol{v})$. In order to lighten the formula, we will omit the symbol Ψ in the future.

To a vector $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$ we can associate an $n \times n$ square matrix corresponding to the product by $\boldsymbol{v}$. Indeed,

$$\begin{aligned} \boldsymbol{u}\boldsymbol{v} &= \boldsymbol{u}(X)\boldsymbol{v}(X) \pmod P \\ &= \sum_{i=0}^{n-1} u_i X^i \boldsymbol{v}(X) \pmod P \\ &= \sum_{i=0}^{n-1} u_i (X^i \boldsymbol{v}(X) \pmod P \\ &= (u_0, \ldots, u_{n-1}) \begin{pmatrix} \boldsymbol{v}(X) & \bmod P \\ X\boldsymbol{v}(X) & \bmod P \\ & \vdots \\ X^{n-1}\boldsymbol{v}(X) & \bmod P \end{pmatrix} \end{aligned}$$

Such a matrix is called the ideal matrix generated by $\boldsymbol{v}$ and P, or simply by $\boldsymbol{v}$ when there is no ambiguity in the choice of P.

Definition 2.0.1 (Ideal Matrix)**.** *Let $P \in \mathbb{F}_q[X]$ a polynomial of degree n and $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$. The ideal matrix generated $\boldsymbol{v}$ is the $n \times n$ square matrix denoted $\mathcal{IM}(\boldsymbol{v})$ of the form:*

$$\mathcal{IM}(\boldsymbol{v}) = \begin{pmatrix} \boldsymbol{v} & \\ X\boldsymbol{v} & \bmod P \\ & \vdots \\ X^{n-1}\boldsymbol{v} & \bmod P \end{pmatrix}$$

As a consequence, the product of two elements of $\mathbb{F}_{q^m}[X]/\langle P\rangle$ is equivalent to the usual product vector-matrix:

$$\boldsymbol{u}\boldsymbol{v} = \boldsymbol{u}\mathcal{IM}(\boldsymbol{v}) = \mathcal{IM}(\boldsymbol{u})^T\boldsymbol{v} = \boldsymbol{v}\boldsymbol{u}.$$

Let S be a finite set. $x \xleftarrow{\$} S$ means that x is an element of S, chosen uniformly at random.

2.1 Presentation of rank metric codes

2.1.1 General definitions

Definition 2.1.1 (Rank metric over $\mathbb{F}_{q^m}^n$)**.** *Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ and $(\beta_1, \ldots, \beta_m) \in \mathbb{F}_{q^m}^m$ a basis of $\mathbb{F}_{q^m}$ viewed as an m-dimensional vector space over $\mathbb{F}_q$. Each coordinate x_j is associated to a vector of $\mathbb{F}_q^m$ in this basis: $x_j = \sum_{i=1}^m x_{ij}\beta_i$. The $m \times n$ matrix associated to $\boldsymbol{x}$ is given by $\boldsymbol{M}(\boldsymbol{x}) = (x_{ij})_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant n}}$.*

The rank weight $\|\boldsymbol{x}\|$ of $\boldsymbol{x}$ is defined as

$$\|\boldsymbol{x}\| \stackrel{def}{=} \operatorname{Rank} \boldsymbol{M}(\boldsymbol{x}).$$

The associated distance $d(\boldsymbol{x}, \boldsymbol{y})$ between elements $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{F}_{q^m}^n$ is defined by $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$.

Definition 2.1.2 ($\mathbb{F}_{q^m}$-linear code)**.** *An $\mathbb{F}_{q^m}$-linear code $\mathcal{C}$ of dimension k and length n is a subspace of dimension k of $\mathbb{F}_{q^m}^n$ embedded with the rank metric. It is denoted $[n, k]_{q^m}$.*

$\mathcal{C}$ can be represented by two equivalent ways:

- *by a generator matrix $\boldsymbol{G} \in \mathbb{F}_{q^m}^{k \times n}$. Each row of $\boldsymbol{G}$ is an element of a basis of $\mathcal{C}$,*

$$\mathcal{C} = \{\boldsymbol{x}\boldsymbol{G}, \boldsymbol{x} \in \mathbb{F}_{q^m}^k\}$$

- *by a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k) \times n}$. Each row of $\boldsymbol{H}$ determines a parity-check equation verified by the elements of $\mathcal{C}$:*

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{0}\}.$$

$\boldsymbol{H}\boldsymbol{v}^T$ is called the syndrome *of $\boldsymbol{v}$ (with respect to $\boldsymbol{H}$).*

We say that $\boldsymbol{G}$ (respectively $\boldsymbol{H}$) is under systematic form if and only if it is of the form $(\boldsymbol{I}_k|\boldsymbol{A})$ (respectively $(\boldsymbol{I}_{n-k}|\boldsymbol{B})$).

Definition 2.1.3 (Support of a word)**.** *Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$. The support E of $\boldsymbol{x}$, denoted* $\operatorname{Supp}(\boldsymbol{x})$*, is the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the coordinates of $\boldsymbol{x}$:*

$$E = \langle x_1, \ldots, x_n \rangle_{\mathbb{F}_q}$$

and we have $\dim E = \|\boldsymbol{x}\|$.

The number of supports of dimension w of $\mathbb{F}_{q^m}$ is denoted by the Gaussian coefficient

$$\begin{bmatrix} m \\ w \end{bmatrix}_q = \prod_{i=0}^{w-1} \frac{q^m - q^i}{q^w - q^i}.$$

2.1.2 Ideal codes

One of the difficulty with code-based cryptography is the size of the key. Indeed, to represent an $[n,k]_{q^m}$ code with a systematic matrix, we need $k(n-k)$ symbols in $\mathbb{F}_{q^m}$, or $k(n-k)\lceil \log q \rceil$ bits. In order to reduce the size of the representation of a code, we introduce the family of ideal codes, which are basically codes with a systematic generator matrix formed with blocks of ideal matrices. More formally,

Definition 2.1.4 (Ideal codes)**.** *Let $P(X) \in \mathbb{F}_q[X]$ be a polynomial of degree n. An $[ns,nt]_{q^m}$ code $\mathcal{C}$ is an (s,t)-ideal code if its generator matrix under systematic form is of the form*

$$\boldsymbol{G} = \begin{pmatrix} & \mathcal{IM}(\boldsymbol{g}_{1,1}) & \dots & \mathcal{IM}(\boldsymbol{g}_{1,s-t}) \\ \boldsymbol{I}_{tn} & \vdots & \ddots & \vdots \\ & \mathcal{IM}(\boldsymbol{g}_{t,1}) & \dots & \mathcal{IM}(\boldsymbol{g}_{t,s-t}) \end{pmatrix}$$

where $\left(\boldsymbol{g}_{i,j}\right)_{\substack{i\in[1..s-t]\\ j\in[1..t]}}$ are vectors of $\mathbb{F}_{q^m}^n$. In this case, we said that $\mathcal{C}$ is generated by the $(\boldsymbol{g}_{i,j})$.

It would be somewhat more natural to choose the generator matrix to be made up of $s \times t$ ideal matrices, rather than to require the code to admit a systematic generator matrix. However, if m and n are two different prime numbers and if P is irreducible, a nonzero ideal matrix is always non-singular. To prove this, we need the following lemma:

Lemma 1. *Let m and n be two different prime numbers. Let $P \in \mathbb{F}_q[X]$ be an irreducible polynomial of degree n and $U \in \mathbb{F}_{q^m}[X]$ a non zero polynomial of degree at most $n-1$. Then P and U are co-prime in $\mathbb{F}_{q^m}[X]$.*

Proof. We will show that P and U have no common root. Let $\mathcal{Z}(P)$ (respectively $\mathcal{Z}(U)$) be the set of the roots of P (respectively U).

Since P is irreducible of degree n, its roots generate $\mathbb{F}_{q^n}$

$$\implies \mathcal{Z}(P) \subset \mathbb{F}_{q^n} \backslash \mathbb{F}_q$$

Since U is of degree at most n, its roots belong to $\mathbb{F}_{q^{m(n-1)!}}$.

