# An efficient message recovery attack on the new Modified-Layered-ROLLO-I

Alex Pellegrini[1]

[1]Eindhoven University of Technology

October 2, 2023

**Abstract**

In this short note we describe a full break of the Layered-ROLLO-I cryptosystem submitted to the Korean post-quantum cryptography competition. We show that Layered-ROLLO-I encapsulation is equivalent to the encryption of a weak instance of the McEliece cryptosystem and give a message recovery attack against the latter. Our attack takes few seconds to decrypt ciphertexts for any security level on a Linux Mint virtual machine which means that an optimized version is probably faster than the actual decapsulation algorithm. Furthermore, our attack applies to any proposed version of Layered-ROLLO-I with effortless modifications.

## 1 The new modified-Layered-ROLLO-I

In this section we describe the system from [KKN23]. The new version of Layered-ROLLO-I uses polynomial masking techniques in order to avoid the reduction to ROLLO-I described in [LP23]. To this end, the new system patch introduces an auxiliary polynomial $P_N$ of small degree and modifies the $P_P$-part of the public key. This technique successfully reduces the number of linear equations that the attack by Lange and Pellegrini sets up.

The following text uses notation as in [LP23], please see there for definitions. The values $(q, n_1, n_2, n_I, m, r, d)$, where $n_I < n_1 < n_2$ are the system parameters. There is also a primitive polynomial $P_2$ of degree $n_2$ which is a system parameter.

The updated key generation procedure of the new system works as follows.

KeyGen:

- Pick random $\mathbf{x}, \mathbf{y} \in S_d^{n_1}(\mathbb{F}_{q^m})$.
- Pick random primitive $P_1 \in \mathbb{F}_{q^m}[x]$ of degree $n_1$.
- Pick random $P_I \in \mathbb{F}_{q^m}[x]/(P_1)$ of degree $n_I$.

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

**Remark 1.1** *Note that in the new specification $P_1$ is claimed to be part of the secret key, while in the reference implementation it is still a* ***fixed*** *polynomial. See* `Layered_ROLLO_Gen-0922/src/schemes/biix/biix_kem.c`*, lines 57-60.*

As described in the reference implementation of the new patch, precisely in line 502 of `Layered_ROLLO_Gen-0922/src/schemes/biix/biix_kem.c` the error vectors are represented by an $\mathbb{F}_{q^m}$-tuple of $\ell := n_2 - n_1 - n_I - n_N - 2$ elements (coefficients vectors). The error vectors can equivalently be represented as polynomials in $\mathbb{F}_{q^m}[x]$ of degree at most $n_E = \ell - 1$.

**Remark 1.2** *When viewing a polynomial $P \in \mathbb{F}_{q^m}[x]$ of degree at most $n$ as an element of $v \in \mathbb{F}_{q^m}^{n+1}$ corresponding to its coefficient vector, we consider the entries of $v$ to be ordered in a way such that $P = \sum_{i=0}^{n} v_i x^i$.*

The updated encapsulation mechanism with updated error-weights works as follows.

Encap($\mathsf{pk}$):

- 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 $n_E < n_2 - n_1 - n_I - n_N - 2$.
- 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 $\mathbf{c}$.

The decapsulation procedure has not been updated, we include it only for completeness.

Decap($\mathsf{sk}$):

- 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 $K = \mathsf{hash}(E)$.

Finally, the suggested parameters for the new modified Layered-ROLLO-I cryptosystem are

| Security parameter | $(q, m, n_I, n_1, n_2, n_N)$ |
|---|---|
| 128 | $(2, 67, 4, 37, 61, 1)$ |
| 192 | $(2, 79, 4, 43, 71, 2)$ |
| 256 | $(2, 97, 4, 53, 103, 4)$ |

Table 1: Suggested parameters for the new version.

## 2 The message recovery attack

This section describes the message recovery attack that we mounted against the Layered-ROLLO-I cryptosystem. The idea is to reduce the problem of solving the modular equation introduced in the encapsulation method into an instance of information set decoding in the Hamming metric with errors confined to a small subset of the positions.

Recall that encapsulation computes the ciphertext

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

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

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

where $R := P_P P_H^{-1} \bmod P_2$. View equation (1) in terms of $\mathbb{F}_{q^m}$ vectors corresponding to the coefficient vectors of the polynomials involved. As in [LP23], we can regard $R$ as the $(n_E + 1) \times n_2$ matrix $G$ over $\mathbb{F}_{q^m}$ representing the multiplication of a polynomial of degree up to $n_E$ by $R$ modulo $P_2$, i.e.

$$G = \begin{pmatrix} R \bmod P_2 \\ Rx \bmod P_2 \\ \vdots \\ Rx^{n_E} \bmod P_2 \end{pmatrix},$$

where each row consists of the coefficient vector of $Rx^i \bmod P_2$ for $i = 0, \ldots, n_E$.

