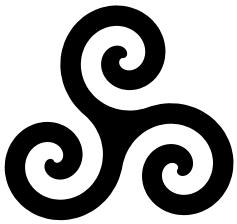
ROLLO - Rank-Ouroboros, LAKE & LOCKER



August 24, 2019

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.*

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

This work 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

Contents

Prologue

The public key encryption protocol NTRU [15] 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 [19] for Hamming metric and also in 2013 for Rank metric with LRPC codes [9]. 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.

1 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,

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

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 1.0.1 (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:*

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

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.

1.1 Presentation of rank metric codes

1.1.1 General definitions

Definition 1.1.1 (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 1.1.2 ($\mathbb{F}_{q^m}$-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 1.1.3 (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

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

1.1.2 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 1.1.4 (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 nonsingular. 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 coprime 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 coprime. □

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 nonsingular, hence $\mathcal{C}$ can be represented under systematic form. □

All the parameters we propose in Section 1.7 verify these conditions.

Remark 1.1. *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 [7] 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 $1 \leqslant i \leqslant s-1$.

1.2 Difficult problems in rank metric

In this section, we introduce the difficult problems on which our cryptosystem is based.

Problem 1.2 (Rank Syndrome Decoding)**.** *Given a full-rank matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$, a syndrome $\boldsymbol{\sigma}$ and a weight w, the RSD problem consists in sampling a vector $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ of weight lower than w such that $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{\sigma}^T$.*

For the security proof we also need a decision version of the problem, which is given in the following definition.

Problem 1.3 (Decision Rank Syndrome Decoding)**.** *Given a full-rank matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ and an integer w, the DRSD problem consists in deciding if a vector $\boldsymbol{\sigma} \in \mathbb{F}_{q^m}^{n-k}$ is sampled according to the uniform distribution over $\mathbb{F}_{q^m}^{n-k}$ or the uniform distribution over the set of syndromes of errors of weight w, i.e.*

$$\boldsymbol{\sigma} \xleftarrow{\$} \mathbb{F}_{q^m}^{n-k} \text{ or } \boldsymbol{\sigma} \xleftarrow{\$} \{\boldsymbol{y} \in \mathbb{F}_{q^m}^{n-k} : \exists \boldsymbol{x} \in \mathbb{F}_{q^m}^n, \|\boldsymbol{x}\| = w, \boldsymbol{y} = \boldsymbol{x}\boldsymbol{H}^T\}$$

Hardness of the problems: The RSD problem has recently been proven hard in [12] with a probabilistic reduction to the well-known NP-Hard syndrome decoding problem in Hamming metric. For the average case, all the best known algorithm are exponential and the problem is considered difficult by the cryptography community. In [8] the DRSD problem has been proven as hard as the same Decision problem in the Hamming metric. This last problem has been proven hard in [6].

Since our cryptosystems use ideal code, we explicitly define the problems on which they are based. These problems are simply a particular case of the RSD problem where the matrix $\boldsymbol{H}$ is a parity-check matrix of an s-ideal code.

Problem 1.4 (s-Ideal Rank Syndrome Decoding)**.** *Given a polynomial $P \in \mathbb{F}_q[X]$ of degree n, $s-1$ vectors $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{s-1} \in \mathbb{F}_{q^m}^n$ generating the systematic parity-check matrix $\boldsymbol{H}$ of an $[ns, n]_{q^m}$ ideal code (see Equation 1), a syndrome $\boldsymbol{\sigma}$ and a weight w, the s-IRSD problem consists in sampling a vector $\boldsymbol{x} = (\boldsymbol{x}_0, \ldots \boldsymbol{x}_{s-1}) \in \mathbb{F}_{q^m}^{sn}$ of weight lower than w such that $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{\sigma}^T$.*

Problem 1.5 (s-Decision Ideal Rank Syndrome Decoding)**.** *Given a polynomial $P \in \mathbb{F}_q[X]$ of degree n, $s-1$ vectors $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{s-1} \in \mathbb{F}_{q^m}^n$ generating the systematic parity-check matrix $\boldsymbol{H}$ of an $[ns, n]_{q^m}$ ideal code and an integer w, the s-DIRSD problem consists in deciding if a vector $\boldsymbol{\sigma} \in \mathbb{F}_{q^m}^{n(s-1)}$ is sampled according to the uniform distribution over $\mathbb{F}_{q^m}^{n(s-1)}$ or the uniform distribution over the set of syndromes of errors of weight w, i.e.*

$$\boldsymbol{\sigma} \xleftarrow{\$} \mathbb{F}_{q^m}^{n(s-1)} \text{ or } \boldsymbol{\sigma} \xleftarrow{\$} \{\boldsymbol{y} \in \mathbb{F}_{q^m}^{n(s-1)} : \exists \boldsymbol{x} \in \mathbb{F}_{q^m}^{ns}, \|\boldsymbol{x}\| = w, \boldsymbol{y} = \boldsymbol{x}\boldsymbol{H}^T\}$$

Hardness of the problems: As we have seen in the Section 1.1.2, the ideal codes are a generalization of quasi-cyclic codes. Although there is no general complexity result for quasi-cyclic codes, decoding these codes is considered hard by the community. In rank metric, there is no known speed-up which exploit this structure. he conclusion is that in practice, the best attacks are the same as those for random codes.