But $\mathrm{GCD}(n, m(n-1)!) = 1$ for m and n are two different prime numbers. Thus

$$\mathbb{F}_{q^{m(n-1)!}} \cap \mathbb{F}_{q^n} = \mathbb{F}_q \implies \mathcal{Z}(P) \cap \mathcal{Z}(U) = \emptyset$$

Hence, P and U are co-prime. □

Now, let $\boldsymbol{u} \in \mathbb{F}_{q^m}^n$ a non zero vector and $P \in \mathbb{F}_q[X]$ an irreducible polynomial of degree n. According to the lemma, there exist a vector $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$ such that

$$\begin{aligned} & \boldsymbol{uv} = 1 \pmod{P} \\ \iff & \boldsymbol{u}\mathcal{IM}(\boldsymbol{v}) = (1,0,\dots,0) \\ \iff & \mathcal{IM}(\boldsymbol{u})\mathcal{IM}(\boldsymbol{v}) = \boldsymbol{I}_n \end{aligned}$$

This demonstrates that every block of ideal matrix of $\boldsymbol{G}$ is non-singular, hence $\mathcal{C}$ can be represented under systematic form. □

All the parameters we propose in Section 2.7 verify these conditions.

Remark 2.1. *With this definition, the ideal codes can be seen as a generalization of Quasi-Cyclic codes. Indeed, the generator matrix under systematic form of a Quasi-Cyclic code [6] is of the same form, except that the ideal matrices are replaced by circulant matrices. Yet, an $n \times n$ circulant matrix can be seen as an element of $\mathbb{F}_{q^m}[X]/\langle X^n - 1\rangle$. Thus ideal codes only differ from Quasi-Cyclic codes by the choice of the polynomial P.*

In our scheme, we only use $[ns, n]_{q^m}$ ideal codes. In order to shorten the notation, we denote these codes an s-ideal code. If $\mathcal{C}$ is an $[sn, n]$ ideal code generated by $(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{s-1})$, we have $\mathcal{C} = \{(\boldsymbol{u}, \boldsymbol{u}\boldsymbol{g}_1, \ldots, \boldsymbol{u}\boldsymbol{g}_{s-1}), \boldsymbol{u} \in \mathbb{F}_{q^m}^n\}$.

We need to be careful when we use these notations in the case of parity-check matrix. Indeed, the parity-check matrix under systematic form of $\mathcal{C}$ is of the form:

$$\boldsymbol{H} = \begin{pmatrix} & \mathcal{IM}(\boldsymbol{h}_1)^T \\ \boldsymbol{I}_{n(s-1)} & \vdots \\ & \mathcal{IM}(\boldsymbol{h}_{s-1})^T \end{pmatrix}. \tag{1}$$

Thus, if $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1 \ldots \boldsymbol{\sigma}_{s-1}) \in \mathbb{F}_{q^m}^{s(n-1)}$ is the syndrome of an error $\boldsymbol{e} = (\boldsymbol{e}_1 \ldots \boldsymbol{e}_{s-1}) \in \mathbb{F}_{q^m}^{ns}$, the parity-check equations

$$\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{\sigma}^T$$

are equivalent to $\boldsymbol{e}_i + \boldsymbol{h}_i \boldsymbol{e}_{s-1} = \boldsymbol{\sigma}_i$ for $1 \leqslant i \leqslant s-1$.

2.2 Difficult problems in rank metric

In this section, we describe difficult problems which can be used for cryptography and discuss their hardness. All problems are variants of the *decoding problem*, which consists of looking for the closest codeword to a given vector: when dealing with linear codes, it is readily seen that the decoding problem stays the same when one is given the *syndrome* of the received vector rather than the received vector. We therefore speak of (rank) *Syndrome Decoding* (RSD).

Definition 2.2.1 (RSD Distribution)**.** *For positive integers, n, k, and w, the* $\mathsf{RSD}(n, k, w)$ Distribution *chooses* $\mathbf{H} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times n}$ *and* $\mathbf{x} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ *such that* $\|\mathbf{x}\| = w$*, and outputs* $(\mathbf{H}, \sigma(\mathbf{x}) = \mathbf{H}\mathbf{x}^\top)$.

Definition 2.2.2 (Computational RSD Problem)**.** *On input* $(\mathbf{H}, \mathbf{y}^\top) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{(n-k)}$ *from the* RSD *distribution, the Computational Rank Syndrome Decoding Problem* $\mathsf{RSD}(n, k, w)$ *asks to compute* $\mathbf{x} \in \mathbb{F}_{q^m}^n$ *such that* $\mathbf{H}\mathbf{x}^\top = \mathbf{y}^\top$ *and* $\|\mathbf{x}\| = w$.

The RSD problem has recently been proven difficult with a probabilistic reduction to the Hamming setting in [11]. For cryptography we also need a decision version of the problem, which is given in the following definition.

Definition 2.2.3 (Decision RSD Problem)**.** *On input* $(\mathbf{H}, \mathbf{y}^\top) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{(n-k)}$*, the Decision RSD Problem* DRSD(n, k, w) *asks to decide with non-negligible advantage whether* $(\mathbf{H}, \mathbf{y}^\top)$ *came from the* RSD(n, k, w) *distribution or the uniform distribution over* $\mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{(n-k)}$*.*

As our cryptosystem uses ideal codes, we explicitly define the problem on this setting. The following definitions describe the DRSD problem in the ideal configuration, and are just a combination of Definition 2.1.4 and 2.2.3. Ideal codes are very useful in cryptography since their compact description allows to decrease considerably the size of the keys.

Definition 2.2.4 (s-IRSD Distribution)**.** *Let* $P \in \mathbb{F}_q[X]$ *an irreducible polynomial of degree* n*. For positive integers* n*,* w *and* s*, let* $S(n, s)$ *be the set of the parity-check matrices* $\mathbf{H}$ *under systematic form of* s*-ideal codes of type* $[sn, n]$ *(see Equation 1). The* s-IRSD(n, w) Distribution *chooses uniformly at random a matrix* $\boldsymbol{H} \xleftarrow{\$} S(n, s)$ *together with a vector* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \xleftarrow{\$} \mathbb{F}_{q^m}^{sn}$ *such that* $\|\mathbf{x}\| = w$ *and outputs* $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$*.*

Definition 2.2.5 (Computational s-IRSD Problem)**.** *Let* $P \in \mathbb{F}_q[X]$ *an irreducible polynomial of degree* n*. For positive integers* n*,* w*,* s*, a random parity check matrix* $\mathbf{H}$ *under systematic form of an* s*-ideal code* $\mathcal{C}$ *and* $\mathbf{y} \xleftarrow{\$} \mathbb{F}_{q^m}^{sn-n}$*, the Computational* s*-ideal RSD Problem* s-IRSD(n, w) *asks to compute* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \in \mathbb{F}_{q^m}^{sn}$ *such that* $\|\mathbf{x}\| = w$ *and* $\mathbf{y} = \mathbf{x}\mathbf{H}^\top$*.*

The problem has a decisional form:

Definition 2.2.6 (Decision s-IRSD Problem)**.** *Let* $P \in \mathbb{F}_q[X]$ *an irreducible polynomial of degree* n*. For positive integers* n*,* w *and* s*, let* $S(n, s)$ *be the set of the parity-check matrices* $\mathbf{H}$ *under systematic form of* s*-ideal codes of type* $[sn, n]$ *(see Equation 1). The Decision* s*-Ideal RSD Problem* s-DIRSD(n, w) *asks to decide with non-negligible advantage whether* $(\mathbf{H}, \mathbf{y}^\top)$ *came from the* s-IRSD(n, w) *distribution or the uniform distribution over* $S(n, s) \times \mathbb{F}_{q^m}^{(sn-n)}$*.*

As for the ring-LPN problem, there is no known reduction from the search version of s-IRSD problem to its decision version. The proof of [2] cannot be directly adapted in the ideal case, however the best known attacks on the decision version of the problem s-IRSD remain the direct attacks on the search version of the problem s-IRSD.

Finally we need to introduce a last problem which is useful for the security proof of our schemes. (see Sections 5).

Problem 2.2 (s-Ideal Rank Support Recovery)**.** *Given a vector* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{s-1} \in \mathbb{F}_{q^m}^n$*, a polynomial* $P \in \mathbb{F}_q[X]$ *of degree* n *and a syndrome* $\boldsymbol{\sigma}$ *and a weight* w*, it is hard to recover a support* E *of dimension lower than* w *such that* $\boldsymbol{e}_0 + \boldsymbol{e}_1\boldsymbol{h}_1 + \ldots \boldsymbol{e}_{s-1}\boldsymbol{h}_{s-1} = \boldsymbol{\sigma} \pmod P$ *where the vectors* $\boldsymbol{e}_i$ *are of support* E*.*

Hardness of the problem: We show that the s-IRSR problem and the s-IRSD problem are equivalent.

