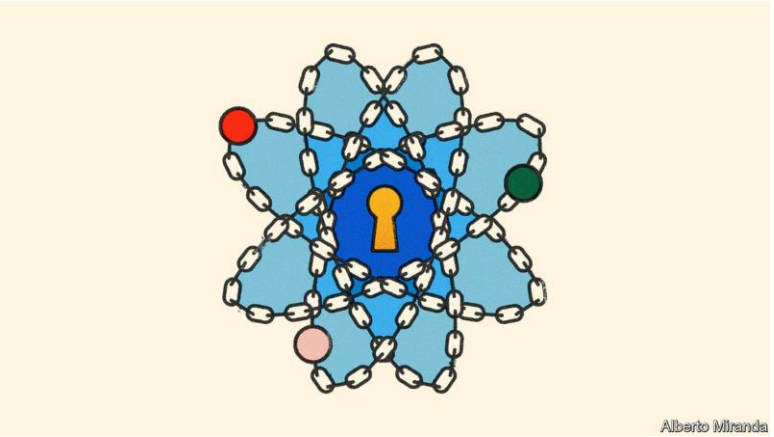
2023 kPQC Winter Camp
02.22.2023 13:30-14:30
중앙대학교 파이퍼홀(103관) 106호

Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes

Chanki Kim
Chosun University
Dept. of Information and Communicaton

Qunatum computing

- Quantum computing (QC): Exploits quantum characteristics such as *irreversibility*, *superposition*, and *entanglement* as computing processes
- For QC, some algorithms can overcome the computational limits of classical computers, which have emerged as NP problems.
- New applications leveraging QC are being developed: large-scale science computing optimization, quantum machine learning



[1] The economist, "How to preserve secrets in a quantum age," Jul. 2022, Available in [online] https://www.economist.com/science-and-technology/2022/07/13/how-to-preserve-secrets-in-a-quantum-age

Security Issues on QC Era

- In the field of security, difficulties in solving certain mathematical problems in a classical computing have been used as the main logic to gaurantee the security of cryptography schemes.
- In the advent of QC, new algorithms are proposed that threaten their security
  - For RSA ciphers in theasymmetric cryptography, a number of prime factorization problems that ensure the security, are broken by the Shor algorithm of quantum computers.
  - For AES ciphers, Grover's algorithm can be leveraged to reduce the complexity of the attack

Table 1 | Examples of widely deployed cryptographic systems and their conjectured security levels

Name	Function	Pre-quantum security level	Post-quantum security level
Symmetric cryptography			
AES-128[8]	Symmetric encryption	128	64 (Grover)
AES-256[8]	Symmetric encryption	256	128 (Grover)
Salsa20[58]	Symmetric encryption	256	128 (Grover)
GMAC[59]	MAC	128	128 (no impact)
Poly1305[60]	MAC	128	128 (no impact)
SHA-256[61]	Hash function	256	128 (Grover)
SHA3-256[62]	Hash function	256	128 (Grover)
Public-key cryptography			
RSA-3072[1]	Encryption	128	Broken (Shor)
RSA-3072[1]	Signature	128	Broken (Shor)
DH-3072[42]	Key exchange	128	Broken (Shor)
DSA-3072[63,64]	Signature	128	Broken (Shor)
256-bit ECDH[4–6]	Key exchange	128	Broken (Shor)
256-bit ECDSA[66,67]	Signature	128	Broken (Shor)

Security levels shown are against the best pre-quantum and post-quantum attacks known. Security level *b* means that the best attacks use approximately 2^b operations. This optimization ignores parallelization requirements; see text for discussion of the impact of such requirements. For hash functions, 'security' in this table refers to pre-image security.

[2]

[2] Bernstein and T. Lange, ``Post-quantum cryptography,'' *Nature,* vol. 549, pp. 188-194, 2017

Post quantum Cryptogrpahy

- The US National Institute of Standards and Technology (NIST) predicts that existing cryptosystems will not be sufficiently secure for quantum computer algorithms
- NIST held a contest to standardize quantum resistant cryptography that can replace existing cryptographic systems, and the following algorithms have been selected in the current 3rd round
  - Key encapsulation mechanism: Classic McElicece, Crystal-kyber, NTRU
  - Signiture: CRYSTALS-DILITHIUM, FALCON, Rainbow
- Many countries including Soth Korea are also triying to develop new cryptosystems for post quantum computing era for cryptography contests.

Types of post-quantum cryptography

- It is classified into the following three types according to the mathematical design and hardness problem that are the basis of security.
  - Lattice-based cryptostystem
  - **Code-based cryptosystem**
  - Multivariate/hash-based cryptosystem

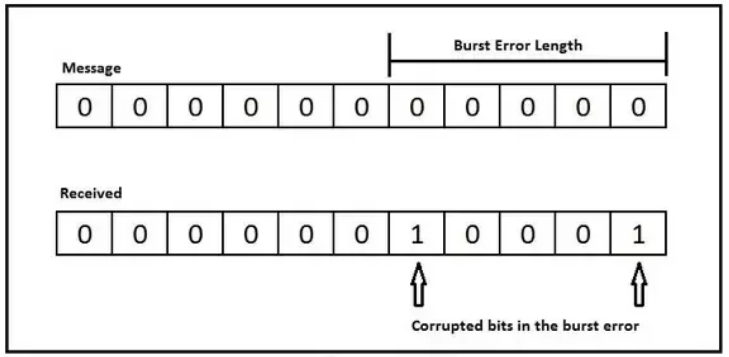
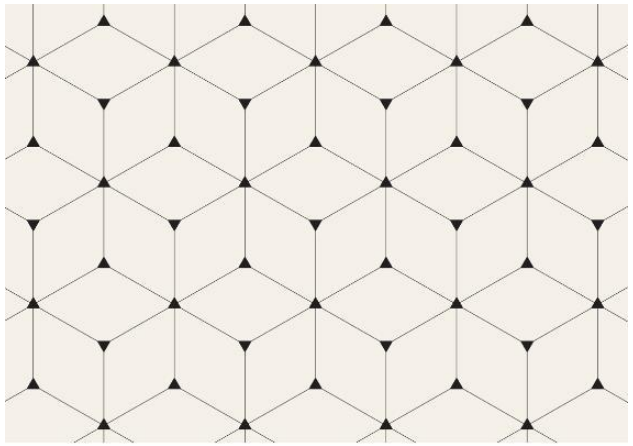


Figure 1

- The hardness problem, which is the basis of the security of quantum resistant cryptography, has not yet been fully verified
- Research on cryptographic systems with various structures is necessary to prepare for the risk of specific schemes being attacked



Code-based PQC

- Most of the code-based cryptography is based on security from the problem that it is difficult to guess the codeword from the syndrome value proposed by McEliece

- According to the code type, the following cryptosystem has been proposed
  - Classic McEliece: PKE/KEM, Goppa code based, NIST finalist
  - HQC: PKE/KEM, Reed-Solomon code based, NIST candidate
  - BIKE: PKE/KEM, QC-MDPC code based, NIST candidate
  - **ROLLO: PKE/KEM, Rank-metric code based, NIST 2 Round**
  - pqsigRM: Signiture, Reed-Muller code based , NIST 1 Round

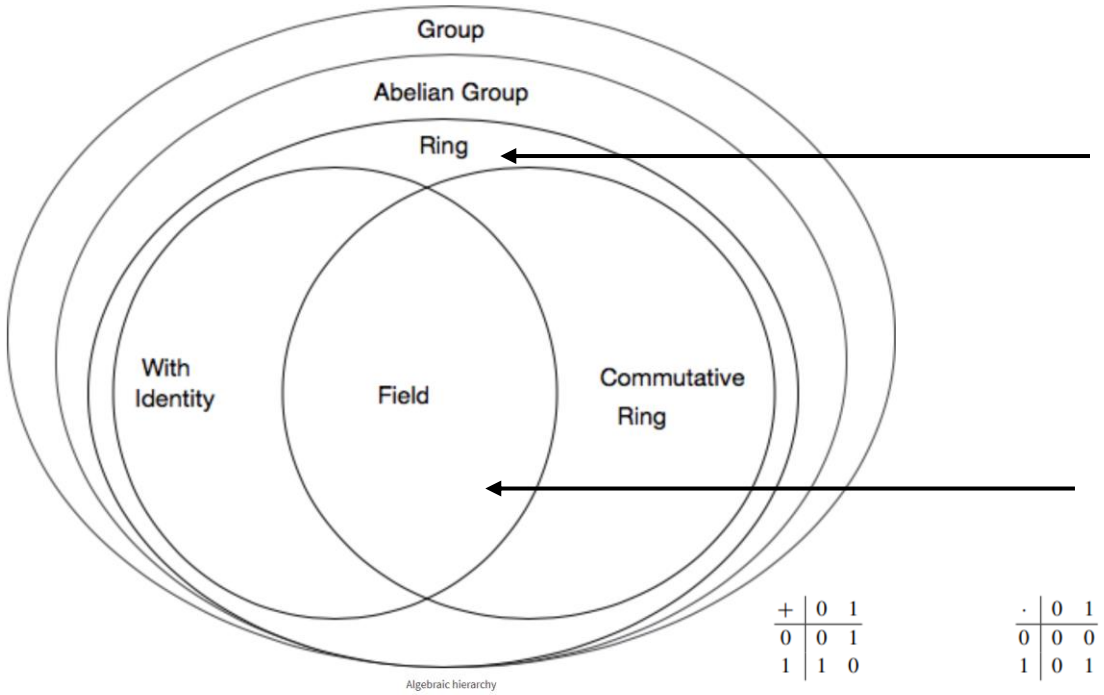
BIKE

ROLLO -
Rank-Ouroboros, LAKE
& LOCKER

April 21, 2020

Preliminaries: Finite field

- For the design of error correcting codes, concepts of group, ring, field are widely used
- Understanding of (abstract) algebra is essential