Finally we need to introduce a last problem which is useful for the security proof of our schemes. (see Sections 4).

Problem 1.6 (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 5.

1.3 The Low Rank Parity Check codes

The LRPC codes have been introduced in [9]. They are good candidates for the cryptosystem of McEliece because the have a weak algebraic structure.

Definition 1.3.1 (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 1.3.2 (Ideal LRPC codes)**.** *Let F be a $\mathbb{F}_q$-subspace of dimension d of $\mathbb{F}_{q^m}$, $(\boldsymbol{h}_1, \boldsymbol{h}_2)$ 2 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 1.7 (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 [13].

1.4 A support recovery algorithm

Notation 1.8. *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. So, $\dim EF \leqslant rd$ with equality with a an overwhelming probability (see [9], Section 3 for more details). We will suppose in the following section that $\dim EF = rd$.

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 $S_i := f_i^{-1} S$ and S_{ij} is defined by $S_{ij} := S_i \cap S_j$.

1.4.1 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 probabilistic support recovery algorithm was recently improved in [3]. The algorithm we present here uses both the general decoding algorithm of the LRPC codes described in [9] and a tweak of the improved algorithm

described in [3] designed to run in constant time.

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

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

Precompute every S_i for $i = 1$ to d

Precompute every $S_{i,i+1}$ for $i = 1$ to $d-1$

for i *from* 1 *to* $d-2$ **do**

  $tmp \leftarrow S + F(S_{i,i+1} + S_{i+1,i+2} + S_{i,i+2})$

  if $\dim(tmp) \leqslant rd$ **then**

    $S \leftarrow tmp$

  end

end

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

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

return E

The algorithm is designed in two parts : the first one is used to recover the whole vector space EF in case S is of dimension $< rd$. This ensures that the second part, which is the general decoding of the LRPC codes, outputs E. Note that we do not need to recover the coordinates of the error vector $\boldsymbol{e}$ since we only need the support E for our cryptosystems.

1.4.2 Probability of failure

The second part of the algorithm will fail if the subspace S' obtained at the end of Part 1 is different from EF. Thus the global probability of failure depends both on the probability of $\dim(S)$ being smaller than rd and the probability of not recovering EF during the first part of the algorithm.

Notation 1.9. *In the following, c is the codimension of S as a subspace of EF :* $\dim(S) = rd - c$. *$P(c = i)$ denotes the probability that* $\dim EF - \dim S = i$ *and $P_{c=i}(failure)$ denotes the probability of not recovering EF when $c = i$.*

Proposition 1.10. *The probability of failure of the new algorithm is* $\sum_{i=1}^{rd-1} P(c = i) \times P_{c=i}(failure)$

Proposition 1.11. *The probability that* $\dim S = rd - i$ *is*

$$P(c = i) = q^{-nrd} \prod_{j=0}^{rd-i} \frac{(q^n - q^j)(q^{rd} - q^j)}{q^{rd-i} - q^j} = \Theta\left(q^{-i(n-rd+i)}\right)$$

Proof. Since each coordinate of $\boldsymbol{s}$ is a random element of EF, $\boldsymbol{s}$ can be associated to a random matrix of size $rd \times n$ over $\mathbb{F}_q$. Thus the probability $P(c = i)$ is equal to the number of matrices of size $rd \times n$ and of rank $rd - i$ divided by the total number of matrices of size $rd \times n$.

The number of matrices of size $rd \times n$ and of rank $rd - i$ is $\prod_{j=0}^{rd-i} \frac{(q^n - q^j)(q^{rd} - q^j)}{q^{rd-i} - q^j}$ [18].

The number of matrices of size $rd \times n$ is q^{nrd} hence the first equality.

The second one is straightforward since $(q^j - q^i) = \Theta(q^j)$ for $j > i$. □

Analysis of $P_{c=1}(failure)$

This algorithm uses the fact that $\dim(S_i \cap E) \geqslant r - 1$ for all $i \in [1..d]$, which means S_i contains at least $r - 1$ independent vectors of E. Since all other vectors in S_i are random, we need to intersect two different S_i in order to recover $r - 2$ independent vectors of E : those are the S_{ij}.

At each iteration, we compute $S_{i,i+1} + S_{i+1,i+2} + S_{i,i+2}$ to find vectors of E. Once we have those, we multiply them by the vector space F to find vectors of EF. If one of these vectors (denoted by $\mathbf{x}$) is not in S, then $S + \mathbf{x} = EF$: we can decode successfully.

