Analysis of Layered-ROLLO-I

Tanja Lange[1] and Alex Pellegrini[1]

[1]Eindhoven University of Technology

September 5, 2023

Abstract

In this short note we show that the proposed parameters of the modified version of Layered-ROLLO-I fall short of the claimed security levels. We further improve on these complexities by providing a direct attack to the modified version which reduces the system to a smaller version of ROLLO-I.

1 Introduction and background

We analyze Layered-ROLLO-I [KKN23b, KKN22a, KKN22b], a rank metric code-based KEM. The scheme was attacked by Chee, Jeong, Lee, and Ryu [CJLR23], showing how to reduce the security to the original ROLLO-I. Six weeks later, the designers of Layered–ROLLO-I released an updated version of their design. We now show that the modified version can be reduced to ROLLO-I [ABD+19] by slightly modifying the attack in [CJLR23]. ROLLO (including ROLLO-I) was deselected from the NIST competition after round 2 because of attacks on rank-metric codes decreasing the security below the required levels.

This report is organized as follows. The rest of this section is dedicated to some background and notation. For the sake of simplicity, we will refrain from introducing the entire framework of rank metric codes, (ideal and blockwise interleaved ideal) low-rank parity check (LRPC) codes, since these notions will not be directly used in the attacks and analysis. Section 2 introduces the original ROLLO-I proposal [ABD+19], submitted to the second round of NIST's competition on post-quantum cryptography, and gives its specification. In section 3 we give the specification of Layered-ROLLO-I and the attack proposed on the KpqC forum [CJLR23]. Finally in section 4 we describe the Modified Layered-ROLLO-I system and recompute the attack costs of Rank Syndrome Decoding (RSD) following the improvements in [BBC+20]. We show that we can slightly modify the attack in section 3 to reduce this system to ROLLO-I and further improve on the attack complexities.

1.1 Notation

In the specifications in this report we will make use of the following objects. Denote by $S_w^n(\mathbb{F}_{q^m})$ the set of vectors of length n and rank weight w over $\mathbb{F}_{q^m}$:

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

The Rank Support Recovery ($\mathsf{RSR}(F, s, r)$) algorithm is used as a decoder in the decapsulation procedures of ROLLO-I and the followup designs. It recovers the support of ($\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}$ generated by) the error vector given the support E of the secret key and the rank of the error. This corresponds to actually finding the error coordinates, by solving a linear system of equations (see p. 13 of the ROLLO specification [ABD+19]).

Let $P(x) \in \mathbb{F}_q[x]$ be a polynomial of degree n. We can identify the vector space $\mathbb{F}_{q^m}^n$ with the ring $\mathbb{F}_{q^m}[x]/(P(x))$, where $(P(x))$ is the ideal of $\mathbb{F}_{q^m}[x]$ generated by $P(x)$. Given $\mathbf{u} \in \mathbb{F}_{q^m}^n$, denote by $\mathbf{u}(x) \in \mathbb{F}_{q^m}[x]$ the polynomial $\mathbf{u}(x) = \sum_{i=0}^{n-1} u_i x^i$. Given $\mathbf{u}, \mathbf{v} \in \mathbb{F}_{q^m}^n$, we define their product $\mathbf{uv}$ as the unique vector $\mathsf{w} \in \mathbb{F}_{q^m}^n$ such that $\mathsf{w}(x) = \mathbf{u}(x)\mathbf{v}(x) \bmod P(x)$. Similarly, we define $Q\mathbf{u} = Q(x)\mathbf{u}(x) \bmod P(x)$ for $Q(x) \in \mathbb{F}_{q^m}[x]$ and $\mathbf{u}^{-1}$ for $\mathbf{u}(x)$ invertible modulo $P(x)$.

1.2 Acknowledgments

The authors would like to thank Daniel J. Bernstein for discussions regarding recovering P_I in the attack presented in section 4.1.

2 ROLLO-I

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

KeyGen:

- Pick random $\mathbf{x}, \mathbf{y} \in S_d^n(\mathbb{F}_{q^m})$.
- Set $\mathbf{h} = \mathbf{x}^{-1}\mathbf{y} \bmod P$.
- Return $\mathsf{pk} = \mathbf{h}$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.