Polynomial ring
Primitive polynomial

Field extension

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

Preliminaries: Polynomial Ring Arithmetic

- **Polynomial (quotient) ring** $\mathbb{F}_{q^m}[X]/< P >$
  - For positive integer $n, m, \text{and } q$
  - Suppose a n-degree (primitive) polynomial P over binary field,
  - From n-length vector $\mathbf{p}' = (p_1, \dots p_n)$, where p_i are elements of field $\mathbb{F}_{q^m}$ we have the maps Φ and Φ^{-1} satisfying

$$\mathbf{p}' = (p_1, \dots p_n) \quad \underset{\Phi^{-1}}{\overset{\Phi}{\rightleftarrows}} \quad P'(X) = \sum_{i=0}^{n-1} p_i \bmod P \in \mathbb{F}_{q^m}^{n}[X]/< P >$$

Preliminaries: Polynomial Ring Arithmetic

- **Polynomial ring notations for two different moduli**
  - We use P and P^b for the moduli of polynomial moduli, where P is a $\frac{n}{b}$-degree primitive polynomial for positive integer b. Then, we have

$$\mathbf{p}_1 = (p_1, \ldots, p_{\frac{n}{b}}) \quad \underset{\Phi_1^{-1}}{\overset{\Phi_1}{\rightleftarrows}} \quad P_1(X) = \sum_{i=0}^{\frac{n}{b}-1} p_i \bmod P \in \mathbb{F}_{q^m}[X]/< P >$$

$$\mathbf{p}_2 = (p_1, \ldots, p_n) \quad \underset{\Phi_2^{-1}}{\overset{\Phi_2}{\rightleftarrows}} \quad P_2(X) = \sum_{i=0}^{n-1} p_i \bmod P^b \in \mathbb{F}_{q^m}[X]/< P^b >$$

Preliminaries: Polynomial Ring Arithmetic

- **Multiplication notations**
  - For vectors $\mathbf{u}, \mathbf{u}' \in \mathbb{F}_{q^m}^{\frac{n}{b}}$, polynomial $P_1 \in \mathbb{F}_{q^m}[X]/< P >$, and map Φ_1, we have

$$\mathbf{u}\mathbf{u}' \bmod \mathrm{P} = \Phi_1^{-1}(\Phi_1(\mathbf{u})\Phi_1(\mathbf{u}')) = \sum_{i=0}^{\frac{n}{b}-1} \sum_{j=0}^{i} u_{i-j} u'_j (X_1)^i mod\ P$$

Vector-vector multiplication

$$P_1\mathbf{u}' \bmod \mathrm{P} = \Phi_1^{-1}(P_1\Phi_1(\mathbf{u}')) = \sum_{i=0}^{\frac{n}{b}-1} P_1 u_i (X_1)^i mod\ P$$

Polynomial-vector multiplication

Preliminaries: Polynomial Ring Arithmetic

- **Conversion maps between two moduli**
  - For the conversion, we use the two maps Ψ and Ω asxc

$$P_1 = \sum_{i=0}^{\frac{n}{b}-1} p_i \bmod P \in \mathbb{F}_{q^m}[X]/< P >$$

$$\Omega(P_2) = \sum_{i=0}^{n-1} p_i \bmod P \in \mathbb{F}_{q^m}[X]/< P >$$

Ψ ↓

↑ Ω

$$\Psi(P_1) = \sum_{i=0}^{\frac{n}{b}-1} p_i \bmod P \in \mathbb{F}_{q^m}[X]/< P^b >$$

$$P_2 = \sum_{i=0}^{n-1} p_i \bmod P \in \mathbb{F}_{q^m}[X]/< P^b >$$

Preliminaries: Polynomial Ring Arithmetic

- **Equvalent circulent matrix conversion**
  - n-tuple vector $\mathbf{v}$(or polynomial $\mathbf{v}$(X)) can be converted into the circulant matrix
  - The vector from vector-vector multiplication $\mathbf{u}\mathbf{u}'\bmod P$ is equal to that from matrix-vector multiplication of $\mathbf{u}\mathcal{I}(\mathbf{u}')$ or$\mathcal{I}(\mathbf{u})\mathbf{u}'$ (Ideal condition)

$$\mathbf{v} = (v_1, v_2, \ldots, v_n) \quad \underset{\mathcal{I}^{-1}}{\overset{\mathcal{I}}{\rightleftarrows}} \quad \mathcal{I}(\mathbf{v}) = \begin{pmatrix} \mathbf{v} \\ X\mathbf{v} \bmod P \\ \ldots \\ X^{n-1}\mathbf{v} \bmod P \end{pmatrix}^{\mathsf{T}}$$

Preliminaries: Polynomial Ring Arithmetic

- **Some useful properties: Proposition 1**
  - If $\sum_{i\in[I]} \deg\left(\mathbf{u}_i(X_1)\right) < n$ for integer set I, we have

$$\prod_{i\in[I]} \mathbf{u}_i(X_1) \bmod P = \Omega\left(\prod_{i\in[I]} \Psi\left(\mathbf{u}_i(X_1)\right) \bmod P^b\right)$$

- Proof sketch) From $\prod_{i\in[I]} \Psi\left(\mathbf{u}_i(X_1)\right) \bmod P^b$, the polynomial reduction is not occurred by P^b if $\sum_{i\in[I]} \deg\left(\mathbf{u}_i(X_1)\right) < n$. Therefore, it returns the same results

Preliminaries: Hamming metric and codes

- **Hamming weight** of vector $\mathrm{wt}_H(\mathbf{v})$ or $|\mathbf{v}|$: Defined as the number of nonzero elements for n-tuple vector $\mathbf{v} = (v_1, v_2, \dots, v_n) \in \left(\mathbb{F}_{q^m}\right)^n$
- (n, k) $\mathbb{F}_{q^m}$ **–linear codes:** $\mathbb{F}_{q^m}$ –linear code C is defined by subset with q^k-cardinality consisting of n-tuple vectors $\mathbf{c}_i$ in $C = \left\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{q^k}\right\} \subset \left(\mathbb{F}_{q^m}\right)^n$
- **Minimum Hamming weight** $\mathrm{wt}_H(C)$ of $\mathbb{F}_{q^m}$ –linear code C: For a codeword $\mathbf{c}_i$, $\mathrm{wt}_H(C) = \min_{\mathbf{c}_i \subset C} \mathrm{wt}(\mathbf{c}_i)$
- (n, k, d) $\mathbb{F}_{q^m}$ **–linear codes:** (n, k, d) $\mathbb{F}_{q^m}$ –linear code C is defined by (n, k) $\mathbb{F}_{q^m}$ –linear codes with minimum Hamming weight $\mathrm{w}_H(C) = d$

Preliminaries: Hamming metric and codes

- From a codeword $\mathbf{c}$ in (n, k, d) $\mathbb{F}_{q^m}$ −linear code C, additive error vector $\mathbf{e}$ with $||\mathbf{e}|| < \left\lfloor \frac{d-1}{2} \right\rfloor$ can be **corrected** from a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$ by sphere packing lemma

Definition 2.21 The *minimum Hamming distance* (or *minimum distance*) $d_{\min}$ of a code C is the minimum of the distances between all pairs of codewords. □

t: Error correction capability
d: Error detection capability

$$2t + d + 1 \leq d_{min}$$

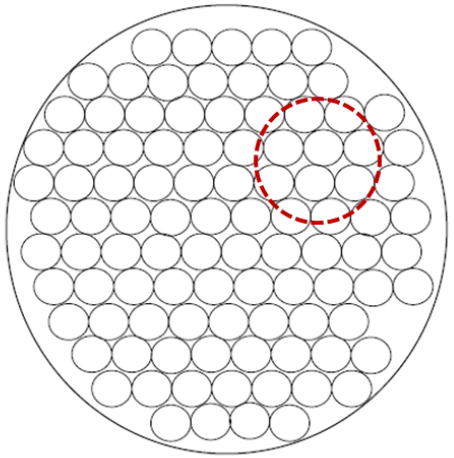


FIGURE 9.2. Sphere packing for the Gaussian channel.

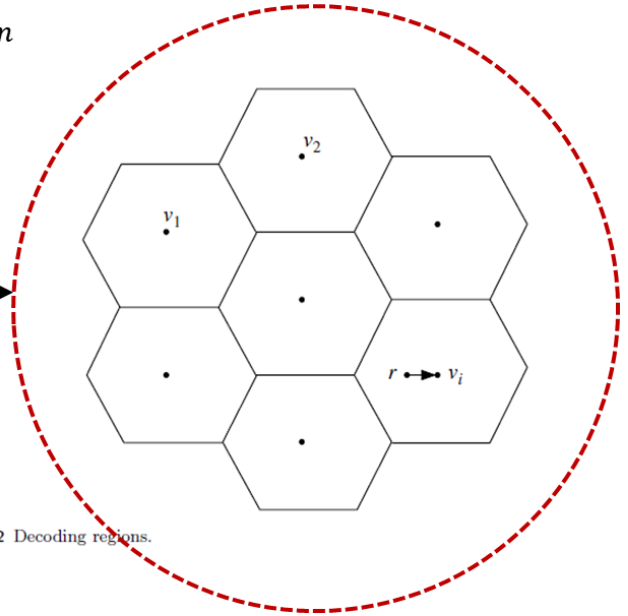


Figure 3.2 Decoding regions.