The s-IRSR problem is trivially reduced to the s-IRSD problem. Indeed to recover the support E of an instance of the s-IRSR problem from a solution $\boldsymbol{x}$ of the s-IRSD problem, we just have to compute the support of $\boldsymbol{x}$.

Reciprocally, the s-IRSD problem can also be reduced to the s-IRSR problem. We prove this property for $s = 2$, the generalization is straightforward. Let us suppose we know the support E of a solution of the 2-IRSR problem for a weight w. We want to find $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1)$ of weight lower than w such that $\boldsymbol{x}_0 + \boldsymbol{x}_1\boldsymbol{h} = \boldsymbol{\sigma} \pmod P$.

This equation is equivalent to

$$\begin{pmatrix} \boldsymbol{I}_n & \mathcal{IM}(\boldsymbol{h})^T \end{pmatrix} (x_{0,0} \dots x_{0,n-1}, x_{1,0} \dots x_{1,n-1})^T = \boldsymbol{\sigma}^T \tag{2}$$

where $\boldsymbol{x}_0 = (x_{1,0} \dots x_{0,n-1})$ and $\boldsymbol{x}_1 = (x_{1,0} \dots x_{1,n-1})$.

Let $(E_1, \dots, E_w)$ be a basis of E. We can express the coordinates of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ in this basis:

$$\forall i \in \{0,1\}, 0 \leqslant j \leqslant n-1, x_{ij} = \sum_{k=1}^{w} \lambda_{ijk} E_k, \text{ with } \lambda_{ijk} \in \mathbb{F}_q$$

Then we rewrite the equations of (2) in the new unknowns λ_{ijk}. We obtain a system of $2nw$ unknowns over $\mathbb{F}_q$ and n equations over $\mathbb{F}_{q^m}$, so nm equations over $\mathbb{F}_q$.

Since E is solution to the 2-IRSR problem, the system has at least one solution and by construction all the solutions have their support included in E of dimension w, so we can find a solution to the 2-IRSD problem by solving this system.

The complexity of known attacks against these problems are described in Section 6.

2.3 The Low Rank Parity Check codes

The LRPC codes have been introduced in [8]. They are good candidates for the cryptosystem of McEliece because the have a weak algebraic structure.

Definition 2.3.1 (LRPC codes)**.** *Let $\boldsymbol{H} = (h_{ij})_{\substack{1 \leqslant i \leqslant n-k \\ 1 \leqslant j \leqslant n}} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ a full-rank matrix such that its coefficients generate an $\mathbb{F}_q$-subspace F of small dimension d:*

$$F = \langle h_{ij} \rangle_{\mathbb{F}_q}$$

Let $\mathcal{C}$ be the code with parity-check matrix $\boldsymbol{H}$. By definition, $\mathcal{C}$ is an $[n,k]_{q^m}$ LRPC code.

Such a matrix $\boldsymbol{H}$ is called homogeneous matrix of weight d and support F.

We can now define the ideal LRPC codes. As we will only use $(2,1)$-ideal LRPC codes in our cryptosystems, we restraint the following definition to this type of codes, but the generalization is straightforward, such as we have done for ideal code.

Definition 2.3.2 (Ideal LRPC codes)**.** *Let F be a $\mathbb{F}_q$-subspace of dimension d of $\mathbb{F}_{q^m}$, $(\boldsymbol{h}_1, \boldsymbol{h}_2)$ two vectors of $\mathbb{F}_{q^m}^n$ of support F and $P \in \mathbb{F}_q[X]$ a polynomial of degree n. Let*

$$\boldsymbol{H} = \begin{pmatrix} \mathcal{IM}(\boldsymbol{h}_1)^T & \mathcal{IM}(\boldsymbol{h}_2)^T \end{pmatrix}$$

By definition, the code $\mathcal{C}$ with parity check matrix $\boldsymbol{H}$ is an ideal LRPC code of type $[2n, n]_{q^m}$.

As we can see, since $P \in \mathbb{F}_q[X]$, the support of $X^i \boldsymbol{h}_1$ is still F for all $1 \leqslant i \leqslant n-1$ hence the necessity to choose P with coefficients in the base field $\mathbb{F}_q$ to keep the LRPC structure of the ideal code.

To hide the structure of an ideal LRPC, we only reveal its systematic parity-check matrix.

Problem 2.3 (Ideal LRPC codes indistinguishability)**.** *Given a polynomial $P \in \mathbb{F}_q[X]$ of degree n and a vector $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$, it is hard to distinguish whether the ideal code $\mathcal{C}$ with the parity-check matrix generated by $\boldsymbol{h}$ and P is a random ideal code or if it is an ideal LRPC code of weight d.*

In other words, it is hard to distinguish if $\boldsymbol{h}$ was sampled uniformly at random or as $\boldsymbol{x}^{-1}\boldsymbol{y} \mod P$ where the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same support of small dimension d.

The ideal LRPC codes are particularly interesting if we choose an irreducible polynomial for P. In this case we counter a structural attack against double circulant LRPC which can be found in [12].

2.4 A support recovery algorithm

Notation 2.4. *Let E be an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension r and F an $\mathbb{F}_q$-subspace of dimension d. We denote by EF the subspace generated the product of the elements of E and F:*

$$EF = \langle \{ef, e \in E, f \in F\} \rangle$$

Let $(e_1, \dots, e_r)$ be basis of E and $(f_1, \dots f_d)$ a basis of F. It is clear that $(e_i f_j)_{\substack{1 \leqslant i \leqslant r \\ 1 \leqslant j \leqslant d}}$ is a generator family of EF. In the typical case $\dim EF = rd$ (see [8], Section 3 for more details on the probability). For the considered parameters, it can happen that $\dim EF < rd$, but this case is also covered by the decoding algorithm without any modification.

Let $\boldsymbol{H} \in \mathbb{F}_{q^m}^{2n \times n}$ be an homogeneous matrix of support F and $\boldsymbol{e} \in \mathbb{F}_{q^m}^{2n}$ an error of support E. Let $\mathcal{C}$ be the LRPC code with parity-check matrix $\boldsymbol{H}$ and $\boldsymbol{s}$ be the syndrome of $\boldsymbol{e}$: $\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s}^T$. In the following section S denotes the support of $\boldsymbol{s}$, it is a subspace of EF so its dimension is at most rd.

S_i is defined by $S_i := f_i^{-1} S$.

2.4.1 Algorithm

The decoding algorithm of LRPC codes first recovers the support of the error vector then solves a linear system in order to recover the error coordinates. For these proposals, we only need to recover the support of the error. The algorithm we present here uses the general decoding algorithm of the LRPC codes described in [8] without the coordinates recovery part.

Algorithm 1: Rank Support Recover (RSR) algorithm

Data: $F = \langle f_1, ..., f_d \rangle$ an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$, $\mathbf{s} = (s_1, \cdots, s_n) \in \mathbb{F}_{q^m}^n$ a syndrome of an error $\boldsymbol{e}$ of weight r and of support E

Result: A candidate for the vector space E

`//Part 1 :` `Compute the vector space` EF

1 Compute $S = \langle s_1, \cdots, s_n \rangle$

`//Part 2 :` `Recover the vector space` E

2 $E \leftarrow \bigcap_{i=1}^{d} f_i^{-1} S$

3 Pre-compute every S_i for $i = 1$ to d

4 **return** E

2.4.2 Probability of failure

First we remark that the previous decoding algorithm works even if the dimension of EF is not maximal, this case happens scarcely and even in this case the intersection of the S_i is still E.

In fact there are only two cases that can make the algorithm fail:

- $\dim S < \dim EF$
- $\dim \bigcap_{i=1}^{d} f_i^{-1} S > r$

We consider in the following the probability of these events.