Encap(pk):

- Pick random $\mathbf{e}_1, \mathbf{e}_2 \in S_r^n(\mathbb{F}_{q^m})$.
- Set $E = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle$.
- Return $K = \mathsf{hash}(E)$ and $\mathbf{c} = \mathbf{e}_1 + \mathbf{e}_2\mathbf{h} \bmod P$.

Decap(sk):

- Set $\mathbf{s} = \mathbf{xc} \bmod P$, $F = \langle \mathbf{x}, \mathbf{y} \rangle$ and $E = \mathsf{RSR}(F, \mathbf{s}, r)$, where $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the $\mathbb{F}_q$-vector space spanned by the columns of $\mathbf{x}$ and $\mathbf{y}$ (interpreted as vectors in $\mathbb{F}_q^m$).
- Return $K = \mathsf{hash}(E)$.

3 Layered-ROLLO-I

This description follows [KKN23b] apart from skipping the explicit maps between coefficient vectors and polynomials. The values (q, n, m, r, d, b, P) are the system parameters, where q, n, m, r, d, b are integers, with n a multiple of b, and $P \in \mathbb{F}_{q^m}[x]$ is a primitive polynomial of degree n/b. For all given parameter sets $b = 2$ and in any case $b < n/b$. The map $\Psi : \mathbb{F}_{q^m}[x]/(P) \to \mathbb{F}_{q^m}[x]/(P^b)$ casts polynomials of the first quotient into the second quotient by mapping the input to the unique polynomial of degree $< n/b$ that is congruent to it modulo P^b. Similarly the map $\Omega : \mathbb{F}_{q^m}[x]/(P^b) \to \mathbb{F}_{q^m}[x]/(P)$ reduces the input modulo P. Since P^b is a multiple of P these maps are well defined.

KeyGen:

- Pick random $\mathbf{x}, \mathbf{y} \in S_d^{n/b}(\mathbb{F}_{q^m})$.
- Pick random irreducible $P_I \in \mathbb{F}_{q^m}[x]/(P)$ of degree $(b-1)$.
- Pick random $P_O, P_N \in \mathbb{F}_{q^m}[x]/(P^b)$ of degree n, with P_O invertible (this last restriction is not stated but required for functionality).
- Set $\mathbf{z} = P_I \mathbf{x}^{-1}\mathbf{y} \bmod P$.
- Set $P_P = P_O \Psi(P_I) \bmod P^b$ and $P_H = P_O \Psi(\mathbf{z}(x)) + P_N P \bmod P^b$.
- Return $\mathsf{pk} = (P_P, P_H)$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y}, P_O, P_I)$.

Encap(pk):

- Pick random $E = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle \in S_r^{n/b}(\mathbb{F}_{q^m})$, with $\mathbf{e}_1, \mathbf{e}_2$ each corresponding to a polynomial of degree $< n/b - b$.
- Set $P_{E_1} = \Psi(\mathbf{e}_1(x))$ and $P_{E_2} = \Psi(\mathbf{e}_2(x))$.
- Compute $\mathbf{c}(x) = P_P P_{E_1} + P_H P_{E_2} \bmod P^b$.
- Return $K = \mathsf{hash}(E)$ and $\mathbf{c}$.

Decap(sk):

- Compute $P_C = P_O^{-1} c(x) \bmod P^b$.
- Compute $\mathbf{c}'(x) = P_I^{-1}\Omega(P_C) \bmod P$.
- Decode $E = \mathsf{RSR}(\langle \mathbf{x}, \mathbf{y} \rangle, \mathbf{x}\mathbf{c}', r)$.
- Return $K = \mathsf{hash}(E)$.

3.1 Reduction of Layered-ROLLO-I to ROLLO-I

Chee, Jeong, Lee, and Ryu [CJLR23] recomputed the costs of some rank decoding attacks, finding out that the proposed parameters were not suitable for the requested security levels. They also proposed a new reduction of Layered-ROLLO-I to ROLLO-I by using exclusively the public key of the former. We give an overview of how the reduction works.

To start with, notice that P_O must has an inverse modulo P^b. This has not been declared in the specification but the decapsulation process requires P_O^{-1}. If not, decapsulation fails. Also, P_I is irreducible of degree $(b-1) < n/b = \deg P$, so it has an inverse modulo P and thus $\Psi(P_I)$ is invertible modulo P^b. Therefore we can invert P_P modulo P^b and compute

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

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

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

where we use the fact that $\Psi(P_I) \bmod P = P_I \bmod P$ and $\Psi(\mathbf{z}(x)) \bmod P = P_I \mathbf{x}^{-1}\mathbf{y}$.