Preliminaries: Rank metric representation

- From an n-length vector $\mathbf{v} = (v_1, v_2, \dots, v_n) \in \left(\mathbb{F}_{q^m}\right)^n$
  - Let β be an element of m-degree primitive polynomial satisfying

$$v_i = \sum_{j=0}^{m-1} v_{ij}\beta^j$$

  - Then, the vector $\mathbf{v}$ can be converted into the matrix $M(\mathbf{v}) \in \left(\mathbb{F}_{q^m}\right)^n$
  - Then, the **rank weight** of vector $\mathrm{wt}_R(\mathbf{v})$ or $||\mathbf{v}||$ is defined as a rank of the matrix $M(\mathbf{v})$

$$\mathbf{v} = (v_1, v_2, \dots, v_n) \xrightarrow{M(\cdot)} M(\mathbf{v}) = \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{n1} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1m} & v_{2m} & \cdots & v_{nm} \end{bmatrix}$$

Preliminaries: Rank metric codes

- **Minimum rank weight** $\mathrm{w}t_R(C)$ of $\mathbb{F}_{q^m}$ −linear code C: For a codeword $\mathbf{c}_i$, $\mathrm{w}_R(C) = \min_{\mathbf{c}_i \subset C} wt(\mathbf{c}_i)$

- (n,k) $\mathbb{F}_{q^m}$ **−linear codes with rank weight** d**:** (n,k) $\mathbb{F}_{q^m}$ −linear code C is defined by (n,k) $\mathbb{F}_{q^m}$ −linear codes with minimum rank weight $\mathrm{w}_r(C) = d$

- From a codeword $\mathbf{c}$ in (n,k) $\mathbb{F}_{q^m}$ −linear code C **with rank weight** d, additive error vector $\mathbf{e}$ with $||\mathbf{e}|| < \left\lfloor \frac{d-1}{2} \right\rfloor$ can be **corrected** from a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$ by sphere packing lemma

Goodness of rank weight code

- Correcting low-rank error can be advantageous for correcting the correlated multiple errors

Error vector $\mathbf{e}$ and $M(\mathbf{e})$

1	0	…	0
1	0	…	0
…	…	…	…
1	0	…	0

- Suppose that a codeword has a 2-D placement in
- Column-wise errors are occurred

From Hamming weight metric

$|\mathbf{e}|$ = column length
Multi-error correctable codes should be used

From rank weight metric

$||\mathbf{e}|| = 1$
Single-error correctable codes are sufficient

Hamming metric and rank metric codes

- **Properties**
  - Product (sub)space and Hamming/rank weight
- **Code design**
  - ➢ Hamming metric codes: maximum distance separable (MDS) codes, moderate-density parity check (MDPC) code, low-density parity-check (LDPC) codes
  - ➢ Rank metric codes: Maximum rank distance (MRD) codes, Low-rank parity-check (LRPC) codes
- **Hardness problem**
  - ➢ Hamming metric codes: (Hamming) Syndrome decoding problem
  - ➢ Rank metric codes: Rank syndrome decoding (RSD) problem
- **Cryptosystem**
  - ➢ Hamming metric codes: HQC(MDS code based), BIKE (LDPC code based)
  - ➢ Rank metric codes: RQC(MRD code based), ROLLO (LRPC code based)

Product subspace and rank weight

- **For the properties of Hamming metric code**
  - $\mathbb{F}_{q^m}$ **−subspace(**$\sim \mathbb{F}_{q^m} - \text{linear code}$**)**
  - **Product subspace**
    - Generally, For two $\mathbb{F}_{q^m}$ −spaces(or linear codes) C_1 and C_2, product space $C_1C_2 = C_1 \cap C_2$ can be generated
    - For the $\mathbb{F}_{q^m}$ −subspace C_1, C_2 with Hamming weight r and d, Hamming weight of C_1C_2 is lower than $\leq rd$ (If Hamming weight r and d is low, most probably, the weight of C_1, C_2 equal to rd**)**

- **For the properties of rank metric code**
  - $\mathbb{F}_q$**-Subspace**
    - Support(generating) set $\text{supp}(\mathbf{v}) =< v_1, \dots, v_n >$ for each element of $\mathbf{v} = (v_1, v_2, \dots, v_n) \in \left(\mathbb{F}_{q^m}\right)^n$
  - **Product subspace**
    - If rank weights weight r and d is low, most probably, the weight of EF equal to rd
    - For two $\mathbb{F}_q$ −subspaces $E = \text{supp}(\mathbf{u}) =< u_1, \dots, u_n >$ and $F = \text{supp}(\mathbf{v}) =< v_1, \dots, v_n >$, product space can be represented as $EF = \text{supp}(\mathbf{uv} \bmod P)$

Hamming metric and rank metric codes

- **For Hamming metric MDS codes:**
  - MDS codes achieves the Singleton bound: $d \leq n - k + 1$
  - Reed-Solomon codes is MDS codes with PCM $\mathbf{A}$ with Vandermonde matrix
  - RS codes are used for PQC scheme HQC (NIST Candidate, PKE/KEMs)

$$A = \begin{bmatrix} 1 & \cdots & 1 & \cdots & 1 \\ a_1 & \cdots & a_k & \cdots & a_n \\ a_1^2 & \cdots & a_k^2 & \cdots & a_n^2 \\ \vdots & & \vdots & & \vdots \\ a_1^{k-1} & \cdots & a_k^{k-1} & \cdots & a_n^{k-1} \end{bmatrix}$$

- **For rank metric MRD codes**
  - MRD codes achieves the Singleton bound: $d \leq \frac{m}{n}(n-k)+1$
  - Gabidulin code is MRD codes with generator matrix $\mathbf{G}$
  - Gabidulin codes are used for RQC(NIST Candidate, PKE/KEMs) (NIST 2 Round Submission, PKE/KEMs)

$$\mathbf{G} = \begin{pmatrix} g_1^{q^0} & g_2^{q^0} & \cdots & g_n^{q^0} \\ g_1^{q^1} & g_2^{q^1} & \cdots & g_n^{q^1} \\ \vdots & \vdots & \vdots & \vdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix}$$

Hamming metric and rank metric codes

- **For Hamming metric MDPC/LDPC codes(in BIKE)**
  - For codelength $2n$ $\mathbf{c} \in \mathcal{C}$, LDPC codes is binary linear codes with $n \times 2n$-sized PCM $\mathbf{H}$ using two circulant matrices $\mathbf{H}_1$ and $\mathbf{H}_2$ of

$$\mathbf{H} = (\mathbf{H}_1 | \mathbf{H}_2),\ \mathbf{H}_1 = \begin{pmatrix} \mathbf{h}_1 \\ X\mathbf{h}_1 \bmod (X^n - 1) \\ \dots \\ X^{n-1}\mathbf{h}_1 \bmod (X^n - 1) \end{pmatrix}^{\mathsf{T}},$$

$$\mathbf{H}_2 = \begin{pmatrix} \mathbf{h}_2 \\ X\mathbf{h}_2 \bmod (X^n - 1) \\ \dots \\ X^{n-1}\mathbf{h}_2 \bmod (X^n - 1) \end{pmatrix}^{\mathsf{T}}$$

,where n-tuple random two vectors $\mathbf{h}_1$ and $\mathbf{h}_2$ satisfying $|\mathbf{h}_1| \leq d$ and $|\mathbf{h}_2| \leq d$

Hamming metric and rank metric codes

- **For rank metric ideal LRPC codes(in ROLLO)**
  - For codelength $2n$ $\mathbf{c} \in \mathcal{C}$, Ideal LRPC codes is $\mathbb{F}_{q^m}$ –linear codes with $n \times 2n$-sized PCM $\mathbf{H}$

$$\mathbf{H} = (\mathbf{H}_1 | \mathbf{H}_2),\ \mathbf{H}_1 = \begin{pmatrix} \mathbf{x} \\ X\mathbf{x} \bmod P \\ \dots \\ X^{n-1}\mathbf{x} \bmod P \end{pmatrix}^{\mathsf{T}}, \mathbf{H}_2 = \begin{pmatrix} \mathbf{y} \\ X\mathbf{y} \bmod P \\ \dots \\ X^{n-1}\mathbf{y} \bmod P \end{pmatrix}^{\mathsf{T}}$$

where $\mathbb{F}_q$-subspace F with rank weight d in $\mathbb{F}_{q^m}$, n-degree polynomial $P \in F_q[X]$, and n-tuple random two vectors $\mathbf{x}$ and $\mathbf{y}$ in F

  - Note that we have $\mathbf{H}_1^{-1} = \begin{pmatrix} \mathbf{x^{-1}} \\ X\mathbf{x^{-1}} \bmod P \\ \dots \\ X^{n-1}\mathbf{x^{-1}} \bmod P \end{pmatrix},\ \mathbf{H}_1^{-1}\mathbf{H}_2 = \begin{pmatrix} \mathbf{x^{-1}y} \\ X\mathbf{x^{-1}y} \bmod P \\ \dots \\ X^{n-1}\mathbf{x^{-1}y} \bmod P \end{pmatrix}^{\mathsf{T}}$

Hamming metric and rank metric codes

- **Hamming metric codes: Syndrome decoding problem (SD)**
  - For a syndrome vector $\mathbf{s}$, it is hard to find a vector $\mathbf{e}$ lower than the Hamming weight w, on the condition that $\mathbf{H}\boldsymbol{e}^{\boldsymbol{T}} = \mathbf{s}^T$
  - Known as NP-Complete

- **Rank metric codes: Rank syndrome decoding problem (RSD)**
  - For a syndrome vector $\mathbf{s}$, it is hard to find a vector $\mathbf{e}$ lower than the rank weight w, on the condition that $\mathbf{H}\boldsymbol{e}^{\boldsymbol{T}} = \mathbf{s}^T$
  - If RSD is in ZPP, the problem is NP(asymptotically, NP) [3]

- **Rank metric codes: Ideal-Rank syndrome decoding problem (I-RSD)**
  - For a vector $\mathbf{h}$ and syndrome vector $\mathbf{s}$, it is hard to find a vector $\mathbf{e} = (\mathbf{e_1}, \mathbf{e_2})$ lower than the rank weight w satisfying the condition that $\mathbf{e}_1 + \mathbf{e}_2\mathbf{h}\, mod\ P = \mathbf{s}$.