Proposition 2.4.1. *The probability that* $\dim S < \dim EF$ *is:*

$$q^{-(n-rd+1)}$$

Proof. As in [8] we use the fact that since the error is chosen randomly, every s_i can be seen as a random element of EF, hence the probability that a set of n elements does not generate the whole vector space of dimension rd (with $n > rd$) is given by the probability that a random $rd \times n$ matrix over $\mathbb{F}_q$ is not invertible, i.e q^{n-rd+1}. □

For the second event $\dim \bigcap_{i=1}^{d} f_i^{-1} S > r$, we first prove the following result:

r	r	m	$log_2(failure)$ from 2.4.2	$log_2(failure)$ from simulations
7	8	63	0	-0.50
7	8	64	-7	-7.01
7	8	65	-14	-14.07
7	8	66	-21	-20.91
7	8	67	-28	-28.19

Table 1: Simulation results for proposition 2.4.2

Proposition 2.4.2. *Let V be the space $\mathbb{F}_{q^m}$ and let r and d be two integers. Let E be a fixed subspace of dimension r and let R_i, $1 \leq i \leq d$, be d independently chosen random subspaces of dimension rd containing the subspace E. The probability that $\dim \bigcap_{i=1}^{d} R_i > r$ is bounded from above by:*

$$q^{rd-r} \left(\frac{q^{rd} - q^r}{q^m} \right)^{d-1} \approx q^{-(d-1)(m-rd-r)}.$$

Proof. It suffices to prove the proposition for $r = 0$, since the general result will follow from the case $r = 0$ by applying it to the quotient space V/E. Fix the first subspace R_1 and let $y \in R_1, y \neq 0$. The probability that $y \in R_i$ equals $(q^{rd} - 1)/q^m$, and by independence the probability that this occurs for all $i = 2 \dots d$ is

$$\left(\frac{q^{rd} - q^r}{q^m} \right)^{d-1}. \tag{3}$$

The expected number of non-zero vectors y in the intersection of all the spaces R_i is therefore $q^{rd} - 1$ times the quantity (3), hence the result, since the probability that such a vector y exists is bounded from above by their expected number. □

In practice, in our case the R_i correspond to the S_i and they are not random, but since $S_i = f_i^{-1} S$, we consider the effect of the multiplication by a f_i^{-1} and we do the simplifying assumption that the S_i behave as random spaces in term of intersection. This assumption is remarkably accurate when confronted with simulations as it is shown in Table 1. From here on we consider that the probability that $\dim \bigcap_{i=1}^{d} S_i > r$ can be approximated by $q^{-(d-1)(m-rd-r)}$ which fits very well all the simulations we have made.

From these two propositions we deduce the Decryption Failure Rate of the RSR algorithm:

Proposition 2.4.3. *Under the assumption (validated by simulations) that for intersections the S_i behave as random subspaces containing E, the Decryption Failure Rate of the RSR algorithm 1 can be approximated by:*

$$q^{-(d-1)(m-rd-r)} + q^{-(n-rd+1)}$$

2.5 Presentation of the schemes

In this subsection, $\mathcal{S}_w^n(\mathbb{F}_{q^m})$ stands for the set of vectors of length n and rank weight w over $\mathbb{F}_{q^m}$:

$$\mathcal{S}_w^n(\mathbb{F}_{q^m}) = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \dim \mathrm{Supp}(\boldsymbol{x}) = w\}$$

2.5.1 ROLLO-I as a KEM

A Key-Encapsulation scheme KEM = (KeyGen, Encap, Decap) is a triple of probabilistic algorithms together with a key space $\mathcal{K}$. The key generation algorithm KeyGen generates a pair of public and secret key (pk, sk). The encapsulation algorithm Encap uses the public key pk to produce an encapsulation c, and a key $K \in \mathcal{K}$. Finally Decap using the secret key sk and an encapsulation c, recovers the key $K \in \mathcal{K}$ or fails and return $\perp$.

ROLLO-I is depicted in fig. 1, then formally described in fig. 2. The RSR algorithm was presented in previous section. P is a irreducible polynomial of $\mathbb{F}_q[X]$ of degree n and constitutes a parameter of the cryptosystem.

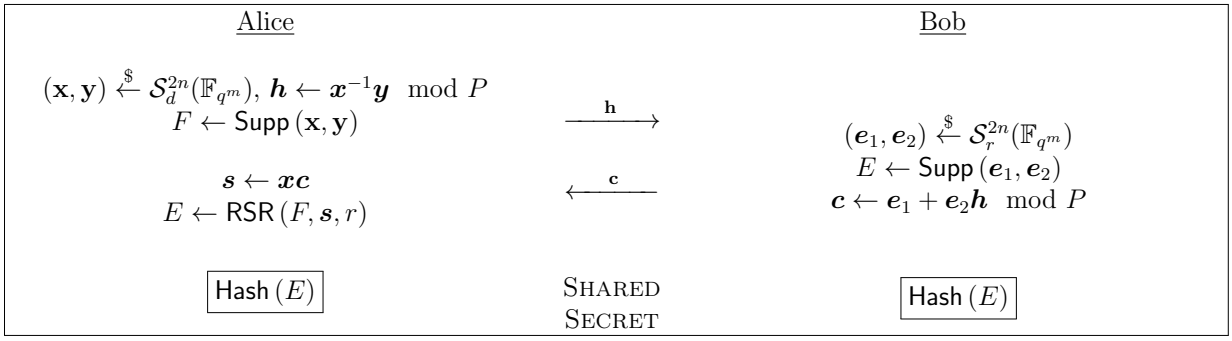


Figure 1: Informal description of ROLLO-I. $\mathbf{h}$ constitutes the public key.

Correctness: Alice recovers $\boldsymbol{s} = \boldsymbol{x}\boldsymbol{c} = \boldsymbol{x}\boldsymbol{e}_1 + \boldsymbol{x}\boldsymbol{e}_2\boldsymbol{h} = \boldsymbol{x}\boldsymbol{e}_1 + \boldsymbol{y}\boldsymbol{e}_2 \mod P$, since $E = \mathsf{Supp}(\boldsymbol{e}_1, \boldsymbol{e}_2)$, $F = \mathsf{Supp}(\mathbf{x}, \mathbf{y})$ and $P \in \mathbb{F}_q[X]$, the coordinates of $\mathbf{s}$ generate a subspace of EF on which Bob can apply the RSR algorithm to recover E.

- KeyGen(1^λ): Picks $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{S}_d^{2n}(\mathbb{F}_{q^m})$. Sets $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y} \mod P$ and return $\mathsf{pk} = \boldsymbol{h}$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- Encap(pk): Picks $(\boldsymbol{e}_1, \boldsymbol{e}_2) \xleftarrow{\$} \mathcal{S}_r^{2n}(\mathbb{F}_{q^m})$, sets $E = \mathsf{Supp}(\boldsymbol{e}_1, \boldsymbol{e}_2), \mathbf{c} = \boldsymbol{e}_1 + \boldsymbol{e}_2\boldsymbol{h} \mod P$. Computes $K = \mathsf{Hash}(E)$ and returns $\mathbf{c}$.
- Decap(sk): Sets $\boldsymbol{s} = \boldsymbol{x}\boldsymbol{c} \mod P$, $F = \mathsf{Supp}(\boldsymbol{x}, \boldsymbol{y})$ and $E \leftarrow \mathsf{RSR}(F, \mathbf{s}, r)$. Recovers $K = \mathsf{Hash}(E)$.

Figure 2: Formal description of ROLLO-I.

Computational costs. The costs are expressed in operations in the base field $\mathbb{F}_q$. The KeyGen cost corresponds to a polynomial modular inversion in $\mathbb{F}_{q^m}[X]/\langle P \rangle$, the Encap and

Decap costs correspond to a polynomial modular addition and a polynomial modular multiplication in $\mathbb{F}_{q^m}[X]/\langle P\rangle$, plus the decoding cost of the RSR algorithm for the decapsulation.

2.5.2 ROLLO-II as a PKE

A Public Key Encryption (PKE) scheme is defined by three algorithms: the key generation algorithm KeyGen which takes on input the security parameter λ and outputs a pair of public and private keys (pk, sk) ; the encryption algorithm $\mathsf{Encrypt}(pk, M)$ which outputs the ciphertext C corresponding to the message M and the decryption algorithm $\mathsf{Decrypt}(sk, C)$ which outputs the plaintext M.

Since ROLLO-II is almost identical to ROLLO-I, we only give its formal description in fig. 3, the correctness and the computational costs are the same. P is a irreducible polynomial of $\mathbb{F}_q[X]$ of degree n and constitutes a parameter of the cryptosystem. The symbol $\oplus$ denotes here the bitwise XOR.

- $\mathsf{KeyGen}(1^\lambda)$: Picks $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{S}_d^{2n}(\mathbb{F}_{q^m})$. Sets $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y} \mod P$ and return $\mathsf{pk} = \boldsymbol{h}$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- $\mathsf{Encrypt}(M, \mathsf{pk})$: Picks $(\boldsymbol{e}_1, \boldsymbol{e}_2) \xleftarrow{\$} \mathcal{S}_r^{2n}(\mathbb{F}_{q^m})$, sets $E = \mathsf{Supp}(\boldsymbol{e}_1, \boldsymbol{e}_2), \mathbf{c} = \boldsymbol{e}_1 + \boldsymbol{e}_2\boldsymbol{h} \mod P$. Computes $cipher = M \oplus \mathsf{Hash}(E)$ and returns the ciphertext $C = (\boldsymbol{c}, cipher)$.
- $\mathsf{Decrypt}(C, \mathsf{sk})$: Sets $\boldsymbol{s} = \boldsymbol{x}\boldsymbol{c} \mod P$, $F = \mathsf{Supp}(\boldsymbol{x}, \boldsymbol{y})$ and $E \leftarrow \mathsf{RSR}(F, \mathbf{s}, r)$. Return $M = cipher \oplus \mathsf{Hash}(E)$.