This shows that the public key of (q, n, m, r, d, b, P)-Layered-ROLLO-I can be reduced to the public key of $(q, n/b, m, r, d, P)$-ROLLO-I. The same can be done for ciphertexts by computing $P_P^{-1}\mathbf{c}$ and proceeding as explained above. Therefore it is possible to reduce an entire instance of (q, n, m, r, d, b, P)-Layered-ROLLO-I to an instance of $(q, n/b, m, r, d, P)$-ROLLO-I.

Their paper does not include a table for the new attack costs, but with the reduction [CJLR23] show that the security is even less than their initial tables show.

4 Modified Layered-ROLLO-I

This section extracts the description of the modified system from [KKN23a].

The values $(q, n_1, n_2, \deg P_I, m, r, d, b)$, where $\deg P_I < n_1 < n_2$ are the system parameters. There are also two primitive polynomials P_1 and P_2 of degrees n_1 and n_2 respectively. These are not stated among the system parameters but are needed for the functioning of the system. In the Implementation Codes from 19 May 2023, the polynomial P_1 is still the same hardcoded polynomial as for ROLLO-I; the polynomial P_2 may be included either in the system parameters or in the public key. The slides introducing modified Layered ROLLO-I point to an increase in the public key size which could be compatible with including P_2 in the public key, however it is not stated and we see no benefits of generating P_2 per user, and thus increasing the size of the public key, over including it in the system parameters. In the following we assume that P_1 and P_2 are part of the system parameters.

KeyGen:

- Pick random $\mathbf{x}, \mathbf{y} \in S_d^{n_1}(\mathbb{F}_{q^m})$.
- Pick random irreducible $P_I \in \mathbb{F}_{q^m}[x]/(P_1)$ of degree $\deg P_I$.
- Pick random $P_O \in \mathbb{F}_{q^m}[x]/(P_2)$.
- Set $\mathbf{z} = P_I \mathbf{x}^{-1}\mathbf{y} \bmod P_1$.
- Set $P_P = P_O\Psi(P_I) \bmod P_2$ and $P_H = P_O\Psi(\mathbf{z}) \bmod P_2$.

- Return $\mathsf{pk} = (P_P, P_H)$ and $\mathsf{sk} = (\mathbf{x}, \mathbf{y}, P_O, P_I)$.

The proposed parameters for the modified Layered-ROLLO-I along with the attacks costs are displayed in Table 1. The complexities of the attacks are computed using the script provided in [LPR23], the Sage script performs puncturing of the public code to find the optimal complexity.

Security parameter	(q, n_1, n_2, m, r, d)	Cost [BBC+20]
128	$(2, 37, 61, 67, 6, 2)$	93.72
192	$(2, 43, 71, 79, 7, 3)$	105.90
256	$(2, 53, 103, 97, 7, 3)$	114.10

Table 1: Suggested parameters and values of the $\log_2$ of attack costs for modified Layered-ROLLO-I's suggested parameters. See Modified Layered-ROLLO-I.

This table shows that the security is still lower for these parameters than the targeted security levels, even though the designers were now aware of the attacks in [BBC+20].

The modified version of Layered-ROLLO-I overcomes the reduction in section 3.1 by replacing the two moduli P and P^b by two primitive polynomials P_1 and P_2 of degree n_1 and n_2, respectively. This implies in particular that the polynomials are coprime. Note that for P_2 it is sufficient to choose it as an irreducible polynomial. In this setting, replacing P with P_1 and P^b with P_2 in equation (1), one cannot simply reduce modulo P_1 as the term kP_2 would not vanish.

This might make it seem like decapsulation cannot recover $(\mathbf{e}_1, \mathbf{e}_2)$ because the moduli are incompatible. The KEM works around this problem by reducing the degrees of $\mathbf{e}_1$ and $\mathbf{e}_2$. In this setting, Ω first lifts to $\mathbb{F}_{q^m}[x]$ choosing the unique polynomial of degree less than n_2 and then reduces modulo P_1, Ψ similarly lifts to $\mathbb{F}_{q^m}[x]$ choosing the unique polynomial of degree less than n_1 and then considers this polynomial modulo P_2. Given that $n_2 > n_1$ no reduction is needed.