[3] P. Gaborit and G. Zemor, On the Hardness of the Decoding and the Minimum Distance Problems for Rank Codes," *IEEE Trans. Inf. Theo.,* vol. 62, No. 12, pp. 7245-7252, 2016

Indistinguishability

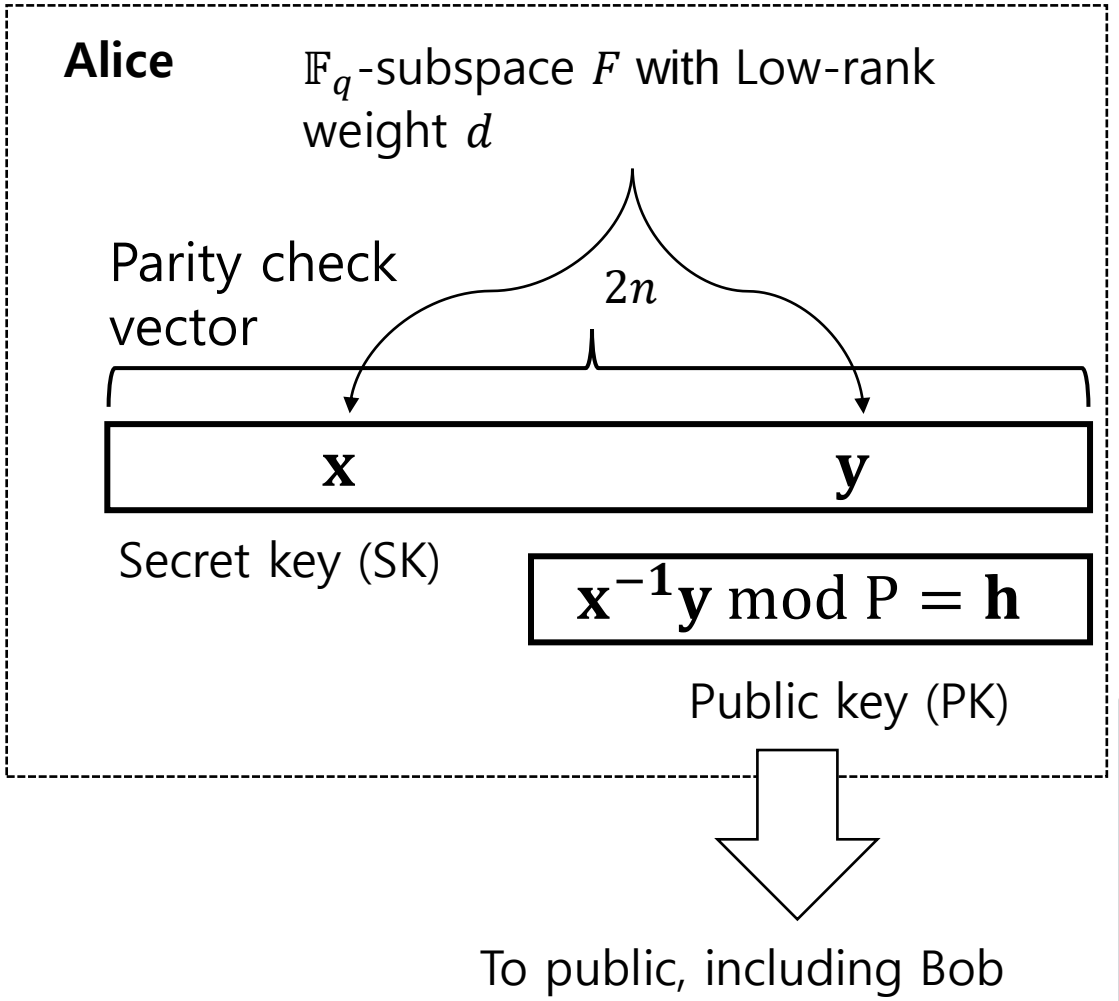
- **Indistinguishability**
  - It is difficult to distinguish or recover the structural characteristics of the codeword from the specific word generated by message and parity combination

- **For Hamming metric codes: Indistinguishability of MDPC/LDPC codes**
  - For the circulant matrix $\mathbf{H}_1$ and $\mathbf{H}_2$with small Hamming row weight d, it is hard to distinguish between the uniformly sampled random matrix R and $\boldsymbol{H}_1^{-1}\boldsymbol{H}_2$.

- **For rank metric codes: Indistinguishability of ideal LRPC codes**
  - For vectors $\mathbf{x}$ and $\mathbf{y}$ with small rank weight d, it is hard to distinguish between the uniformly sampled random vector $\mathbf{h}$ and $\mathbf{x}^{-1}\mathbf{y}$ mod P.
  - Application for the cryptosystem (ROLLO): $\mathbf{x}^{-1}\mathbf{y}$ mod P can be used for PK in a cryptosystem with a public key (PK) $\mathbf{x}^{-1}\mathbf{y}$ mod P.

Existing ROLLO Schemes

- Alice and Bob wants to have a **shared key.**
  For this,
  - **Key generation:** Alice generates public key (PK) and secret key (SK)
  - **Encapsulation:** Bob generate ciphertext (CT) to encapsulate SS using PK
  - **Decapsulation:** Alice decapsulate CT to obtain SS using SK



1. Key generation

- Generate Low-rank $\mathbb{F}_q$-subspace $F \in \left(\mathbb{F}_{q^m}\right)^n$ with rank weight d
- Select a random vector $\mathbf{x}, \mathbf{y} \in F$
- Generate PK and SK as
  - **PK:** $\mathbf{h} = \mathbf{x^{-1}y} \bmod \mathrm{P}$
  - **SK:** $< \mathbf{x}, \mathbf{y} >$

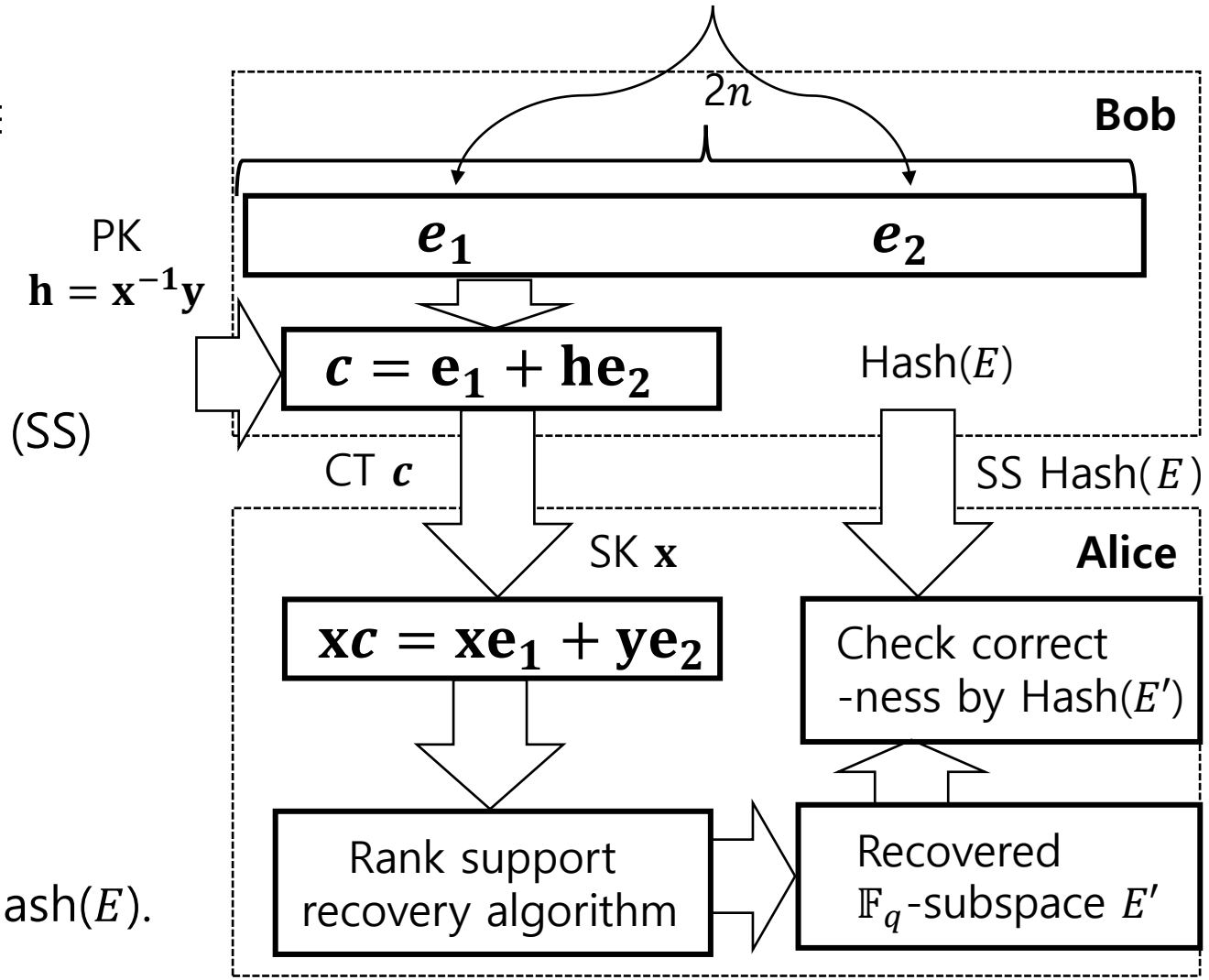
Existing ROLLO Schemes

2. Encapsulation

- Generate Low-rank $\mathbb{F}_q$-subspace $E \in (\mathbb{F}_{q^m})^n$ with rank weight r
- Generate n –tuple two error vector $\boldsymbol{e}_1, \boldsymbol{e}_2 \in E$
- Transmit $\boldsymbol{c} = \mathbf{e_1} + \mathbf{he_2}$ mod P
- Generate Hash(E) for shared secret (SS)

3. Decapsulation

- From SK $\mathbf{x}$ derive $\mathbf{x}\boldsymbol{c}$ mod $P = \mathbf{xe_1} + \mathbf{ye_2}$ mod P
- Recover $\mathbb{F}_q$-subspace E in the rank support recovery (RSR) algorithm
- Verify the correctness by comparing recovered has E' with received SS Hash(E).
- with Hash(E')=Hash(E)