We know that every S_{ij} contains at least $r - 2$ vectors of E. To study what happens during each iteration of the algorithm, we suppose that S_{ij} contains exactly $r - 2$ vectors of E. Two cases may occur during each of the $d - 2$ iterations :

- If $S_{i,i+1} = S_{i+1,i+2}$, then $\dim(S_{i,i+1} + S_{i+1,i+2} + S_{i,i+2}) = r - 2$, since the equality implies that each vector that we find is in S_i, S_{i+1} and S_{i+2} at the same time. In that case the algorithm might not find new vectors of EF. This equality happens with probability q^{2-r}.

- If $S_{i,i+1} \neq S_{i+1,i+2}$, then $\dim(S_{i,i+1} + S_{i+1,i+2} + S_{i,i+2}) = r$: the inequality implies that $\dim(S_{i,i+1} + S_{i+1,i+2}) = r - 1$ and, since $S_{i,i+2}$ is different from both of the other S_{ij} (otherwise we would be in the first case), the union of the three S_{ij} is exactly E. In that case the algorithm always finds EF.

Since each iteration can fail to recover EF with probability q^{2-r}, the probability of not finding EF when $\dim(S) = rd - 1$ is $q^{(2-r)(d-2)}$.

Analysis of $P_{c>1}(failure)$

Since $P(c = i)$ decreases extremely fast (by at least a factor q^{-n+rd}), we have $P(c \geqslant 3) \leqslant P(c = 1) \times P_{c=1}(failure)$ for all sets of parameters.

The case $c = 2$ is more complicated. Indeed, in practice, Algorithm 1 can still decode when $c = 2$ but the probability of failure in this case is difficult to evaluate. However our simulations have shown that, for every set of parameters where $P(c = 2) < P(c = 1) \times P_{c=1}(failure)$ (i.e ROLLO-I-192, ROLLO-I-256, ROLLO-III-192 and ROLLO-III-256), we have $P_{c=2}(failure) < 2^{-20}$, hence the DFR is always majored by $P(c = 1) \times P_{c=1}(failure)$.

We can conclude this section with the following proposition.

Proposition 1.12. *According to Propositions 1.10 and 1.11 and the analysis of $P_{c=i}(failure)$, the Decryption Failure Rate (DFR) of our schemes is majored by $q^{(2-r)(d-2)}q^{-(n-rd+1)} = q^{-(n-2*(r+d)+5)}$ for all sets of parameters.*

Notice that the algorithm supposes that m is sufficiently higher than $2rd - r$ to work, which will be the case for all parameters considered.

1.5 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 $\mathbb{F}_{q^m}$ and $\mathcal{S}_{1,w}^n(\mathbb{F}_{q^m})$ stands for the set of vectors of length n of rank weight w, *such that its support contains* 1:

$$\mathcal{S}_w^n(\mathbb{F}_{q^m}) = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \dim \mathrm{Supp}(\boldsymbol{x}) = w\}$$
$$\mathcal{S}_{1,w}^n(\mathbb{F}_{q^m}) = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \dim \mathrm{Supp}(\boldsymbol{x}) = w, 1 \in \mathrm{Supp}(\boldsymbol{x})\}$$

1.5.1 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.

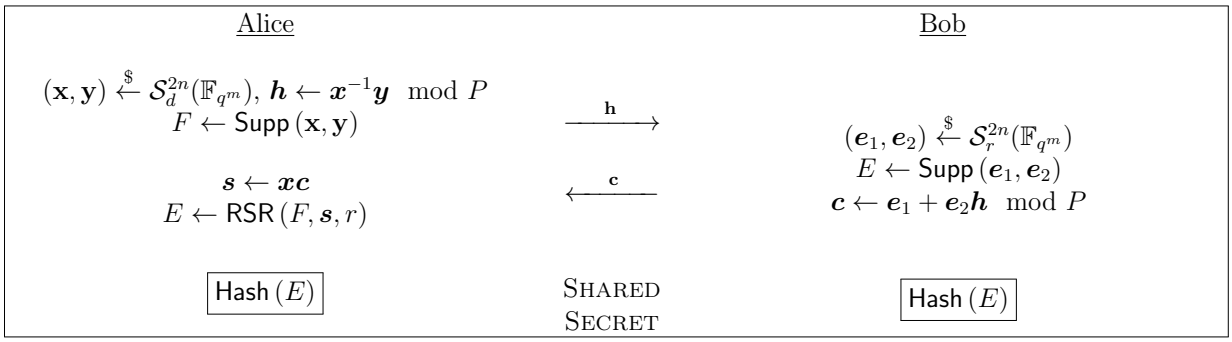


Figure 1: Informal description of ROLLO-I. **h** constitutes the public key.

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 **s** generate a subspace of EF on which Bob can apply the RSR algorithm to recover E.

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

- $\mathsf{KeyGen}(1^\lambda)$: 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 $\mathsf{pk} = \boldsymbol{h}$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- $\mathsf{Encap}(\mathsf{pk})$: 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 $K = \mathsf{Hash}(E)$ and returns $\mathbf{c}$.
- $\mathsf{Decap}(\mathsf{sk})$: Sets $\boldsymbol{s} = \boldsymbol{x}\boldsymbol{c} \mod P$, $F = \mathsf{Supp}(\boldsymbol{x}, \boldsymbol{y})$ and $E \leftarrow \mathsf{RSR}(F, \mathbf{s}, r)$. Recovers $K = \mathsf{Hash}(E)$.

