A Constant-time AVX2 Implementation of a Variant of ROLLO

Tung Chou, Jin-Han Liou

Academia Sinica

September 15, 2022

ROLLO as a NISTPQC candidate

- A code-based KEM but in "**rank-metric**"
- Did not enter the 3rd round because of new algebraic attacks.
  - See "*Improvements of algebraic attacks for solving the rank decoding and minrank problem*", Asiacrypt 2020
- The `ROLLO` team proposed larger parameter sets.
- "NTRU-like" as `BIKE`.
- Small public keys due to the ring structure.
- No fast constant-time software.

Construction

- The ring is $\mathbb{F}_{2^{mn}}$.
- **Support**: the $\mathbb{F}_2$-linear subspace spanned by the coefficients in $\mathbb{F}_{2^m}$.
- **Rank weight**: dimension of support.
- A secret key is of the form $(h_0, h_1) \in \mathbb{F}_{2^{mn}}^2$. Each h_i is of low rank weight.
- The public key is $h = h_1/h_0 \in \mathbb{F}_{2^{mn}}$.
- Encapsulation: $c = e_0 + h \cdot e_1 \in \mathbb{F}_{2^{mn}}$ with low-weight e_i's.
- Decapsulation: Compute $h_0 c = h_0 e_0 + h_1 e_1$. Use $\mathtt{RSR}$ to compute Support(e_0, e_1).

Main optimizations techniques

1. Using `RSR+` instead if `RSR`
2. Fast constant-time generation of and multiplications by low-weight elements in $\mathbb{F}_{2^{mn}}$
3. Fast constant-time Zassenhaus algorithm:
   - Gaussian elimination
   - Retrieval of intersection

- The techniques can be used for ROLLO also.
- Similar techniques can be used for embedded systems.

RSR versus RSR+

Algorithm 2 Rank Support Recover (RSR) algorithm

Parameters: $m, n, d, r \in \mathbb{Z}$.

Input: An $\mathbb{F}_2$-subspace F of dimension d in $\mathbb{F}_{2^m}$, represented as $(f_1, \ldots, f_d) \in \mathbb{F}_{2^m}^d$ such that $F = \langle f_1, \ldots, f_d \rangle$, and $s = \sum_{i=0}^{n-1} s_i y^i \in \mathbb{F}_{2^{mn}}$.

Output: An $\mathbb{F}_2$-subspace E of $\mathbb{F}_{2^m}$.

```
Compute S = ⟨s_0, ..., s_{n-1}⟩.
E ← ⋂_{i=1}^{d} f_i^{-1} S
return E
```

Algorithm 3 RSR$^+$ algorithm

Parameters: $m, n, d, r \in \mathbb{Z}$.

Input: An $\mathbb{F}_2$-subspace F of dimension d in $\mathbb{F}_{2^m}$, represented as $(f_1, \ldots, f_d) \in \mathbb{F}_{2^m}^d$ such that $F = \langle f_1, \ldots, f_d \rangle$, and $s = \sum_{i=0}^{n-1} s_i y^i \in \mathbb{F}_{2^{mn}}$.

Output: An $\mathbb{F}_2$-subspace E of $\mathbb{F}_{2^m}$ or $\perp$.

```
Compute S = ⟨s_0, ..., s_{n-1}⟩.
E ← ⋂_{i=1}^{d} f_i^{-1} S
if dim(S) ⩽ dr then
    return E
else
    return ⊥
end if
```

Field multiplications and generation of low-weight elements

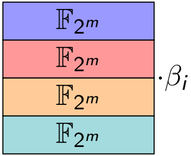
- We often need to perform multiplications with "low-weight" elements

$$\alpha_1\beta_1 + \alpha_2\beta_2 + \cdots + \alpha_d\beta_d \in \mathbb{F}_{2^{mn}},$$

  where $\alpha_i \in \mathbb{F}_{2^n}$ and $\beta_i \in \mathbb{F}_{2^m}$.

- Generating low-weight elements $\rightarrow$ generating α_i's and β_i's and check the ranks by Gaussian elimination.

- We deal with β_i's first and then α_i's using matrix transposition.