RSR decoding algorithm

- **Sketch of the RSR algorithm**
  - Using $\mathbb{F}_q$-subspace F and SK $\mathbf{x}, \mathbf{y}$, recover the $\mathbb{F}_q$-subspace E from $\mathbf{xe_1} + \mathbf{ye_2}$ mod P
  - Main contribution
    - 1. Calculate $\mathbf{xc} = \mathbf{xe_1} + \mathbf{ye_2}$
    - 2. Derive $\mathbf{f}_i^{-1}\mathbf{xc}$ from linearly independent vectors in $F = < \mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_d >$
    - 3. Recover the support From $E' = \bigcap_{i \in [d]} \mathbf{f}_i^{-1}\mathbf{xc}$
  - Required complexity: $\mathcal{O}(r^2 d^3 m)$
  - DFR (decryption failure rate): Lower than $q^{-(rd-n)}$

- **Remarks**
  - For lower values r, d, a lower decoding complexity can be expected

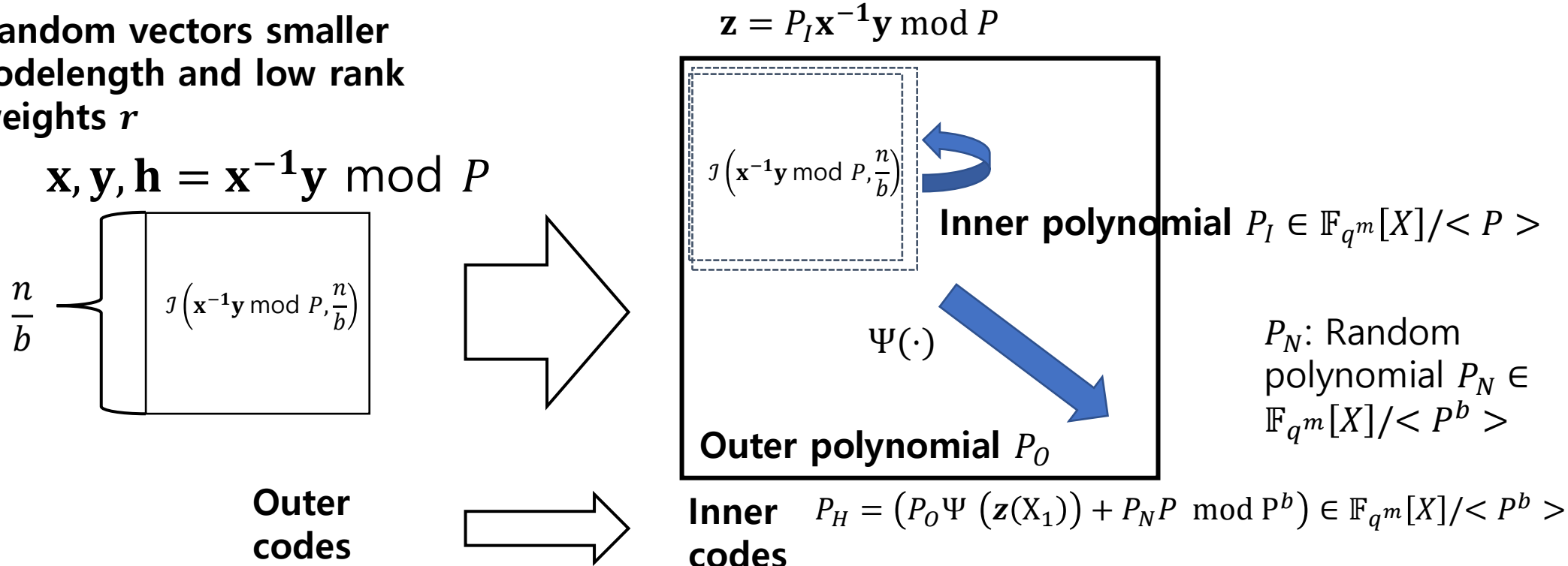
Rank Syndrome Decoding (RSD) algorithm

- **Rank syndrome decoding (RSD algorithm)**
  - By recovering E or F, generic or structural attacks was proposed, which determines the security level of the cryptosystem
    - **Generic attacks**: In order to recover F, derive all the combination with rank weight d (required complexity $\sim\Omega(q^d)$) first. The best strategy needs the complexity of $\mathcal{O}\left(n^3m^3q^{d\left\lceil\frac{m}{2}\right\rceil-m-n}\right)$
    - **Structural attacks:** In order to recover F, derive all the combination with rank weight r (required complexity $\sim\Omega(q^r)$), first. The best strategy needs the complexity of $\mathcal{O}\left(n^3m^3q^{r\left\lceil\frac{m(n+1)}{2n}\right\rceil-m}\right)$

- **Remarks**
  - For lower values r, d, we can obtain a lower security level, which lowers the security of the cryptosystem

New design criteria for the proposed KEM

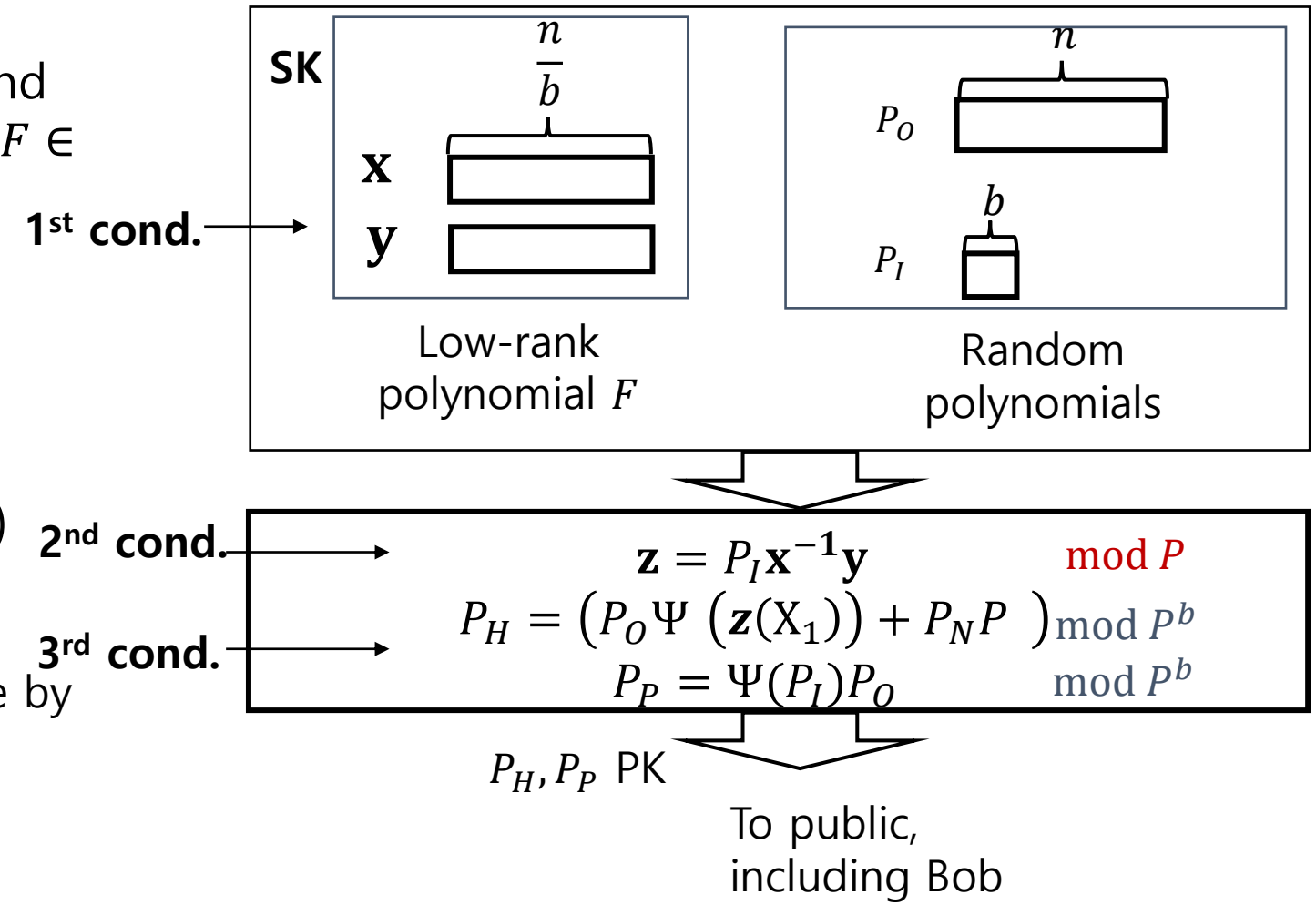
- Thus, the public key is designed from the ideal LRC codes with the **small codelength, small rank weights r and d**, and the following conditions
  - **1st condition**: All the operation is based on the polynomial ring
  - **2nd condition**: a low-rank codeword is not shown in the attacker
  - **3rd condition**: Properties of codeword are not specified in the ciphertext