Figure 2: Formal description of ROLLO-I.

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.

1.5.2 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.

1.5.3 ROLLO-III as a KEM

This variant is based on the works of Ouroboros [1], adapted to the context of ideal codes in rank metric. Unlike ROLLO-I, the security of this KEM does not depend on the indistinguishability of ideal LRPC but only on the hardness of the DIRSD problem. This weaker security assumption comes at the price of a keysize slightly larger and a ciphertext size two times larger than ROLLO-I. We will deal with the details of these differences in the appropriate sections.

As previously, ROLLO-III is depicted in fig. 4, then formally described in fig. 5.

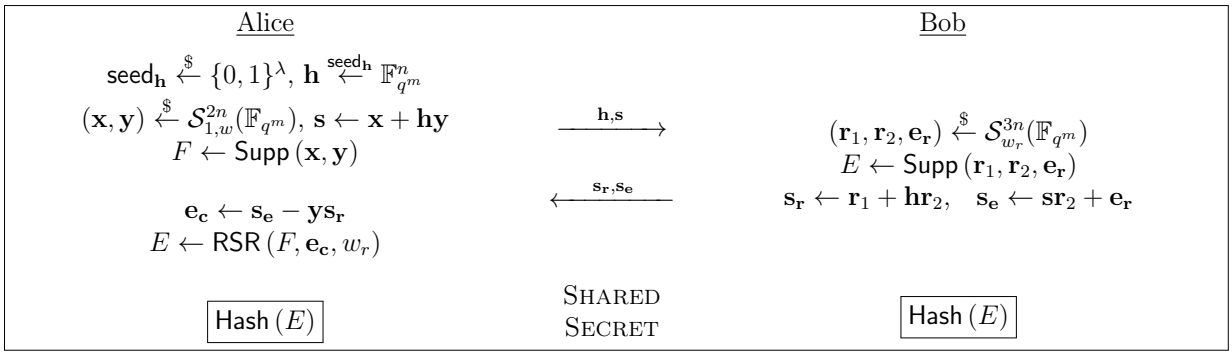


Figure 4: Informal description of ROLLO-III. $\mathbf{h}$ and $\mathbf{s}$ constitute the public key.

Correctness: Alice recovers $\mathbf{e_c} = \mathbf{s_e} - \mathbf{ys_r} = \mathbf{sr}_2 + \mathbf{e_r} - \mathbf{y}(\mathbf{r}_1 + \mathbf{hr}_2) = (\mathbf{x} + \mathbf{hy})\mathbf{r}_2 + \mathbf{e_r} - \mathbf{y}(\mathbf{r}_1 + \mathbf{hr}_2) = \mathbf{xr}_2 - \mathbf{yr}_1 + \mathbf{e_r}$, since $E = \mathsf{Supp}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{e_r})$ and $F = \mathsf{Supp}(\mathbf{x}, \mathbf{y})$, and since $1 \in F$, the coordinates of $\mathbf{e_c}$ generate a subspace of EF on which one can apply the RSR algorithm to recover E.

- $\mathsf{KeyGen}(1^\lambda)$: Picks $\mathsf{seed}_{\mathbf{h}} \xleftarrow{\$} \{0,1\}^{\lambda}$, $\mathbf{h} \xleftarrow{\mathsf{seed}_{\mathbf{h}}} \mathbb{F}_{q^m}^n$, $(\mathbf{x},\mathbf{y}) \xleftarrow{\$} \mathcal{S}_{1,w}^{2n}(\mathbb{F}_{q^m})$.
  Sets $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$ and returns $\mathsf{pk} = (\mathbf{h}, \mathbf{s})$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- $\mathsf{Encap}(\mathsf{pk})$: Picks $(\mathbf{r}_1,\mathbf{r}_2,\mathbf{e}_{\mathbf{r}}) \xleftarrow{\$} \mathcal{S}_{w_r}^{3n}(\mathbb{F}_{q^m})$, sets $E = \mathsf{Supp}(\mathbf{r}_1,\mathbf{r}_2,\mathbf{e}_{\mathbf{r}}), \mathbf{s}_{\mathbf{r}} = \mathbf{r}_1 + \mathbf{hr}_2, \mathbf{s}_{\mathbf{e}} = \mathbf{sr}_2 + \mathbf{e}_{\mathbf{r}}$
  Computes $K = \mathsf{Hash}(E)$, and returns $\mathbf{c} = (\mathbf{s}_{\mathbf{r}}, \mathbf{s}_{\mathbf{e}})$
- $\mathsf{Decap}(\mathsf{sk})$: Sets $\mathbf{e_c} = \mathbf{s_e} - \mathbf{ys_r}$, $F = \mathsf{Supp}(\boldsymbol{x}, \boldsymbol{y})$ and $E \leftarrow \mathsf{RSR}(F, \mathbf{e_c}, w_r)$
  Recovers $K = \mathsf{Hash}(E)$

Figure 5: Formal description of ROLLO-III.