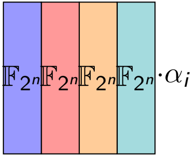


Field multiplications and generation of low-weight elements

- We often need to perform multiplications with "low-weight" elements
$$\alpha_1\beta_1 + \alpha_2\beta_2 + \cdots + \alpha_d\beta_d \in \mathbb{F}_{2^{mn}},$$
where $\alpha_i \in \mathbb{F}_{2^n}$ and $\beta_i \in \mathbb{F}_{2^m}$.

- Generating low-weight elements $\rightarrow$ generating α_i's and β_i's and check the ranks by Gaussian elimination.

- We deal with β_i's first and then α_i's using matrix transposition.



Computing intersection of linear subspaces

- The Zassenhaus algorithm: given matrices U and V, compute

$$\mathrm{RowSpace}(U) \cap \mathrm{RowSpace}(V)$$

- Compute

$$Z = \begin{pmatrix} U & U \\ V & 0 \end{pmatrix}$$

- Row reduce Z to

$$Z' = \begin{pmatrix} A & C \\ 0 & B \\ 0 & 0 \end{pmatrix}$$

- Then the intersection is $\mathrm{RowSpace}(B)$.
- How to carry out Gaussian elimination? How to extract B?

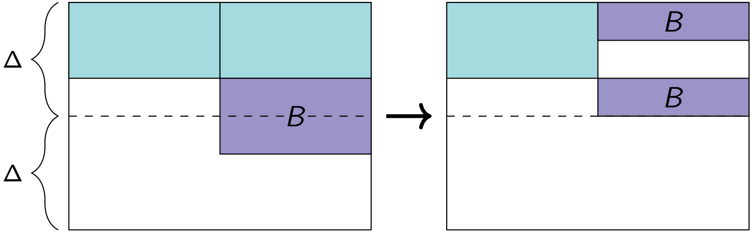
Gaussian elimination

Input: $A \in \mathbb{F}_2^{\mu \times \nu}$.

1. Set $p = 1$.
2. Set v to the logical OR of $A_p, \ldots, A_\mu$.
3. Find the index j of the first nonzero entry in v. If $v = 0$, set j to any value in $\{1, \ldots, \nu\}$.
4. For $i \in \{p+1, \ldots \mu\}$, $A_p \leftarrow A_p + A_i \cdot (1 - A_{p,j})$.
5. For $i \in \{1, \ldots, \mu\} \setminus \{p\}$, $A_i \leftarrow A_i + A_p \cdot A_{i,j}$.
6. If $p + 1 \le \min(\mu, \nu)$, increase p by 1 and go back to Step 2.

Made constant-time with intrinsics such as `_mm256_movemask_epi8`, `_tzcnt_u64` and `_mm256_permutevar8x32_epi32`.

Retrieval of the intersection



Let $Z^{(L)}, Z^{(R)}$ be the left, right parts of the matrix.

1. if $Z_i^{(L)} \neq 0$ and $Z_{i+\Delta}^{(L)} = 0$, set $Z_i^{(R)}$ to $Z_{i+\Delta}^{(R)}$.
2. if $Z_i^{(L)} \neq 0$ and $Z_{i+\Delta}^{(L)} \neq 0$, set $Z_i^{(R)}$ to 0.

Performance and notes

instance	key gen.	encap.	decap	level	reference
ROLLO-I-128	11034623	984432	9775241	1	Aguilar-Melchor et al.
	11204649	320835	9744693		
ROLLO^{+}-I-128	851823	30361	673666	1	this paper
ROLLO^{+}-I-192	980860	38748	878398	3	
ROLLO^{+}-I-256	1477519	55353	1635966	5	
ROLLO^{+}-II-128	4663096	70621	876533	1	this paper
ROLLO^{+}-II-192	4058419	94138	1060271	3	
ROLLO^{+}-II-256	4947630	90021	1497315	5	
`bikel1`	589625	114256	1643551	1	Chen-Chou-Krausz
`bikel3`	1668511	267644	5128078	3	

Table: Cycle counts on Intel Skylake and Coffe Lake