Encap(pk):

- Pick random $E = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle \in S_r^{n_2}(\mathbb{F}_{q^m})$, with $\mathbf{e}_1, \mathbf{e}_2$ each corresponding to a polynomial of degree $< n_2 - n_1 - \deg(P_I)$.
- Set $P_{E_1} = \mathbf{e}_1(x)$ and $P_{E_2} = \mathbf{e}_2(x)$.
- Compute $\mathbf{c}(x) = P_P P_{E_1} + P_H P_{E_2} \bmod P_2$.
- Return $K = \mathsf{hash}(E)$ and $\mathbf{c}$.

Decap(sk):

- Compute $\mathbf{c}'' = P_O^{-1}\mathbf{c}(x) \bmod P_2$.
- Compute $\mathbf{c}' = P_I^{-1}\Omega(\mathbf{c}'') \bmod P_1$.

- Decode $E = \mathsf{RSR}(\langle \mathbf{x}, \mathbf{y} \rangle, \mathbf{x}\mathbf{c}', r)$.
- Return $K = \mathsf{hash}(E)$.

Decapsulation works because

$$\begin{aligned}
\mathbf{c}'' &= P_O^{-1}(P_P P_{E_1} + P_H P_{E_2}) \\
&= P_O^{-1}(P_O \Psi(P_I) P_{E_1} + P_O \Psi(\mathbf{z}(x)) P_{E_2}) \\
&= \Psi(P_I) P_{E_1} + \Psi(\mathbf{z}(x) P_{E2} \bmod P_2
\end{aligned}$$

and the degree of $\Psi(P_I)P_{E_1} + \Psi(\mathbf{z}(x))P_{E_2}$ is $< n_2$ by the choice of the error vectors. Hence, $\mathbf{c}'' = \Psi(P_I)P_{E_1} + \Psi(\mathbf{z}(x)P_{E2}$ in $\mathbb{F}_{q^m}[x]$ and thus the reduction modulo P_1 preserves the factors P_I which can then be divided out.

4.1 Reduction of modified Layered-ROLLO-I to ROLLO-I

In this section we will describe a reduction of the modified Layered-ROLLO-I to ROLLO-I. Along the way we compute P_I and P_O, meaning that the system leaks private information.

The idea of the reduction remains the same, observing that P_H/P_P cancels the P_O. However, because of the coprimality of the moduli, we cannot proceed directly from there to reducing modulo P_1. However, we know that the polynomials involved have very low degrees. Let $R = P_H/P_P \bmod P_2$ then $\deg(R) < n_2$ and $R = \Psi(\mathbf{z})/\Psi(P_I) \bmod P_2$ with $\deg(\mathbf{z}) < n_1$ and $\deg(P_I)$ small. Note that the division might cancel common factors of P_I and $\mathbf{z}$, however, given the degrees this is unlikely.

Let M_R be the $(\deg P_I + 1) \times n_2$ matrix over $\mathbb{F}_{q^m}$ representing multiplication of a polynomial of degree $\deg P_I$ by R modulo P_2. Consider

$$\Psi(P_I) M_R = \Psi(\mathbf{z}) \tag{2}$$

as a linear system of equations in the coefficients of $\Psi(P_I)$ and $\Psi(\mathbf{z})$, where in this case we view $\Psi(\mathbf{z})$ as an element of $\mathbb{F}_{q^m}^{n_2}$ consisting of the unknown coefficients of $\Psi(\mathbf{z})$ and $n_2 - n_1$ trailing zeroes. Since $\deg(\Psi(P_I)) + n_1 < n_2$, the system has a solution corresponding to the representatives of P_I and $\mathbf{z}$ modulo P_1 (here we remove the Ψ notation as the solutions will have degree lower than n_1).