Figure 3: Formal description of ROLLO-II.

2.6 Representation of objects

Field elements. Elements of $\mathbb{F}_{q^m}$ are represented as vectors of size m over $\mathbb{F}_q$. For ROLLO, q is always chosen equal to 2 (see section 2.7) thus $e \in \mathbb{F}_{q^m}$ is represented as $(e_0, \ldots, e_{m-1}) \in \mathbb{F}_2^m$. In the reference implementation, elements are stored using $8 \times \lceil m/64 \rceil$ bytes in which the unused $64 \times \lceil m/64 \rceil - m$ bits are zero-padded. In the optimized implementation, elements are stored using $16 \times \lceil m/128 \rceil$ bytes in which the unused $128 \times \lceil m/128 \rceil - m$ are zero-padded. The first bit e_0 corresponds to the constant coefficient of the polynomial e.

The polynomials used to construct $\mathbb{F}_{2^m}$ as an extension of $\mathbb{F}_2$ are given table 2.

m	P_m
67	$X^{67} + X^5 + X^2 + X + 1$
79	$X^{79} + X^9 + 1$
83	$X^{83} + X^7 + X^4 + X^2 + 1$
97	$X^{97} + X^6 + 1$

Table 2: Polynomials used to construct $\mathbb{F}_{2^m}$.

Vectors. Elements of $\mathbb{F}_{q^m}^n$ are represented as n-dimensional arrays of $\mathbb{F}_{q^m}$ elements.

Vector spaces. Let E be an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ and $(e_1, \dots e_r) \in \mathbb{F}_{q^m}^r$ a basis of E. We suppose that Alice and Bob have agreed on a basis $(\beta_1, \dots, \beta_m)$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. There exists a matrix $\boldsymbol{M} \in \mathbb{F}_q^{r \times m}$ such that $(e_1, \dots e_r)^T = \boldsymbol{M}(\beta_1, \dots, \beta_m)^T$. In order to have a unique representation, the natural way is to choose the row echelon form of $\boldsymbol{M}$ to represent E (this is equivalent to choose a basis of E). This representation only depends on E. $\boldsymbol{M}$ is then converted into a byte string before being hashed.

Seeds. The considered seed-expander has been provided by the NIST. It is initialized with a byte string of length 40 of which 32 are used as the `seed` and 8 are used as the `diversifier`. In addition, it is initialized with `max_length` equal to $2^{32} - 1$.

2.6.1 Parsing vectors from/to byte strings

Vectors of $\mathbb{F}_{q^m}^n$ are converted to byte strings using a compact representation, in which the unused bits of each element are removed thus leading to a $\left\lceil \frac{nm}{8} \right\rceil$ long byte string.

2.7 Parameters for our schemes

2.7.1 General remarks

In this Section, we propose several sets of parameters for ROLLO-I and ROLLO-II, achieving 128, 192, or 256 bits of security and corresponding therefore to NIST's security strength categories 1, 3, and 5 respectively.

All of our submissions use ideal codes over $\mathbb{F}_{2^m}$ in order to reduce the size of the key and to allow to compute the syndrome of an error as sums and products of polynomials in $\mathbb{F}_{2^m}[X]/\langle P \rangle$, with $P \in \mathbb{F}_2[X]$ of degree n. In order to avoid folding attacks (see [13]), P is chosen irreducible. Moreover, to decrease the computational costs, we want P to be sparse. We have obtained these polynomials with the Magma software. More details are available at http://magma.maths.usyd.edu.au/magma/handbook/text/193#1685.

The best known attacks against our cryptosystems consist in solving an instance of the IRSD problem. The most important parameter for the complexity of algorithms which solve this problem is the weight of the error. That is why we want this parameter to increase at each level of security. Moreover, in order to get homogeneous parameters, we choose the same value for the weight of the error in all sets of parameters, 5 for 128 bits of security, 6 for 192 bits of security and 7 for 256 bits of security.

All the parameters have been chosen so that the best known attack requires at least 2^λ elementary operations for λ bits of security. We refer the reader to Section 6 for more details on best known attacks.

2.7.2 ROLLO-I

Choice of parameters. In section 5.2, the security of the protocol is reduced to the ILRPC problem 2.3 and the IRSR problem 2.2. Our parameters are chosen in function of the best known attacks on these problems described in Section 6. Because of the recent results of [4] and [5] the best attacks for our type of parameters (except for 128 security bits) are now based on algebraic attacks and our parameters are chosen accordingly.

Notice that for 128 security bits parameters, the practical security is rather higher than 128 at about 170 bits. It was possible to have a smaller security parameter closer to 128 by considering smaller r and d, but since there was almost no advantage on the size of the key, we chose to keep the security higher than 128.

The probability of decryption failure (DFR) comes from the probability that the RSR algorithm 1 fails (see Proposition 2.4.3).

Size of parameters. One may use seeds to represent the random data in order to decrease the keysize. We use the NIST seed expander initialized with 40 bytes long seeds.

The public key pk is composed of a vector $\boldsymbol{h} \in \mathbb{F}_{2^m}^n$, so its size is $\lceil \frac{nm}{8} \rceil$ bytes.

The secret key sk is composed of two random vectors of $\mathcal{S}_d^n(\mathbb{F}_{2^m})$, so its size is 40 bytes.

The ciphertext ct is composed of a vector of $\mathbb{F}_{2^m}^n$, so its size is $\lceil \frac{nm}{8} \rceil$ bytes.

The shared secret ss is composed of $K = \mathsf{Hash}(E)$, so its size is 64 bytes (SHA512 output size).

Instance	q	n	m	r	d	P	security	DFR
ROLLO-I-128	2	83	67	7	8	$X^{83} + X^7 + X^4 + X^2 + 1$	128	2^{-28}
ROLLO-I-192	2	97	79	8	8	$X^{97} + X^6 + 1$	192	2^{-34}
ROLLO-I-256	2	113	97	9	9	$X^{113} + X^9 + 1$	256	2^{-33}

Table 3: Parameters for ROLLO-I.

Instance	pk size	sk size	ct size	ss size	Security
ROLLO-I-128	696	40	696	64	128
ROLLO-I-192	958	40	958	64	192
ROLLO-I-256	1371	40	1371	64	256

Table 4: Resulting sizes in bytes for ROLLO-I using NIST seed expander initialized with 40 bytes long seeds. The security is expressed in bits.

2.7.3 ROLLO-II

Choice of parameters. In section 5.3, the security of the protocol is reduced to the ILRPC problem 2.3 and the IRSR problem 2.2. As for ROLLO-I our parameters are chosen in

function of the best known attacks on these problems described in Section 6, also algebraic attacks for this type of parameters.

The probability of decryption failure (DFR) comes from the probability that the RSR algorithm 1 fails (see Proposition 2.4.3).

Size of parameters. One may use seeds to represent the random data in order to decrease the keysize. We use the NIST seed expander initialized with 40 bytes long seeds.

The public key pk is composed of a vector $\boldsymbol{h} \in \mathbb{F}_{2^m}^n$, so its size is $\lceil \frac{nm}{8} \rceil$ bytes.

The secret key sk is composed of two random vectors of $\mathcal{S}_d^n(\mathbb{F}_{2^m})$, so its size is 40 bytes.

The ciphertext ct is composed of a vector of $\mathbb{F}_{2^m}^n$ and a message of 64 bytes masked by random value obtained via an hash. To obtain the IND-CCA2 security, we need to add another hash to the ciphertext (see Section 5.3.2 for more details) so the ciphertext size is $\lceil \frac{nm}{8} \rceil + 2 * 64$ bytes (two hashes).

Instance	q	n	m	r	d	P	security	DFR
ROLLO-II-128	2	189	83	7	8	$X^{189} + X^6 + X^5 + X^2 + 1$	128	2^{-134}
ROLLO-II-192	2	193	97	8	8	$X^{193} + X^{15} + 1$	192	2^{-130}
ROLLO-II-256	2	211	97	8	9	$X^{211} + X^{11} + X^{10} + X^8 + 1$	256	2^{-136}

Table 5: Parameters for ROLLO-II.