Computational costs. The costs are expressed in operations in the base field $\mathbb{F}_q$. The KeyGen cost corresponds to a polynomial modular addition and a polynomial modular multiplication in $\mathbb{F}_{q^m}[X]/\langle P \rangle$, the Encap costs corresponds to two polynomial modular additions and two polynomial modular multiplications and Decap costs correspond to a polynomial modular addition and a polynomial modular multiplication, plus the decoding cost of the RSR algorithm.

1.6 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 1.7) thus $e \in \mathbb{F}_{q^m}$ is represented as $(e_0, \ldots, e_{m-1}) \in \mathbb{F}_2^m$. Elements are stored using $\lceil \frac{m}{64} \rceil$ quadwords in which the unused $64 \times \lceil \frac{m}{64} \rceil - m$ bits 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 1.

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

m	P_m
79	$X^{79} + X^9 + 1$
83	$X^{83} + X^7 + X^4 + X^2 + 1$
89	$X^{89} + X^{38} + 1$
101	$X^{101} + X^7 + X^6 + X + 1$
107	$X^{107} + X^9 + X^7 + X^4 + 1$
113	$X^{113} + X^9 + 1$
127	$X^{127} + X + 1$
131	$X^{131} + X^8 + X^3 + X^2 + 1$

Table 1: Polynomials used to construct $\mathbb{F}_{2^m}$.

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 seedexpander 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 $2^{32} - 1$.

1.6.1 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.

1.7 Parameters for our schemes

1.7.1 General remarks

In this Section, we propose several sets of parameters for ROLLO-I, ROLLO-II and ROLLO-III, 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 code 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 [14]), 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 5 for more details on best known attacks.

1.7.2 ROLLO-I

Choice of parameters. In section 4.2, the security of the protocol is reduced to the ILRPC problem 1.7 and the IRSR problem 1.6. Our parameters are chosen in function of the best known attacks on these problems described in Section 5.

The probability of decryption failure (DFR) comes from the probability that the RSR algorithm 1 fails (see Proposition 1.12).

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).

The aforementioned sizes are the ones used in our reference implementation.

Instance	q	n	m	r	d	P	security	DFR
ROLLO-I-128	2	47	79	5	6	$X^{47}+X^5+1$	128	2^{-30}
ROLLO-I-192	2	53	89	6	7	$X^{53}+X^6+X^2+X+1$	192	2^{-32}
ROLLO-I-256	2	67	113	7	8	$X^{67}+X^5+X^2+X+1$	256	2^{-42}

Table 2: Parameters for ROLLO-I.

Instance	pk size	sk size	ct size	ss size	Security
ROLLO-I-128	465	40	465	64	128
ROLLO-I-192	590	40	590	64	192
ROLLO-I-256	947	40	947	64	256

Table 3: Resulting sizes in bytes for ROLLO-I using NIST seed expander initialized with 40 bytes long seeds. The security is expressed in bits.

1.7.3 ROLLO-II

Choice of parameters. In section 4.3, the security of the protocol is reduced to the ILRPC problem 1.7 and the IRSR problem 1.6. Our parameters are chosen in function of the best known attacks on these problems described in Section 5.

The probability of decryption failure (DFR) comes from the probability that the RSR algorithm 1 fails (see Proposition 1.12).

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 4.3.2 for more details) so the ciphertext size is $\lceil \frac{nm}{8} \rceil + 128$ bytes (SHA512 output size).

The aforementioned sizes are the ones used in our reference implementation.

Instance	q	n	m	r	d	P	security	DFR
ROLLO-II-128	2	149	83	5	8	$X^{149} + X^{10} + X^9 + X^7 + 1$	128	2^{-128}
ROLLO-II-192	2	151	107	6	8	$X^{151} + X^3 + 1$	192	2^{-128}
ROLLO-II-256	2	157	127	7	8	$X^{157} + X^6 + X^5 + X^2 + 1$	256	2^{-132}

Table 4: Parameters for ROLLO-II.

Instance	pk size	sk size	ct size	Security
ROLLO-II-128	1546	40	1674	128
ROLLO-II-192	2020	40	2148	192
ROLLO-II-256	2493	40	2621	256

Table 5: Resulting sizes in bytes for ROLLO-II using NIST seed expander initialized with 40 bytes long seeds. The security is expressed in bits.

1.7.4 ROLLO-III

Choice of parameters. In section 4.4, the security of the protocol is reduced to the DIRSD problem 1.5. Our parameters are chosen in function of the best known attacks on this problem described in Section 5.

The probability of decryption failure (DFR) comes from the probability that the RSR algorithm 1 fails (see Proposition 1.12).