Layered ROLLO-I: Procedures

1. Key generation

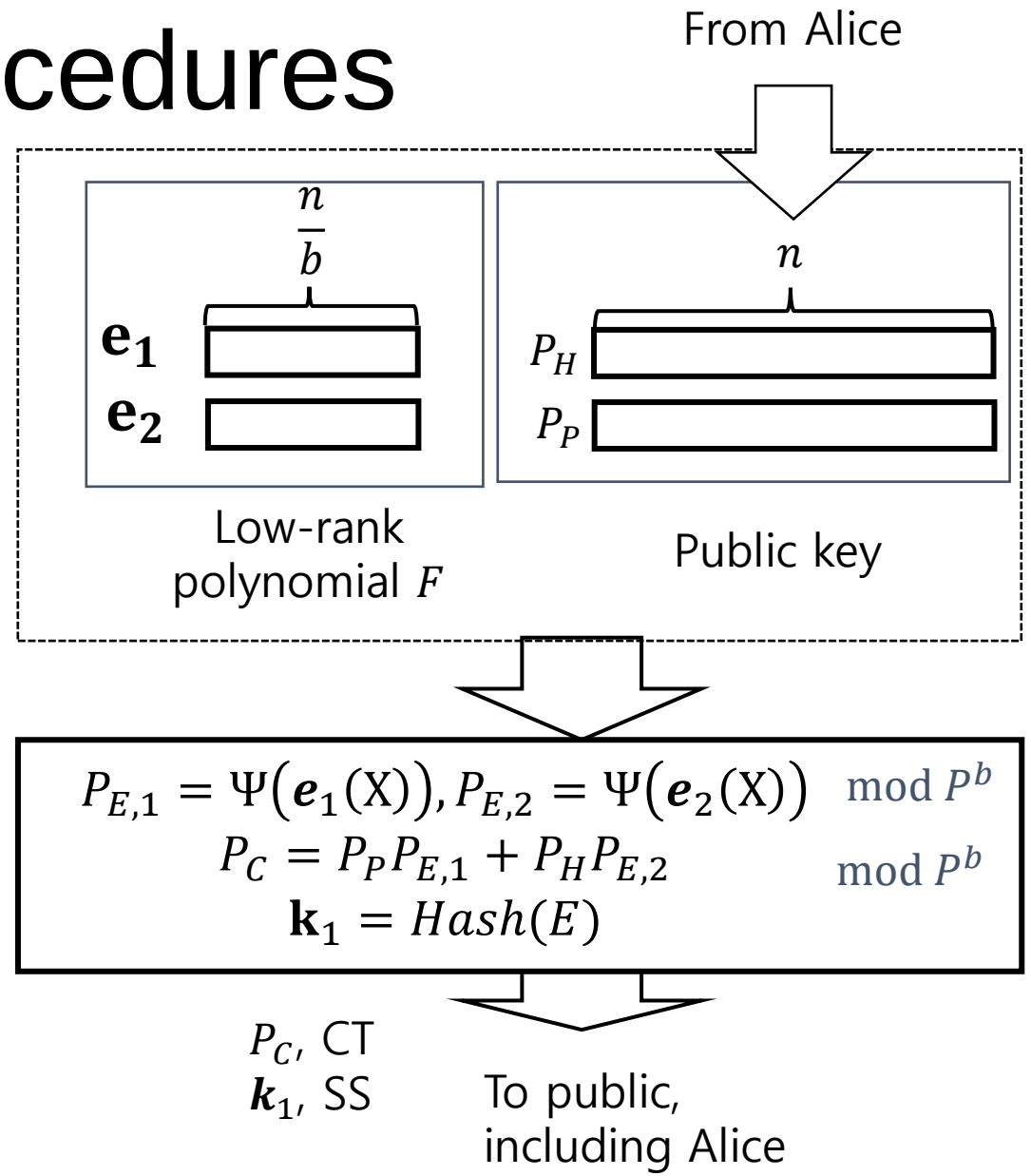
- Generate two random vectors $\mathbf{x}$ and $\mathbf{y}$ from the low-rank $\mathbb{F}_q$-subspace $F \in (\mathbb{F}_{q^m})^{\frac{n}{b}}$ with rank weight d
- Generate b-degree and n-degree random polynomials P_I and P_O.
- Generate $\mathbf{z}$ and P_H as
  - $\mathbf{z} = P_I\mathbf{x}^{-1}\mathbf{y} \bmod P$,
  - $P_H = \left(P_O\Psi\left(\boldsymbol{z}(\mathrm{X}_1)\right) + P_N P \bmod \mathrm{P}^b\right)$
- Finally, construct SK and PK as
  - **PK:** P_H, $P_P = \Psi(P_I)P_O \bmod P^b$
    (NOTE: We use an additional key size by P_P, which amounts to $\left\lceil \frac{n\log_2 m}{8} \right\rceil$[Byte])
  - **SK:** $\mathbf{x}, \mathbf{y}$, P_O, P_I



Layered ROLLO-I: Procedures

2. Encapsulation

- Generate Low-rank $\mathbb{F}_q$-subspace $E \in \left(\mathbb{F}_{q^m}\right)^{\frac{n}{b}}$ with rank weight r
- Generate $\frac{n}{b}$ –tuple two error vector $\boldsymbol{e}_1, \boldsymbol{e}_2 \in E$
- Obtain CT polynomial P_C as $P_C = \left(P_P P_{E,1} + P_H P_{E,2}\right) mod\ P^b$
- Obtain $\mathbf{k}_1 = Hash(E)$ to have a shared secret (SS)



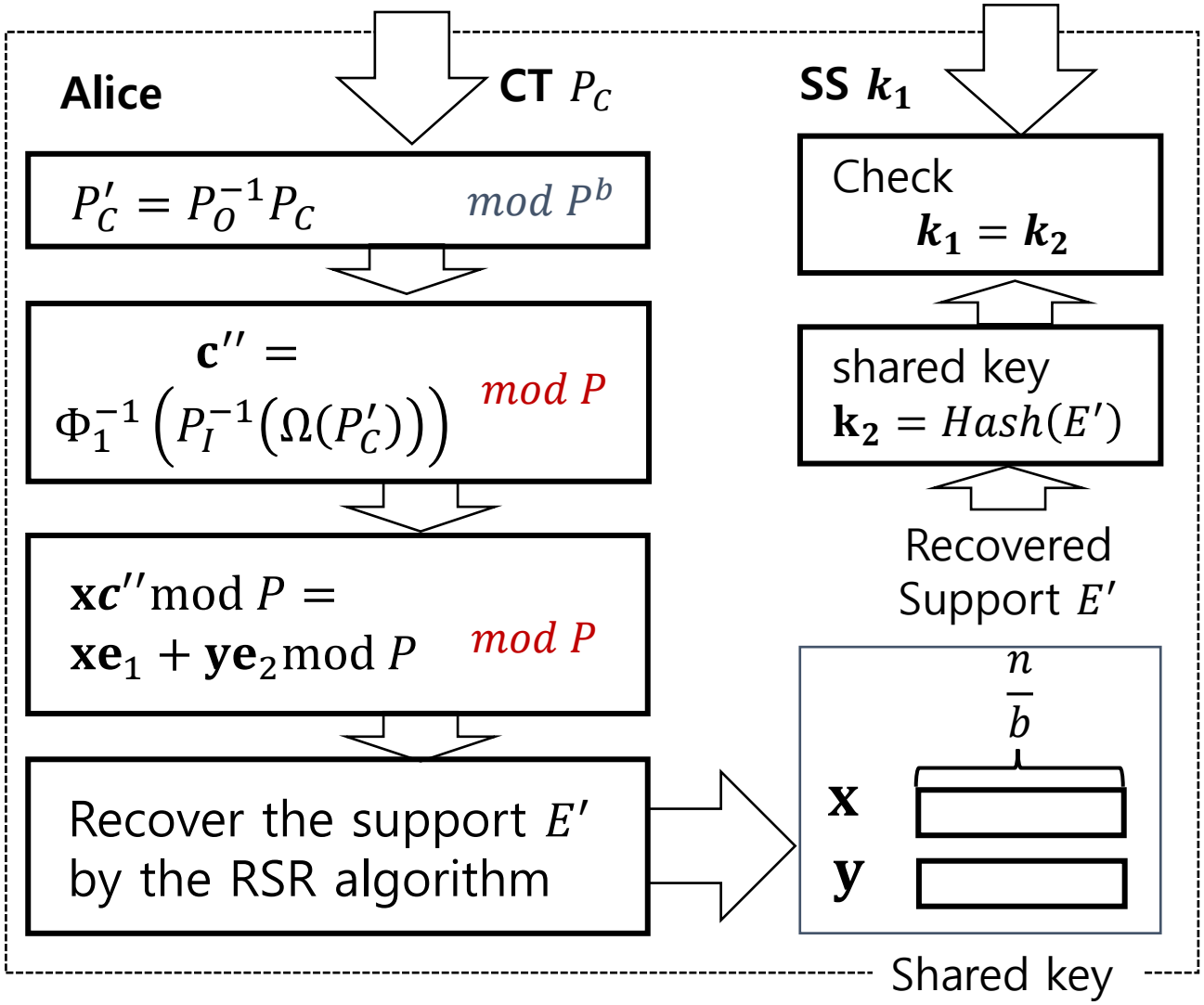
Layered ROLLO-I: Procedures

From Bob

3. Decapsulation

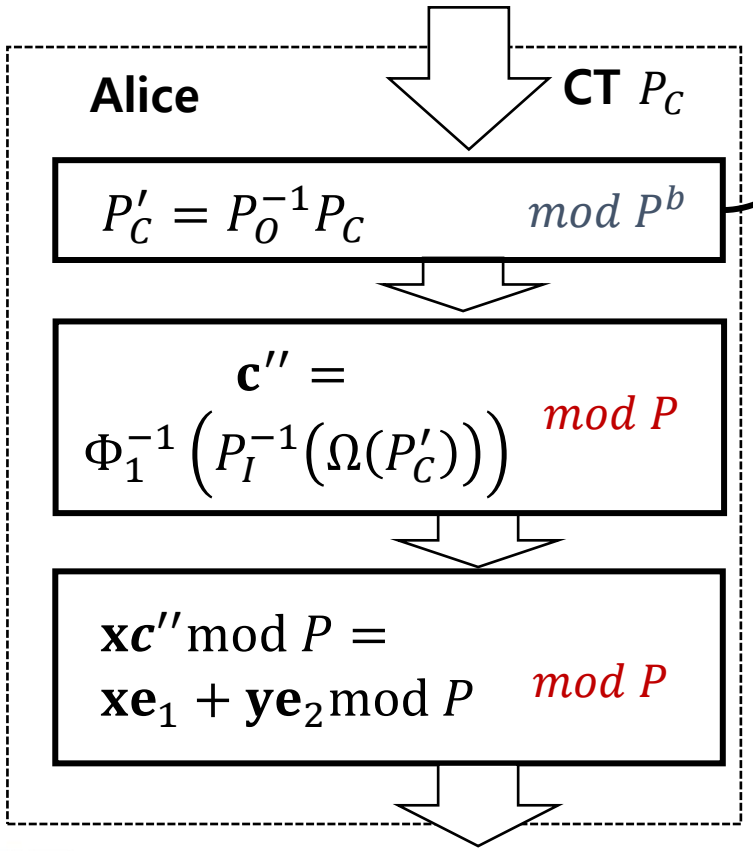
- Obtain the codeword $\mathbf{xc}'' = \mathbf{xe}_1 + \mathbf{ye}_2 \bmod P$ from P_C by
  - $P_C' = P_O^{-1} P_C \bmod P^b$
  - $\boldsymbol{c}'' = \Phi_1^{-1}\left(P_I^{-1}\left(\Omega\left(P_C'\right)\right)\right) \bmod P$
  - $\mathbf{xc}'' \bmod P = \mathbf{xe}_1 + \mathbf{ye}_2 \bmod P$