Instance	pk size	sk size	ct size	Security
ROLLO-II-128	1941	40	2089	128
ROLLO-II-192	2341	40	2469	192
ROLLO-II-256	2559	40	2687	256

Table 6: Resulting sizes in bytes for ROLLO-II using NIST seed expander initialized with 40 bytes long seeds. The security is expressed in bits.

3 Performances

This section provide performance measures of our implementations.

Benchmark platform. The benchmarks have been performed on a machine that has 16GB of memory and an Intel® Core™ i7-7820X CPU @ 3.6GHz for which the Hyper-Threading, Turbo Boost and SpeedStep features were disabled. The scheme have been compiled with `gcc` (version 9.2.0) and use the `openssl` (version 1.1.1d) library as a provider for SHA2. For each parameter set, the results have been obtained by computing the mean from 1000 random instances. In order to minimize biases from background tasks running on the benchmark platform, each instances have been repeated 100 times and averaged.

3.1 ROLLO-I

Reference Implementation

The performance of our reference implementation on the aforementioned benchmark platform are described in Tab. 7 (thousands of CPU cycles). The following compilation flags have been used: `-O3 -flto`.

Instance	KeyGen	Encap	Decrypt
ROLLO-I-128	3530	392	1062
ROLLO-I-192	4669	458	1284
ROLLO-I-256	6962	591	1992

Table 7: Performances of ROLLO-I reference implementation in thousands of CPU cycles.

Optimized Implementation

An optimized implementation using AVX2 instructions have been provided. Its performances on the aforementioned benchmark platform are described in Tab. 8 (millions of CPU cycles). The following compilation flags have been used: `-O3 -flto -mavx2 - mpclmul -msse4.2 -maes`.

Instance	KeyGen	Encrypt	Decap
ROLLO-I-128	939	113	686
ROLLO-I-192	1142	127	803
ROLLO-I-256	1582	151	1347

Table 8: Performances of ROLLO-I optimized implementation in thousands of CPU cycles.

3.2 ROLLO-II

Reference Implementation

The performance of our reference implementation on the aforementioned benchmark platform are described in Tab. 9 (thousands of CPU cycles). The following compilation flags have been used: `-O3 -flto`.

Optimized Implementation

An optimized implementation using AVX2 instructions have been provided. Its performances on the aforementioned benchmark platform are described in Tab. 10 (thousands of CPU cycles). The following compilation flags have been used: `-O3 -flto -mavx2 -mpclmul -msse4.2 -maes`.

Instance	KeyGen	Encrypt	Decap
ROLLO-II-128	16987	1309	3218
ROLLO-II-192	19153	1441	3684
ROLLO-II-256	22694	1643	4260

Table 9: Performances of ROLLO-II reference implementation in thousands of CPU cycles.

Instance	KeyGen	Encrypt	Decap
ROLLO-II-128	3690	331	1195
ROLLO-II-192	3689	334	1258
ROLLO-II-256	4296	361	1604

Table 10: Performances of ROLLO-II optimized implementation in thousands of CPU cycles.

3.3 Constant time Implementation

We provided a constant time version of the RSR algorithm 1 in the `Additional_Implementations/` folder. This implementation provides the following:

- Constant-time implementation of the Gaussian reduction and of the intersection of vector spaces
- Manipulation of vector spaces of fixed dimension, regardless of the actual dimension of the vector space S

The performances of the reference constant-time version are presented figure 11 for ROLLO-I and 12 for ROLLO-II.

Instance	KeyGen	Encrypt	Decap
ROLLO-I-128	3537	395	1754
ROLLO-I-192	4662	455	2286
ROLLO-I-256	6959	590	3594

Table 11: Performances of ROLLO-I constant time implementation in thousands of CPU cycles.

4 Known Answer Test Values

Known Answer Test (KAT) values have been generated using the script provided by the NIST. They are available in the `KAT/Reference_Implementation/` and `KAT/Optimized_Implementation/` folders.

Instance	KeyGen	Encrypt	Decap
ROLLO-II-128	16986	1307	4484
ROLLO-II-192	19702	1438	5200
ROLLO-II-256	23364	1618	6241

Table 12: Performances of ROLLO-II constant time implementation in thousands of CPU cycles.

Notice that one can generate the aforementioned test files using the `kat` mode of our implementation. The procedure to follow in order to do so is detailed in the technical documentation.

5 Security

5.1 Security Models and Hybrid Argument

IND-CPA. IND-CPA is generally proved through the following game: the adversary $\mathcal{A}$ chooses two plaintexts μ_0 and μ_1 and sends them to the challenger who flips a coin $b \in \{0,1\}$, encrypts μ_b into ciphertext c and returns c to $\mathcal{A}$. The encryption scheme is said to be IND-CPA secure if $\mathcal{A}$ has a negligible advantage in deciding which plaintext c encrypts. This game is formally described hereunder on Fig. 4.

$\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\mu_0, \mu_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk})$
4. $\mathbf{c}^* \leftarrow \mathsf{Encrypt}(\mathsf{pk}, \mu_\mathbf{b}, \theta)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \mathbf{c}^*)$
6. `RETURN` b'

Figure 4: Experiment against the indistinguishability under chosen plaintext attacks

The global advantage for polynomial time adversaries (running in time less than t) is:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(\lambda, t) = \max_{\mathcal{A} \leq t} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda), \tag{4}$$

where $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda)$ is the advantage the adversary $\mathcal{A}$ has in winning game $\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left| \Pr[\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] \right|. \tag{5}$$

Hybrid argument. Alternatively (and equivalently by the hybrid argument), it is possible to construct a sequence of games from a valid encryption of a first message μ_0 to a

valid encryption of another message μ_1 and show that these games are two-by-two indistinguishable. We follow this latter approach and prove the security of our KEM similarly to [1].

5.2 IND-CPA security proof of ROLLO-I

Theorem 5.1. *Under the* Ideal LRPC indistinguishability *2.3 and the* Ideal-Rank Support Recovery *2.2 Problems, the KEM presented earlier in section 2.5.1 is indistinguishable against Chosen Plaintext Attack in the Random Oracle Model.*

Proof. We are going to proceed in a sequence of games. The simulator first starts from the real scheme. First we replace the public key matrix by a random element, and then we use the ROM to solve the Ideal-Rank Support Recovery.

We start from the normal game G_0: We generate the public key $\boldsymbol{h}$ honestly, and $E, \boldsymbol{c}$ also

- In game G_1, we now replace $\boldsymbol{h}$ by a random vector, the rest is identical to the previous game. From an adversary point of view, the only difference is the distribution on $\boldsymbol{h}$, which is either generated at random, or as a product of low weight vectors. This is exactly the *Ideal LRPC indistinguishability* problem, hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_0} \leq \mathsf{Adv}_{\mathcal{A}}^{G_1} + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ILRPC}}$$

- In game G_2, we now proceed as earlier except we receive $\boldsymbol{h}, \boldsymbol{c}$ from a Support Recovery challenger. After sending $\boldsymbol{c}$ to the adversary, we monitor the adversary queries to the Random Oracle, and pick a random one that we forward as our simulator answer to the Ideal-Rank Support Recovery problem. Either the adversary was able to predict the random oracle output, or with probably $1/q_G$, we picked the query associated with the support E (by q_G we denote the number of queries to the random oracle G), hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq 2^{-\lambda} + 1/q_G \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{IRSR}}$$

  which leads to the conclusion.

□

5.3 IND-CCA2 security proof of ROLLO-II

5.3.1 IND-CPA security proof of the ROLLO-II PKE

Theorem 5.2. *Under the* Ideal LRPC indistinguishability *2.3 and the* Ideal-Rank Support Recovery *2.2 Problems, the encryption scheme presented earlier in section 2.5.2 in indistinguishable against Chosen Plaintext Attack in the Random Oracle Model.*

Proof. We are going to proceed in a sequence of games. The simulator first starts from the real scheme. First we replace the public key matrix by a random element, and then we use the ROM to solve the QC-Rank Support Recovery.