Instance	q	n	m	w	$w_{\mathbf{r}}$	P	security	DFR
ROLLO-III-128	2	47	101	5	6	$X^{47}+X^5+1$	128	2^{-30}
ROLLO-III-192	2	59	107	6	8	$X^{59}+X^7+X^4+X^2+1$	192	2^{-36}
ROLLO-III-256	2	67	131	7	8	$X^{67}+X^5+X^2+X+1$	256	2^{-42}

Table 6: Parameters for ROLLO-III.

Instance	pk size	sk size	ct size	ss size	Security
ROLLO-III-128	634	40	1188	64	128
ROLLO-III-192	830	40	1580	64	192
ROLLO-III-256	1138	40	2196	64	256

Table 7: Resulting sizes in bytes for ROLLO-III using NIST seed expander initialized with 40 bytes long seeds. The security is expressed in bits.

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 random vector $\boldsymbol{h} \in \mathbb{F}_{2^m}^n$ and a vector $\boldsymbol{s} \in \mathbb{F}_{2^m}^n$, so its size is $\lceil \frac{nm}{8} \rceil + 40$ bytes.

The secret key sk is composed of two random vectors of $\mathcal{S}_{1,w}^n(\mathbb{F}_{2^m})$, so its size is 40 bytes.

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

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

The aforementioned sizes are the ones used in our reference implementation.

2 Performances

In this section, we provide concrete performance measures of our implementation. For each parameter set, results have been obtained by running 10,000 random instances and computing their average median time. The benchmarks have been performed on a machine running Archlinus. The latter 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 8.2.1) and use the `openssl` library (version 1.1.1b).

2.1 ROLLO-I

Reference Implementation

The performance of our reference implementation on the aforementioned benchmark platform are described in Tab. 8 (millions of CPU cycles) and Tab. 9 (timing in ms). The following compilation flags have been used: `-O3 -flto -pedantic`.

Instance	KeyGen	Encap	Decrypt
ROLLO-I-128	1.03	0.16	0.81
ROLLO-I-192	1.29	0.18	1.54
ROLLO-I-256	2.29	0.27	2.90

Table 8: Performances of ROLLO-I reference implementation in millions of CPU cycles.

Instance	KeyGen	Encap	Decrypt
ROLLO-I-128	0.29	0.05	0.23
ROLLO-I-192	0.36	0.05	0.42
ROLLO-I-256	0.64	0.07	0.81

Table 9: Performances of ROLLO-I reference implementation in ms.

Optimized Implementation

An optimized implementation using AVX2 instructions have been provided. Its performances on the aforementioned benchmark platform are described in Tab. 10 (millions of CPU cycles) and Tab. 11 (timing in ms). The following compilation flags have been used: `-O3 -flto -mavx2 - mpclmul -msse4.1 -pedantic`.

2.2 ROLLO-II

Reference Implementation

The performance of our reference implementation on the aforementioned benchmark platform are described in Tab. 12 (millions of CPU cycles) and Tab. 13 (timing in ms). The following compilation flags have been used: `-O3 -flto -pedantic`.

Optimized Implementation

An optimized implementation using AVX2 instructions have been provided. Its performances on the aforementioned benchmark platform are described in Tab. 14 (millions of CPU cycles) and Tab. 15 (timing in ms). The following compilation flags have been used: `-O3 -flto -mavx2 - mpclmul -msse4.1 -pedantic`.

Instance	Keygen	Encrypt	Decap
ROLLO-I-128	0.36	0.08	0.65
ROLLO-I-192	0.46	0.08	1.29
ROLLO-I-256	0.70	0.10	2.47

Table 10: Performances of ROLLO-I optimized implementation in millions of CPU cycles.

Instance	Keygen	Encrypt	Decap
ROLLO-I-128	0.10	0.02	0.18
ROLLO-I-192	0.13	0.02	0.36
ROLLO-I-256	0.20	0.03	0.69

Table 11: Performances of ROLLO-I optimized implementation in ms.

Instance	Keygen	Encrypt	Decap
ROLLO-II-128	8.69	0.84	3.17
ROLLO-II-192	10.67	0.98	4.02
ROLLO-II-256	12.44	1.14	4.95

Table 12: Performances of ROLLO-II reference implementation in millions of CPU cycles.

Instance	Keygen	Encrypt	Decap
ROLLO-II-128	2.41	0.23	0.88
ROLLO-II-192	2.96	0.27	1.11
ROLLO-II-256	3.45	0.32	1.37

Table 13: Performances of ROLLO-II reference implementation in ms.

Instance	Keygen	Encrypt	Decap
ROLLO-II-128	2.46	0.29	1.90
ROLLO-II-192	2.98	0.33	2.50
ROLLO-II-256	2.84	0.34	3.03

Table 14: Performances of ROLLO-II optimized implementation in millions of CPU cycles.

Instance	Keygen	Encrypt	Decap
ROLLO-II-128	0.69	0.08	0.53
ROLLO-II-192	0.83	0.09	0.69
ROLLO-II-256	0.79	0.10	0.84

Table 15: Performances of ROLLO-II optimized implementation in ms.