- From $\mathbf{xc}'' \bmod P$, recover the support E' by the RSR algorithm
- Derive shared key $\mathbf{k_2} = Hash(E')$ and if $\mathbf{k_1} = \mathbf{k_2}$, use the support E' as a shared key



Layered ROLLO-I: Procedures

- **In the decapsulation phase**



$$\Omega(\{P_O^{-1}P_C \mod P^b\}) \mod P$$
$$= \Omega(\{(\Psi(P_I)\Psi(\mathbf{e}_1(X_1))) \mod P^b\}) + \quad (1)$$
$$\Omega(\{\Psi(P_I\mathbf{x}^{-1}\mathbf{y}(X_1))\Psi(\mathbf{e}_2(X_1)) \mod P^b\}) + \quad (2)$$
$$\Omega(\{\Psi(P_N P)\Psi(\mathbf{e}_2(X_1)) \mod P^b\}) \mod P. \quad (3)$$

(1) For the first summand of (8), note that $\deg(P_I)+\deg(\mathbf{e}_1) \le n$. According to Proposition 1, we have

$$\Omega(\{(\Psi(P_I)\Psi(\mathbf{e}_1(X_1)) \mod P^b\}) \mod P = P_I\mathbf{e}_1 \mod P.$$

⇨ $P_I\mathbf{e}_1$ mod P

(2) In addition, note that $\deg(P_I)+\deg(\mathbf{x}^{-1}\mathbf{y}(X_1))+\mathbf{e}_2(X_1) < n$. Thus, we have

$$\Omega(\Psi(P_I\mathbf{x}^{-1}\mathbf{y}(X_1))\Psi(\mathbf{e}_2(X_1)) \mod P^b) \mod P = P_I\mathbf{x}^{-1}\mathbf{y}\mathbf{e}_2 \mod P.$$

⇨ $P_I\mathbf{x}\mathbf{y}^{-1}\boldsymbol{e}_2$ mod P

(3) Also, the third summand of (8) becomes 0 by the map Ω.

⇨ **0** mod P

In summary, we have

$$(P_I^{-1}\Omega(P'_C)) \mod P = P_I^{-1}P_I\mathbf{e}_1(X_1) + P_I^{-1}P_I\mathbf{x}^{-1}\mathbf{y}\mathbf{e}_2(X_1) \mod P = \mathbf{e}_1(X_1) + \mathbf{x}^{-1}\mathbf{y}\mathbf{e}_2(X_1) \mod P.$$

Computational Complexity Analysis

- Compared to ROLLO-I, there are several advantages and drawbacks of the proposed approach regarding its computational complexity
  - **Advantage**: Key generation and decapsulation
    - **Parameters r or d(rank weight)**: For RSR algorithm in decapsulation, the complexity from lower r or d is decreased by the quadratic or cubic level (with $\mathcal{O}(r^2d^3m)$)
    - **Parameter b**: It did not change the complexity RSR algorithm. However, small codelength is better for the key generation and lower DFR
  - **Drawback:** Key generation, encapsulation, and decapsulation
    - For the overall procedure, additional operation regarding $\mathrm{mod}\ P^b$ is necessary, which increase the complexity
    - Especially, the complexity for computing inverse of P_I and P_O is high
      - Larger b increase the complexity for invese of P_I, P_O and $\mathrm{mod}\ P^b$ operation and thus, lower b is better

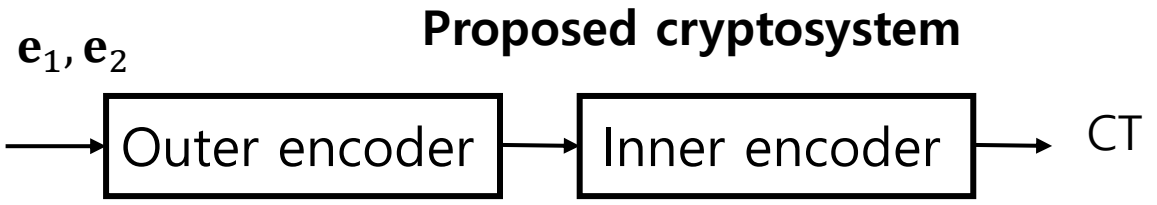
Security Analysis

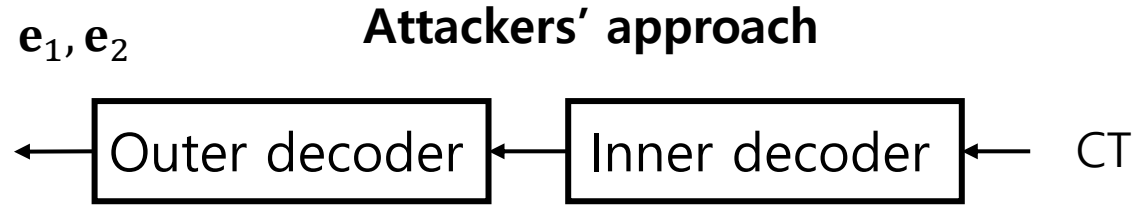
- **Indistinguishability:** In the existing PK $\mathbf{h} = \mathbf{x}^{-1}\mathbf{y}$, new public key P_H multiplying two random polynomial P_I and P_O do not change the property

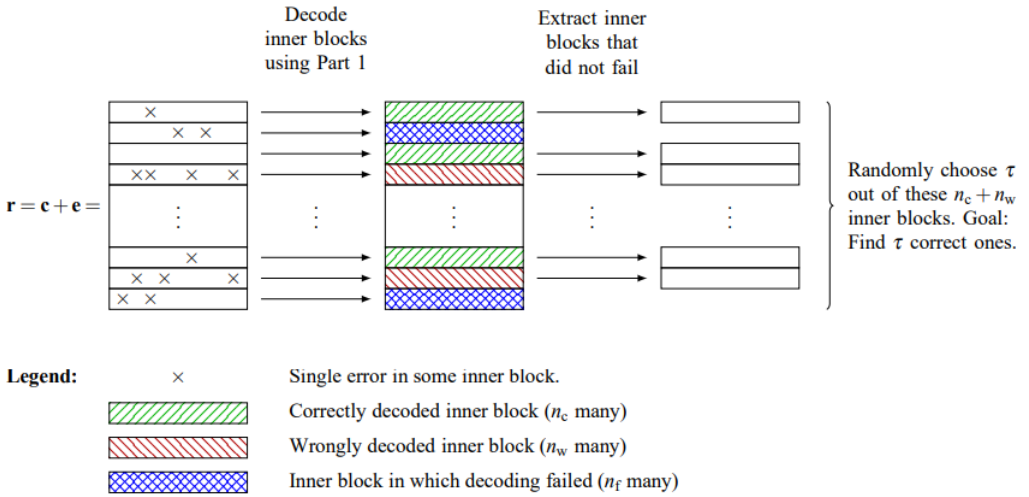
- **Possible Attack Scenarios:** Firstly, we should check some attack scenarios by the additional PK information P_P
  - **Direct attack:** Using the equation $P_P^{-1}P_C \bmod P^b$
    - $\left(P_I^{-1}P_O^{-1}P_c mod\ P^b\right) mod\ P$ returns the different polynomials from $P_I^{-1}\left(P_O^{-1}P_C\ mod\ P^b\right)\ mod\ P$
    - By $\mathrm{Deg}(P_I^{-1}) \sim \frac{n}{b}$ and $\mathrm{Deg}(P_O^{-1}) \sim n$, reduction for Proposition 1 is not applied
  - **Polynomial attack:** Using the information on P_P
    - An attack cannot guess P_O and P_I from the degree or coffeicients on P_P(indistinguishability)

Security Analysis

- **Existing Attacks on Generalized concatenated (GC) codes [3]**
  - **Motivation**: The proposed KEM can be considered as an encoding process of generalized concatenated (GC) codes
  - Then, the attack can be analyzed using the existing Sendrier's attack
    - Sendrier's attack consists of two phases
    - **1st phase:** From the multiple observation of codewords, firstly find a structure of the inner codes
    - **2nd phase:** Based on this, recover the outer codes







[3] S Puchinger, S. Muelich, K. Ishak, and M. Bossert, ``Code-based cryptosystems using generalized concatenated codes,'' ₩textit{App. of Comp. Alg.}, Kalamata:Greece, Jul. 20-23, 2015.