We start from the normal game G_0: We generate the public key $\boldsymbol{h}$ honestly, and $E, \boldsymbol{c}$ also

- In game G_1, we now replace $\boldsymbol{h}$ by a random vector, the rest is identical to the previous game. From an adversary point of view, the only difference is the distribution on $\boldsymbol{h}$, which is either generated at random, or as a product of low weight vectors. This is exactly the *Ideal LRPC indistinguishability* problem, hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_0} \leq \mathsf{Adv}_{\mathcal{A}}^{G_1} + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ILRPC}}$$

- In game G_2, we now proceed as earlier except we replace $G(E)$ by random. It can be shown, that by monitoring the call to the ROM, the difference between this game and the previous one can be reduced to the QC-Rank Support Recovery problem, so that:

$$\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq 2^{-\lambda} + q_G \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{IRSR}}.$$

- In a final game G_3 we replace $\boldsymbol{d} = M \oplus \mathsf{Rand}$ by just $\boldsymbol{d} = \mathsf{Rand}$, which leads to the conclusion.

□

5.3.2 A IND-CCA2 conversion of the ROLLO-II PKE

Let $\mathcal{E}$ be an instance of the ROLLO-II cryptosystem as described above. Let $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{K}$. The KEM-DEM version of the ROLLO-II cryptosystem is defined as follows (following [15]) :

When applying the HHK [15] framework for the Fujisaki-Okamoto transformation, one can show that the final transformation is CCA-2 secure such that:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CCA}-2} \leq q_G \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{2q_G + 1}{|\mathcal{M}|} + 3\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CPA}}$$

As our scheme is CPA secure, the last term is negligible, we can handle exponentially large message space for a polynomial number of query, so the previous is too.

As shown before, our scheme is gamma-spread so again for a polynomial number of verification query, the term in q_V is negligible.

The tricky term remaining is $q_G \cdot \delta$, this is the product of the number of queries to the random oracle, by the probability of generating an decipherable ciphertext in an honest execution. For real application, we want schemes to be correct enough so that the probability of such occurrence is very small. This often leads, in application in running with a probability of a magnitude of 2^{-64}. This may seem not low enough for pure cryptographic security, however it should be noted this number, corresponds to the number of request,

- $\mathsf{Setup}(1^\lambda)$: as before, except that k will be the length of the symmetric key being exchanged, typically $k = 256$.
- KeyGen (param): exactly as before.
- Encap (pk): generate $\mathbf{m} \xleftarrow{\$} \mathbb{F}^k$ (this will serve as a seed to derive the shared key). Derive the randomness $\theta \leftarrow \mathcal{G}(\mathbf{m})$. Generate the ciphertext $c \leftarrow (\mathbf{u}, \mathbf{v}) = \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m}, \theta)$, and derive the symmetric key $K \leftarrow \mathcal{K}(\mathbf{m}, \mathbf{c})$. Let $\mathbf{d} \leftarrow \mathcal{H}(\mathbf{m})$, and send $(\mathbf{c}, \mathbf{d})$.
- Decap $(\mathsf{sk}, \mathbf{c}, \mathbf{d})$: Decrypt $\mathbf{m}' \leftarrow \mathcal{E}.\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$, compute $\theta' \leftarrow \mathcal{G}(\mathbf{m}')$, and (re-)encrypt $\mathbf{m}'$ to get $\mathbf{c}' \leftarrow \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m}', \theta')$. If $\mathbf{c} \neq \mathbf{c}'$ or $\mathbf{d} \neq \mathcal{H}(\mathbf{m}')$ then abort. Otherwise, derive the shared key $K \leftarrow \mathcal{K}(\mathbf{m}, \mathbf{c})$.

Figure 5: Description of our proposal ROLLO-II.KEM.

adversarially generated where the simulator gives an honest answer to a decryption query, which would mean that a single user would be able to do as many queries as expected by the whole targeted users in a live application, so a little trade-off at this level seems more than fair.

Security concerns and implementation details. Notice that while NIST only recommends SHA512 as a hash function, the transformation of [15] would be dangerous – at least in our setting – if one sets $\mathcal{G} = \mathcal{H}$. Indeed, publishing the randomness $\theta = \mathcal{G}(\mathbf{m}) = \mathcal{H}(\mathbf{m}) = \mathbf{d}$ used to generate $\mathbf{e}_1$ and $\mathbf{e}_2$, would allow one to retrieve the secret E. We therefore suggest to use SHA3-512 for $\mathcal{G}$ and SHA512 for $\mathcal{H}$.

6 Known Attacks

In this section, we present the best known attacks against the IRSR 2.2, ILRPC 2.3 and DIRSD 2.2.6 problems on which our schemes are based.

Both the ILRPC and DIRSD problems are decision problems. However, at the current state-of-the-art, the best attacks consist in solving a search problem: finding a codeword of a small weight in an ideal LRPC code for the ILRPC and solving an instance of the IRSD problem for the DIRSD problem.

There exist two types of attacks on these problems:

- the combinatorial attacks where the goal is to find the support of the error or of the codeword.
- the algebraic attacks where the opponent tries to solve an algebraic system by Gröbner basis.

First, we deal with the combinatorial attacks for the IRSD and the ILRPC problems and in a third subsection we discuss about the algebraic attacks.

6.1 Attack on the IRSD problem

For an $[sn, n]$ ideal code over $\mathbb{F}_{q^m}$ the best combinatorial attack to solve the IRSD problem 2.2.6 with an error of weight r is in:

$$\mathcal{O}\left(((s-1)nm)^{\omega} q^{r\left\lceil \frac{m(n+1)}{sn} \right\rceil - m}\right)$$

operations in $\mathbb{F}_q$. ω is the exponent of the complexity of the solution of a linear system.

This attack is an improvement of a previous attack described in [9], a detailed description of the attack can be found in [3]. The general idea of the attack is to adapt the Information Set Decoding attack for the Hamming distance. For the rank metric, the attacker tries and guesses a subspace which contains the support of the error and then solves a linear system obtained from the parity-check equations to check if the choice was correct.

The complexity of the best attack against IRSR problem is the same since there is no known way to compute the support of an error without first computing this error.

This attack is a generic attack against the RSD problem, there is no known improvement which exploit the ideal structure of the code.

Remark 6.1. *Since the linear system is not random, it is reasonable to take $\omega = 2$ for the choice of the parameters of ROLLO-I and ROLLO-II, even if the attack described in [3] takes $\omega = 3$.*

Let us remark that the choice of our parameter is flexible. We could take $\omega = 0$ and increase the parameters, which corresponds to only keeping the exponential complexity of the attack, for instance by slightly increasing m.

6.2 Structural attack on ideal LRPC codes

Let $\mathcal{C}$ be an $[2n, n]_{q^m}$ ideal LRPC code generated by the two polynomials $(\boldsymbol{x}, \boldsymbol{y})$ of support F of dimension d. Let $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y}$ which generates the systematic parity-check matrix of $\mathcal{C}$. The problem is to recover the structure of $\mathcal{C}$, given only access to $\boldsymbol{h}$.

The most efficient known attack is to find a codeword of weight d in an $[2n - \left\lfloor \frac{n}{d} \right\rfloor, n - \left\lfloor \frac{n}{d} \right\rfloor]_{q^m}$ subcode $\mathcal{C}'$ of the dual code $\mathcal{C}^{\perp}$ generated by $\boldsymbol{h}$, as described in [10]. The best algorithm is the same decoding algorithm used in the previous subsection [3]. Its complexity is in:

$$\mathcal{O}\left((nm)^{\omega} q^{d\left\lceil \frac{\left(n - \left\lfloor \frac{n}{d} \right\rfloor\right)m}{2n - \left\lfloor \frac{n}{d} \right\rfloor} \right\rceil - m}\right)$$

However, the dual of an ideal LRPC code contains much more codeword of weight d than a random code with the same parameters. Indeed, let $\boldsymbol{H}$ be the matrix of size $n \times 2n$ generated by $(\boldsymbol{x}, \boldsymbol{y})$. By definition, $\boldsymbol{H}$ is a generator matrix of $\mathcal{C}^{\perp}$. Let $(\boldsymbol{h}_i)_{1 \leqslant i \leqslant n}$ be the rows of $\boldsymbol{H}$. For all $1 \leqslant i \leqslant n, \mathrm{Supp}(\boldsymbol{h}_i) = F \implies \mathcal{C}^{\perp}$ contains q^n codewords of the same support. Thus, we have considered an attack in

$$\mathcal{O}\left((nm)^{\omega} q^{d\left\lceil \frac{m}{2} \right\rceil - m - n}\right)$$

for the choice of the parameters of ROLLO-I and ROLLO-II.