Instance	Keygen	Encap	Decap
ROLLO-III-128	0.20	0.35	0.68
ROLLO-III-192	0.24	0.42	1.64
ROLLO-III-256	0.44	0.79	2.89

Table 16: Performances of ROLLO-III reference implementation in millions of CPU cycles.

Instance	Keygen	Encap	Decap
ROLLO-III-128	0.06	0.10	0.19
ROLLO-III-192	0.07	0.12	0.45
ROLLO-III-256	0.12	0.22	0.80

Table 17: Performances of ROLLO-III reference implementation in ms.

2.3 ROLLO-III

Reference Implementation

The performance of our reference implementation on the aforementioned benchmark platform are described in Tab. 16 (millions of CPU cycles) and Tab. 17 (timing in ms). The following compilation flags have been used: `-O3 -flto -pedantic`.

Optimized Implementation

An optimized implementation using AVX2 instructions have been provided. Its performances on the aforementioned benchmark platform are described in Tab. 18 (millions of CPU cycles) and Tab. 19 (timing in ms). The following compilation flags have been used: `-O3 -flto -mavx2 - mpclmul -msse4.1 -pedantic`.

2.4 Constant time Implementation

The decoding algorithm 1 has been designed to perform the same number of intersections for any dimension of the input subspace S. Measuring the execution time taken to perform an intersection, an attacker might be able to recover the codimension of S in EF, but the

Instance	Keygen	Encrypt	Decap
ROLLO-III-128	0.10	0.16	0.51
ROLLO-III-192	0.12	0.18	1.36
ROLLO-III-256	0.18	0.26	2.30

Table 18: Performances of ROLLO-III optimized implementation in millions of CPU cycles.

Instance	Keygen	Encrypt	Decap
ROLLO-III-128	0.03	0.04	0.14
ROLLO-III-192	0.04	0.05	0.38
ROLLO-III-256	0.05	0.07	0.63

Table 19: Performances of ROLLO-III optimized implementation in ms.

results of [2] show that for ROLLO-I and ROLLO-III, use of ephemeral keys negate this attack, and for ROLLO-II, the probability of obtaining subspaces S of codimension 1 is low enough that the attack is not practical.

3 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.

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.

4 Security

4.1 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. 6.

$\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
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 6: Experiment against the indistinguishability under chosen plaintext attacks

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

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(\lambda, t) = \max_{\mathcal{A} \leq t} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda), \tag{3}$$

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

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left| \Pr[\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] \right|. \tag{4}$$

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].

4.2 IND-CPA security proof of ROLLO-I

Theorem 4.1. *Under the* Ideal LRPC indistinguishability *1.7 and the* Ideal-Rank Support Recovery *1.6 Problems, the KEM presented earlier in section 1.5.1 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

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

- 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

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

  which leads to the conclusion.

□

4.3 IND-CCA2 security proof of ROLLO-II

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

Theorem 4.2. *Under the* Ideal LRPC indistinguishability *1.7 and the* Ideal-Rank Support Recovery *1.6 Problems, the encryption scheme presented earlier in section 1.5.2 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.

□

4.3.2 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 $\mathcal{K}$ be hash functions, typically SHA512 as advised by NIST. The KEM-DEM version of the ROLLO-II cryptosystem is defined as follows (following [16]) :

When applying the HHK [16] 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.

- $\mathsf{Setup}(1^\lambda)$: 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 7: Description of our proposal ROLLO-II.KEM.

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, 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.

4.4 IND-CPA security proof of ROLLO-III

Theorem 4.3. *ROLLO-III is* IND-CPA *secure under the assumption of the hardness of the 2-DIRSD and 3-DIRSD problems 1.5.*

The proof follows the same ideas as the one from [1, Proof of Theorem 1].

Proof. It is worth noticing first that an adversary $\mathcal{A}$ succeeds in breaking the scheme if he manages to recover the support $\mathbf{E} = \mathrm{Supp}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{e}_\mathbf{r})$ given only the public key $\mathsf{pk} = (\mathbf{h}, \mathbf{s})$ and the transcripts $(\mathbf{s}_\mathbf{r} = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2, \mathbf{s}_\mathbf{e} = \mathbf{s}\mathbf{r}_2 + \mathbf{e}_\mathbf{r})$. This is exactly an instance of the IRSR problem defined in Sec. 1.2 (see Pb. 1.6). It has also been shown in that section that the IRSR problem is equivalent to the IRSD problem, and the security reduction exploits the equivalence between these problems.

The aim is to prove that an adversary distinguishing one game from another can be exploited to break either the 2-DIRSD or the 3-DIRSD assumption (respectively on $[2n, n]$

or $[3n, n]$ codes) in polynomial time. We are going to proceed in a sequence of games moving from the real world with a valid encryption, to an idealistic version where both the ciphertext and the key are random. Let $\mathcal{A}$ be a probabilistic polynomial time adversary against the IND-CPA of our scheme and consider the following games where we consider that $\mathcal{A}$ receives the encapsulation at the end of each game.