Since multiplication by $R$ modulo $P_2$ represents an automorphism of the field $\mathbb{F}_{q^m}[x]/(P_2)$ (recall that $P_2$ is irreducible), $G$ has full rank. In other words $G$ generates a linear $[n_2, n_E + 1]$-code over $\mathbb{F}_{q^m}$.

With this in mind we can rewrite (1) as

$$\bar{\mathbf{c}} = \mathbf{e}_1 G + \mathbf{e}_2, \tag{2}$$

which corresponds to a McEliece-like encryption of the message $\mathbf{e}_1$. Due to the low degree of the polynomial representing it, $\mathbf{e}_2$ is also a relatively low Hamming weight error vector. Therefore we can now employ Prange's algorithm [Pra62] to simultaneously recover $\mathbf{e}_1$ and $\mathbf{e}_2$.

In the settings of all the versions of Layered-ROLLO-I, we can actually exploit some extra information about the error vector $\mathbf{e}_2$ in order to improve

over the plain Prange's algorithm. Indeed, the core of Prange's algorithm, lies in finding an invertible submatrix of $G$ that consists of a subset of columns corresponding to error-free positions in the ciphertext. By remark 1.2 the error vector $\mathbf{e}_2$ has non-zero entries only in the first $n_E + 1$ coordinates. Therefore, we can search for an invertible submatrix of $G$ in its last $n_2 - n_E + 1$ columns. Picking $n_E + 1$ random columns of a rank $n_E + 1$ matrix over $\mathbb{F}_{q^m}$, where $q$ and $m$ are given by the suggested parameters, will constitute an invertible matrix with overwhelming probability. We can also just take the last $n_E + 1$ columns.

Let $G_{\mathsf{inv}}$ be such a matrix. The last step of Prange's algorithm is to compute $\mathbf{e}_1 = \bar{\mathbf{c}}' G_{\mathsf{inv}}^{-1}$, where $\bar{\mathbf{c}}'$ consists of the coordinates of $\bar{\mathbf{c}}$ corresponding to the columns of $G_{\mathsf{inv}}$. Finally, compute $\mathbf{e}_2 = \bar{\mathbf{c}}_{n_E+1} - \mathbf{e}_1 G_{n_E+1}$, where $\bar{\mathbf{c}}_{n_E+1}$ and $G_{n_E+1}$ contain the first $n_E + 1$ positions of the $n_2$ total positions.

We implemented a non-optimized version of our attack in SageMath. An average of the time required, on a Linux Mint virtual machine, to recover the plaintext for the proposed parameters is given in the following table. It is worth noting that in our experiments we always used error vectors of the maximum allowed Hamming weight in order to simulate the worst case scenario for our attack.

| Security level | max degree of $\mathbf{e}_1$ and $\mathbf{e}_2$ | Time (s) |
|---|---|---|
| 128 | 17 | 2.21 |
| 192 | 19 | 3.18 |
| 256 | 39 | 6.65 |

Table 2: Average time in seconds (on 50 samples for each security level) needed to recover a plaintext.

The most time consuming steps of our attack are computing the matrix $G$ and $G_{\mathsf{inv}}^{-1}$. If an attacker breaks many ciphertexts for the same public key this computation is done only once.

Due to its simplicity, we are confident that the attack decrypts faster than the actual $\mathsf{RSR}$ decoding algorithm used in Layered-ROLLO-I. Both, the attack and the regular decapsulation procedure, use polynomial division modulo $P_2$. The matrix-vector multiplications in the attack are no more expensive than the polynomial inversion and multiplication modulo $P_1$ in the regular decapsulation procedure. Finally, RSR is very likely to take longer than the steps to compute $G$ and $G_{\mathsf{inv}}^{-1}$.

**Remark 2.1** *While this note presents the message-recovery attack on the latest version of Layered ROLLO-I we would like to point out that it worked for the two previous versions as well. Even for the November 2022 submission the degrees of* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *were smaller than half of* $n_2$*, which is relevant for the positions in* $G_{\mathsf{inv}}$ *not to overlap with the positions in* $\mathbf{e}_2$*. However, the subsequent modifications have decreased the degrees* $n_E$ *of the error terms and thus made the attack faster.*

# References


[KKN23] Chanki Kim, Young-Sik Kim, and Jong-Seon No. Comments and modification on Layered ROLLO on kPQC-forum. Slides attached to reply on Kpqc Bulletin, 2023.

[LP23] Tanja Lange and Alex Pellegrini. Analysis of Layered ROLLO-I. Email to KpqC Bulletin, 2023.

[Pra62] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.