There exists a specific attack on the ideal LRPC codes which can be found in [13]. In this article, the authors present an attack against double circulant LRPC codes but it can be adapted straightforwardly in the case of ideal LRPC codes. However, the crucial point of this attack is that the polynomial $X^n - 1$ has always $X - 1$ as divisor and may have many more factors depending on n and q. In the case of ideal LRPC codes, we can choose an irreducible polynomial P of degree n of $\mathbb{F}_q[X]$ to generate the quotient-ring $\mathbb{F}_q[X]/\langle P\rangle$, which completely negates this specific attack.

6.3 Algebraic attacks

For a long time algebraic attacks against the RSD problem were thought to be harmless, two recent papers [4] and [5] have permitted to have a clear and better view of the situation of algebraic attacks for rank metric. These results show that for our cryptographic when $r = O(\sqrt{n})$ they are in fact the best attacks. We sum up in the following these results.

The purpose of algebraic attacks is to consider an RSD instance and to write it as a system of equations, called a *modeling*, then if one solves this system, that is to say finds one solution, it is a solution to the RSD instance.

Ourivski and Johansson pioneered the algebraic attack against RSD by giving a modeling in [18], then Levy and Perret first proposed to solve it using Gröbner basis computations in [16]. In [4], a new modeling was proposed, it is based on maximal minors taken from a slightly different version of Ourivksi and Johansson's modeling.

From now on, in this section, we deal with RSD instances based on $[n, k]$-code over $\mathbb{F}_{2^m}$ with target rank weight r.

The modeling uses the fact that the vector $\mathbf{e}$ of small weight r can be written as a product $\boldsymbol{SC}$ where $\boldsymbol{S}$ is a matrix containing a basis of the support of $\mathbf{e}$ and $\boldsymbol{C}$ is a matrix containing the coordinates of each component of $\mathbf{e}$ in this basis. Those matrices are called the *support* and the *coordinate* matrices.

Roughly, the system will consist in $m\binom{n-k-1}{r}$ equations in $\binom{n}{r}$ variables which are maximal determinants, in [4], this system was then solved by computing its Gröbner basis; in [5], a different modeling enables one to solve it directly by linearization. More precisely, the condition

$$m\binom{n-k-1}{r} \geq \binom{n}{r} \tag{6}$$

is inherent to this system; when it is fulfilled, it corresponds to *the overdetermined case*, if it is not, it corresponds to *the underdetermined case*.

Overdetermined case. If the condition (6) is fulfilled, solving the RSD problem is equivalent to solving a linear system with $m\binom{n-k-1}{r}$ equations in $\binom{n}{r}$ variables, which can be done with a cost in

$$\mathcal{O}\left(m\binom{n-k-1}{r}\binom{n}{r}^{\omega-1}\right). \tag{7}$$

On the one hand, if the condition (6) is *widely* fulfilled, there is an optimization in [5] to reduce the complexity of (7) using a punctured version of the code. On the other hand, if (6) is not fulfilled, one can reduce to it by guessing few variables at an exponential cost, it is an hybrid attack, also in [5].

Underdetermined case. For cryptographic purpose, the parameters are chosen so that they belong to an area where the condition (6) is obviously not fulfilled and where the exponential cost of the hybrid attack would make it impractical, this particular area is called the *underdetermined case*. The best known complexity in this case is also described in [5]: it uses a variation of the aforementioned system together with a new setting coming from the reduction between RSD and the MinRank problem. This new system, despite being bigger, is sometimes sparser, so its resolution could take advantage of Wiedemann algorithm to solve sparse linear systems.

More precisely, the complexity in the underdetermined case is

$$\mathcal{O}\left((B_b + C_b)A_b^{\omega-1}\right) \tag{8}$$

where

$$\begin{aligned}
A_b &:= \sum_{j=1}^{b} \binom{n}{r}\binom{mk+1}{j} \\
B_b &:= \sum_{j=1}^{b} \left(m\binom{n-k-1}{r}\binom{mk+1}{j}\right) \\
C_b &:= \sum_{j=1}^{b}\sum_{i=1}^{j} \left((-1)^{i+1}\binom{n}{r+i}\binom{m+i-1}{i}\binom{mk+1}{j-i}\right).
\end{aligned}$$

and where b is the smallest positive integer so that the condition $A_b - 1 \leq B_c + C_b$ is fulfilled.

When $b \geq 2$, one uses Wiedemann algorithm, resulting in a new complexity of

$$\mathcal{O}\left(\frac{B_b\binom{k+r+1}{r} + C_b(mk+1)(r+1)}{B_b + C_b}\left(\sum_{j=1}^{b}\binom{n}{r}\binom{mk+1}{j}\right)^2\right). \tag{9}$$

6.4 Quantum speed-up

For computational attacks, the quantum speed-up is easy to analyze. According to [7], a slight generalization of Grover's quantum search algorithm allows to divide by a factor 2 the exponential complexity of the attacks. Thus the complexity of the quantum computational attack is

- $\mathcal{O}\left(((s-1)nm)^{\omega} q^{\frac{1}{2}(r\left\lceil\frac{m(n+1)}{sn}\right\rceil - m)}\right)$ for the attack on the $s-$ IRSD problem.
- $\mathcal{O}\left((nm)^{\omega} q^{\frac{1}{2}(d\left\lceil\frac{m}{2}\right\rceil - m - n)}\right)$ for the attack on ideal LRPC codes.

- at the best of our knowledge for algebraic attacks there is not any real quantum speed up

7 Advantages and Limitations

7.1 Strengths

The proposed schemes are very efficient, both in terms of size of keys and computational complexity. They also benefit from a constant time decoding algorithm and its failure probability is very well studied and estimated and can easily be chosen to meet security standards. Moreover, the choice of parameters is very versatile. For ROLLO-I and ROLLO-II, there is a reduction to a well understood generic problem IRSD, which is a natural generalization of Quasi-Cyclic RSD problem. This type of problems has been used for many years for Hamming and Euclidean distances.

7.2 Limitations

Rank metric has very nice features, but the use of rank metric for cryptographic purposes is not very old (1991). It may seems as a limitation, but still in recent years there have been a lot of activities on understanding the inherent computational difficulty of the related problems. The combinatorial attacks are very well understood, and the recent results of [4, 5] permit to have a clear view on the complexity of algebraic attacks.

Like for cryptosystems à la McEliece, ROLLO-I and ROLLO-II security proofs rely on the hardness of retrieving the structure of a structured code, in our case the ideal LRPC codes. However, this problem has been also studied for Hamming and euclidean metrics and is considered hard by the community (for instance, MDPC and NTRU-like cryptosystems are based on it).

References

[1] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *CoRR*, abs/1612.05572, 2016.

[2] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. pages 92–110.

[3] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A new algorithm for solving the rank syndrome decoding problem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, Vail, USA, 2018. IEEE.

[4] 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 *Advances in Cryptology - EUROCRYPT 2020 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020. Proceedings*, 2020. To appear, preprint available on https://arxiv.org/pdf/1910.00810.pdf.

[5] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Algebraic attacks for solving the rank decoding and minrank problems without Gröbner basis, 2020. Preprint available on https://arxiv.org/pdf/2002.08322.pdf.

[6] Philippe Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.

[7] Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. Ranksynd a PRNG based on rank metric. In *Post-Quantum Cryptography 2016*, pages 18–28, Fukuoka, Japan, February 2016.

[8] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC'2013*, Bergen, Norway, 2013. Available on www.selmer.uib.no/WCC2013/pdfs/Gaborit.pdf.

[9] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Trans. Information Theory*, 62(2):1006–1019, 2016.

[10] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. New results for rank-based cryptography. In *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 1–12, 2014.

[11] Philippe Gaborit and Gilles Zémor. On the hardness of the decoding and the minimum distance problems for rank codes. *IEEE Trans. Inform. Theory*, 62(12):7245–7252, 2016. https://arxiv.org/pdf/1404.3482.pdf.

[12] Adrien Hauteville and Jean-Pierre Tillich. New algorithms for decoding in the rank metric and an attack on the lrpc cryptosystem. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2747–2751. IEEE, 2015.

[13] Adrien Hauteville and Jean-Pierre Tillich. New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2015*, pages 2747–2751, Hong Kong, China, June 2015.

[14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998.

[15] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.

[16] Françoise Lévy-dit Vehel and Ludovic Perret. Algebraic decoding of codes in rank metric. In *proceedings of YACC06*, Porquerolles, France, June 2006. available on http://grim.univ-tln.fr/YACC06/abstracts-yacc06.pdf.

[17] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.

[18] Alexei V. Ourivski and Thomas Johansson. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38(3):237–246, 2002.