- **Game $\mathbf{G_1}$** : This game corresponds to an honest run of the protocol. In particular, the simulator has access to all keys / randomness.

- **Game $\mathbf{G_2}$** : Now the simulator picks uniformly at random $\mathbf{x}, \mathbf{y}$ (resulting in a random $\mathbf{s}$). He then proceeds honestly.

  An adversary distinguishing between those two games, can distinguish between a well-formed pk and a random one. The public key in the first game correspond to a valid 2-DIRSD instance, while it is a random one in the second.

  Hence $\mathsf{Adv}^{G_1-G_2}_{\mathcal{A}_{1,2}} \leq \mathsf{Adv}(\text{2-DIRSD})(\lambda)$

- **Game $\mathbf{G_3}$** : Now the simulator also picks uniformly at random $\mathbf{e_r}$, $\mathbf{r}_1$ and $\mathbf{r}_2$ and uses them to generate $\mathbf{s_r}, \mathbf{s_e}$.

  An adversary has access to:

$$\begin{pmatrix} \mathbf{s_r} \\ \mathbf{s_e} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathcal{IM}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathcal{IM}(\mathbf{s}) \end{pmatrix} (\mathbf{r}_1, \mathbf{e_r}, \mathbf{r}_2)^\top$$

  The syndrome $(\mathbf{s_r}, \mathbf{s_e})$ follows the QCRSD distribution in game $\mathbf{G_2}$ and the uniform distribution over $(\mathbb{F}_2^n)^2$ in $\mathbf{G_3}$. If an adversary is able to distinguish games $\mathbf{G_2}$ from $\mathbf{G_3}$, then a simulator can break the underlying problem.

  Hence $\mathsf{Adv}^{G_2-G_3}_{\mathcal{A}_{2,3}} \leq \mathsf{Adv}(\text{3-DIRSD})(\lambda)$.

$$\mathsf{Adv}^{\text{IND-CPA}}_{\text{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{2-DIRSD}}(\lambda) + \mathsf{Adv}^{\text{3-DIRSD}}(\lambda).$$

Therefore, a PPT adversary $\mathcal{A}$ breaking the protocol with non negligible advantage can be used to solve the QCRSD problem. □

5 Known Attacks

In this section, we present the best known attacks against the IRSR 1.6, ILRPC 1.7 and DIRSD 1.5 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 Groebner 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.

5.1 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 1.4 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 [10], a detailed description of the attack can be found in [4]. 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 5.1. *Since the linear system is not random, it is reasonable to take $\omega = 2$ for the choice of the parameters of ROLLO-I, ROLLO-II and ROLLO-III, even if the attack described in [4] 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.

5.2 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 [11]. The best algorithm is the same decoding algorithm used in the previous subsection [4]. 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 [14]. 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.

5.3 Algebraic attacks

The second way to solve the equations of the system $\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s}^T$ is to use the Groebner basis [17]. The advantage of these attacks is that they are independent of the size of q. They mainly depend on the number of unknowns with respect to the number of equations. However, in the case $q = 2$ the number of unknowns is generally too high for that the algorithms by Groebner basis are more efficient than the combinatorial attacks. We have chosen our parameters such that the best attacks are combinatorial, the expected complexity of the algorithms by Groebner basis is based on the article [5].

5.4 Quantum speed-up

For computational attacks, the quantum speed-up is easy to analyze. According to [8], 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

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

6 Advantages and Limitations

6.1 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.

ROLLO-III also benefit from the nice features of the LRPC protocol but with a tight reduction to the generic s-DIRSD problems. It comes at price, since the ciphertext size is doubled, but due to the inherent difficulty of decoding codes in rank metric, parameters are rather low, and compare very well to other type of protocols.

6.2 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 and it seems very hard to improve on their general complexity.

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] Nicolas Aragon and Philippe Gaborit. A key recovery attack against lrpc usingdecryption failures. In *Coding and Cryptography, International Workshop, WCC 2019*, 2019.

[3] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes: New decoding algorithms and application to cryptography. Available on http://arxiv.org/abs/1111.4301, 2018.

[4] 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.

[5] Luk Bettale, Jean-Charles Faugere, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3):177–197, 2009.

[6] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *LNCS*, pages 245–255. Springer, 1996.

[7] 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.

[8] 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.

[9] 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`.

[10] 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.

[11] 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.

[12] Philippe Gaborit and Gilles Zémor. On the hardness of the decoding and the minimum distance problems for rank codes. *IEEE Trans. Information Theory*, 62(12):7245–7252, 2016.

[13] Adrien Hauteville and Jean-Pierre Tillich. New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem, 2015. abs/1504.05431.

[14] 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.

[15] 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.

[16] 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.

[17] 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`.

[18] Pierre Loidreau. *Rank metric and cryptography.* Accreditation to supervise research, Université Pierre et Marie Curie - Paris VI, January 2007.

[19] 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.

[20] Victor Shoup. Ntl: A library for doing number theory., 2001.