We can actually compute P_I from a subset of the equations defined by (2). Indeed, let $J \subset \{n_1, \ldots, n_2 - 1\}$ having cardinality $\#J = \deg P_I + 1$. Denote by $M_R(J)$ the submatrix of M_R consisting of the columns indexed by J. We only require $M_R(J)$ to be invertible, which holds for most choices of J as this system is defined over $\mathbb{F}_{q^m}$, so typically we take the last $\deg(P_I) + 1$ columns. Then from (2) we can compute P_I by solving

$$\Psi(P_I) M_R(J) = \mathbf{0} \tag{3}$$

Since also $\lambda \Psi(P_I) M_R(J) = \mathbf{0}$ for any constant $\lambda \in \mathbb{F}_{q^m}$ we can recover P_I only up to such a constant factor. We will now show that this is not a problem. Let $P'_I = \lambda P_I$. We can recover $P'_O = P_P / P'_I = P_O/\lambda$, then $\mathbf{z}' = P_H/P'_O = P'_I\mathbf{x}^{-1}\mathbf{y} = \lambda P_I \mathbf{x}^{-1}\mathbf{y}$, and finally $\mathbf{x}^{-1}\mathbf{y} = \mathbf{z}'/P'_I$ which corresponds to a ROLLO-I public key.

Similarly, for the ciphertext we can recover $\lambda\mathbf{c}'' = (P'_O)^{-1}\mathbf{c} = \lambda\Psi(P_I)P_{E_1} + \lambda\Psi(\mathbf{z}(x))P_{E_2} \bmod P_2$. Since λ is constant, the degree of the right-hand side is below n_2 and we can reduce modulo P_1 and divide by $P'_I = \lambda P_I$ to get $P_{E_1} + \lambda\mathbf{x}^{-1}\mathbf{y}P_{E_2}$, matching the ROLLO-I ciphertexts. Note that the degree constraint on $\deg(P_{E_i}) < n_2 - n_1 - \deg(P_I)$ for all proposed parameters implies that $\deg(P_{E_i}) < n_1$, hence this is a valid ciphertext. While this is not pointed out in the slides, this is also required for the ROLLO-I decoder to work as $\mathsf{RSR}(\langle \mathbf{x}, \mathbf{y}\rangle, \mathbf{x}\mathbf{c}', r)$ in the regular decapsulation procedure.

Our experiments show that the time to recover P_I is split roughly equally between the costs of polynomial division modulo P_2 to obtain R on the one side and the costs of computing the matrix M_R and computing the left kernel of $M_R(J)$ on the other side, where J is a set of $\deg P_I + 1$ columns J chosen as explained above.

The time in seconds to compute the public key transformation described in this section, on a Linux Mint virtual machine, is stated in Table 2.

Security level	$\deg P_I$	Time (s)
128	11	1.85
192	15	2.42
256	20	4.21

Table 2: Average time in seconds (on 50 samples for each security level) needed to reduced a modified Layered-ROLLO-I public key to ROLLO-I public key.

Note that here we use $\deg P_I$ as stated on the slides. The parameters file in the implementation package instead uses $\deg P_I = 4$ for all security levels.

4.2 Updated estimates for the security of modified Layered ROLLO-I

For each security level, the parameters of the reduced ROLLO-I along with the attacks costs are reported in Table 3, again using the same script.

Security parameter	(q, n, m, r, d)	Cost [BBC+20]
128	$(2, 37, 67, 6, 2)$	84.48
192	$(2, 43, 79, 7, 3)$	95.04
256	$(2, 53, 97, 7, 3)$	99.69

Table 3: Values of the $\log_2$ of attack costs for modified Layered-ROLLO-I's suggested parameters.

The reduction in code length and dimension from using n_2 to using n_1 reduces the security further compared to Table 1 by more than 10 bits for each security level.

References

[ABD+19] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. ROLLO. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[BBC+20] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 507–536, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

[CJLR23] Seongtaek Chee, Kyung Chul Jeong, Nari Lee, and Hansol Ryu. Analysis of Layered ROLLO-I. Email to KpqC Bulletin, 2023.

[KKN22a] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Layered ROLLO-I. Submission to KpqC Round 1, 2022.

[KKN22b] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes. Cryptology ePrint Archive, Report 2022/1572, 2022. https://eprint.iacr.org/2022/1572.

[KKN23a] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Comments and modification on Layered ROLLO on kPQC-forum. Slides attached to reply on Kpqc forum, 2023.

[KKN23b] Chanki Kim, Young-Sik Kim, and Jong-Seon No. New design of blockwise interleaved ideal low-rank parity-check codes for fast post-quantum cryptography. *IEEE Commun. Lett.*, 27(5):1277–1281, 2023.

[LPR23] Tanja Lange, Alex Pellegrini, and Alberto Ravagnani. On the security of REDOG. Cryptology ePrint Archive, Paper 2023/1205, 2023. https://eprint.iacr.org/2023/1205.