Security Analysis

- **For an attack scenario in the proposed KEM**
  - 1st Phase
    - Attacker **do not collect a multiple codeword** because generator polynomials P_I and P_O of the inner code are changed in each CT
    - Instead, attack can guess P_O or P_I and if the exact polynomial is found, they can proceed to the 2nd phase
    - If the attack guess P_I(easier than guessing P_O), the required complexity is $\mathcal{O}(\, q^{(b-1)m})$ by guessing each coefficients of the codes
    - For each guessing, the attacker should proceed to the 2nd phase because attacker cannot convince the exact value only from guessed $P_H^{'}$
  - 2nd Phase
    - In each case, attacker use the existing attacks for the ideal LRPC codes.
  - Total complexity
    - Generic attack: Complexity amounts to $\mathcal{O}\left(q^{(b-1)m} \times \binom{n}{\frac{n}{b}}^2 m^3 q^{\sum_{i\in[1,b]}(d_i)^3\left\lceil\frac{m}{2}\right\rceil - m - \frac{n}{b}}\right)$
    - Structural attack: Complexity amounts to $\mathcal{O}\left(q^{(b-1)m} \times \left(\frac{nm}{b}\right)^3 q^{r\left\lceil\frac{m\left(\frac{n}{b}+1\right)}{\frac{2n}{b}}\right\rceil - m}\right)$

Suggested Parameters

- **Suggested parameter for the proposed KEM**
  - D_c decareaes, size of PK increase, Other parameters are maintained

기존 NIST 제출 ROLLO-I 파라미터

인스턴스	q	n	m	r	d	b	S_G	S_S	D_C	DFR	공개키 크기
ROLLO-I-128	2	83	67	7	8	1	207.7	155.3	20.7	2^{-27}	696
ROLLO-I-192	2	97	79	8	8	1	279.0	178.7	21.3	2^{-33}	958
ROLLO-1-256	2	113	97	9	9	1	383.6	266.8	22.4	2^{-32}	1371

제안 방법론에 기반한 KEM 파라미터

인스턴스	q	n	m	r	d	b	S_G	S_S	D_C	DFR	공개키 크기
Proposed-128	2	74	67	3	2	2	138.1	130.1	13.24	2^{-31}	1240
Proposed-192	2	86	79	4	3	2	199.2	225.2	16.05	2^{-35}	1857
Proposed-256	2	106	97	5	3	2	287.0	275.0	17.00	2^{-38}	2571

Implementation environments

- **RBC(Rank-based cryptography) library**
  - ➢ RBC is an open-source library for rank-metric code-based PQC in 2021 which includes operations MRD and LRPC codes
  - ➢ Consists Python Wrapper, mainly C language with AVX-2 instruction for Intel CPU

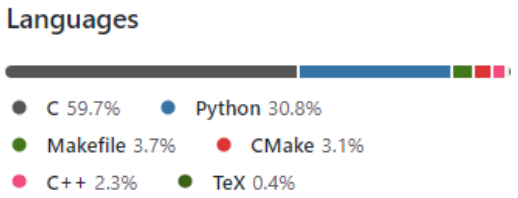
- **Simulation environment**
  - ➢ The 12th Gen. Intel® Core™ i9-12900K multi-core CPU with AVX-2 support
  - ➢ 32GB DDR5 4.800MHz memory
  - ➢ Ubuntu Linux 20.04 LTS
  - ➢ Measured as an worst processing cycle in the 100 iteration

ROLLO-I-128

```
carisis@carisis-ECTest:~/Downloads/BII_LRPC_Init$ ./bin/rollo_i_128

Keygen: 3810534 CPU cycles
Encaps: 389839 CPU cycles
Decaps: 5887980 CPU cycles

secret1: 2885e255fbc20c57baf8fbd2386cd7b6694798fb6cfa6ec28f5ce4988a2e2451b1c1fef
649e8d2a6ca0b444d8fe828dc8f9b6ca9053ae2d2ba1e2351bcbbe094
secret2: 2885e255fbc20c57baf8fbd2386cd7b6694798fb6cfa6ec28f5ce4988a2e2451b1c1fef
649e8d2a6ca0b444d8fe828dc8f9b6ca9053ae2d2ba1e2351bcbbe094
```

Prop-128

```
carisis@carisis-ECTest:~/Downloads/BII_LRPC_Init$ ./bin/biix_128
4 4

Keygen: 3459638 CPU cycles
Encaps: 1017531 CPU cycles
Decaps: 3208056 CPU cycles

secret1: 25e2734b9d3fd9f666dc4b1115f66e630133ff4b3a603356cff361ac4855ec23b9f0f1a
1ab763f655e5497b0a142ce5f2873b0c6b990b9be6cac34d65b2034a815f03af22b32998563b4727
c9d564cf6b2c144c9c6809b3f70b8ac0e98ae0b8d4b47b1039087e64d90915069754c66bbacbf1fd
bde5425d2651af77d18027257
secret2: 25e2734b9d3fd9f666dc4b1115f66e63
1ab763f655e5497b0a142ce5f2873b0c6b990b9be
c9d564cf6b2c144c9c6809b3f70b8ac0e98ae0b8d
bde5425d2651af77d18027257
```



Code implementation method

- **Essential library:** python3, python-yaml, cmake, make, gcc, openssl

```
carisis@carisis-ECTest:~/Downloads/BII_LRPC_Init_2$ sudo apt install python3 pyt
hon-yaml cmake make gcc openssl
```

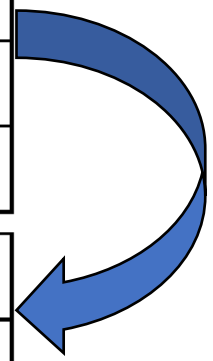
- **Implementation method:** Mainly use the functions in RBC library, and adds some functions and parameters for the additional functionality
- New code builds the performance results and KAT for proposed KEM and existing ROLLO-I together, thus their performances can be directly compared.
  - Build command: python rbc-lib.py
  - Execution command: ./bin/biix_XXX
  - KAT(Known answer test) generation command: ./bin/kat_biix_XXX

```
carisis@carisis-ECTest:~/Downloads/BII_LRPC_Init_2$ python rbc-lib.py

### Parsing configuration file

### Removing previous build

### Preprocessing library
    Templating core files
    Preprocessing core67
    Preprocessing core79
    Preprocessing core97
```

Code performance analysis

- **Performance measure:** The number of CPU processing cycle for key generation, encapsulation, and decapsulation
  - ➢ The proposed KEM have **processing cycle** reduction by **40-70%** for the same security level compared to the existing ROLLO-I

Instances	Keygen.	Encap.	Decap.	Total
ROLLO-I-128	6,019,622	574,711	8,287,089	14,881,422
ROLLO-I-192	4,388,835	577,348	7,955,763	12,922,035
ROLLO-I-256	8,361,499	672,956	10,878,644	19,903,099
Proposed-128	2,609,907	661,423	5,570,494	8,841,824
Proposed-192	2,921,813	755,759	5,253,698	8,931,270
Proposed-256	3,757,592	918,300	10,424,395	15,100,287



Conclusion

- In this study, research is conducted to improve rank-based codes and ROLLO-I, which have the advantage of small key size among quantum resistant cryptography, through a new block-based hierarchical structure

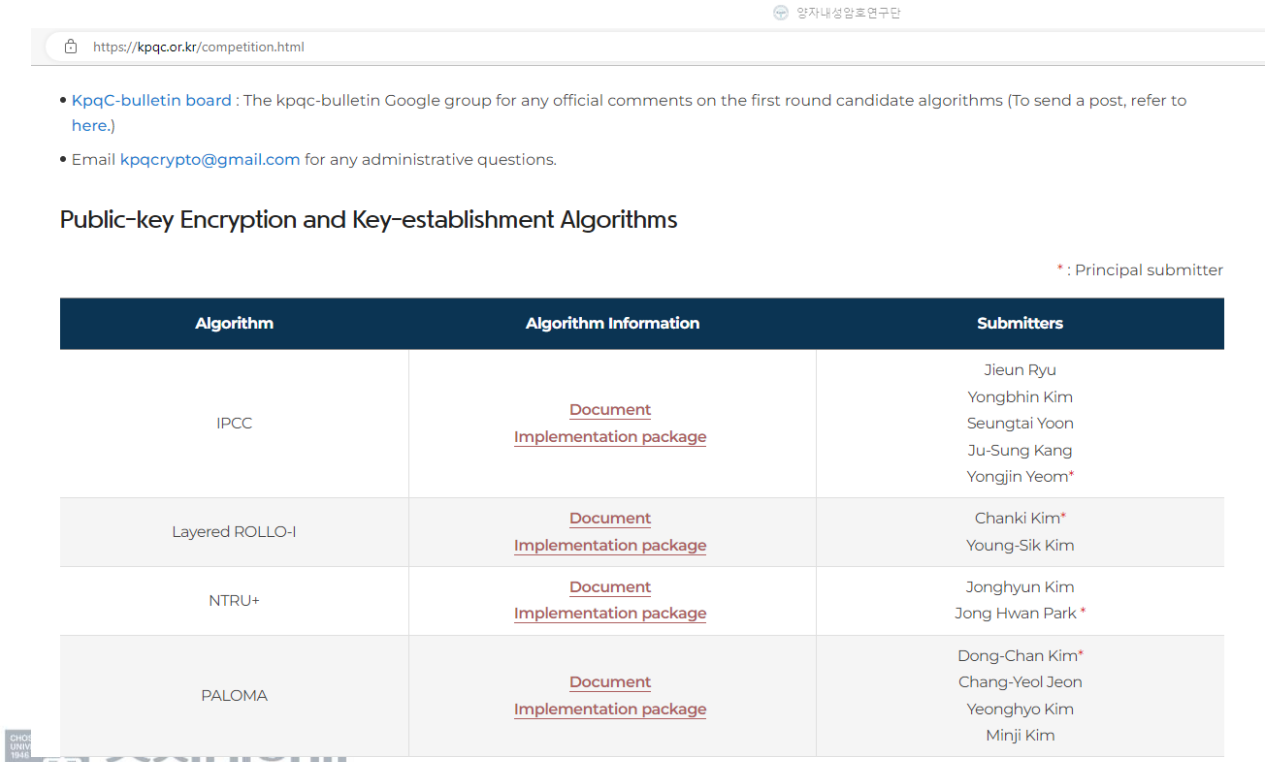
  - The proposed method reduces the rank weight of the ideal LRPC code used in ROLLO-I to a small size, and introduces a hierarchical structure for compensating the decrease in security level

  - Compared to the existing ROLLO-I the proposed parameters utilize an increased public key size, but have the advantage of faster cryptographic operation

  - This can be used to solve the problem for high decoding complexity, which is one of the major difficulties of rank-based codes and the proposed approach makes rank-based encryption systems more competitive.

Further Information

- KPQC Hompage: https://kpqc.or.kr/competition.html(Documents and source code for 1 round submission)
- Cryptography Arxiv: Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes (iacr.org)
- Layered-ROLLO-I Homepage: To be announced shortly
- Or contact me (carisis@jbnu.ac.kr)


양자내성암호연구단

https://kpqc.or.kr/competition.html

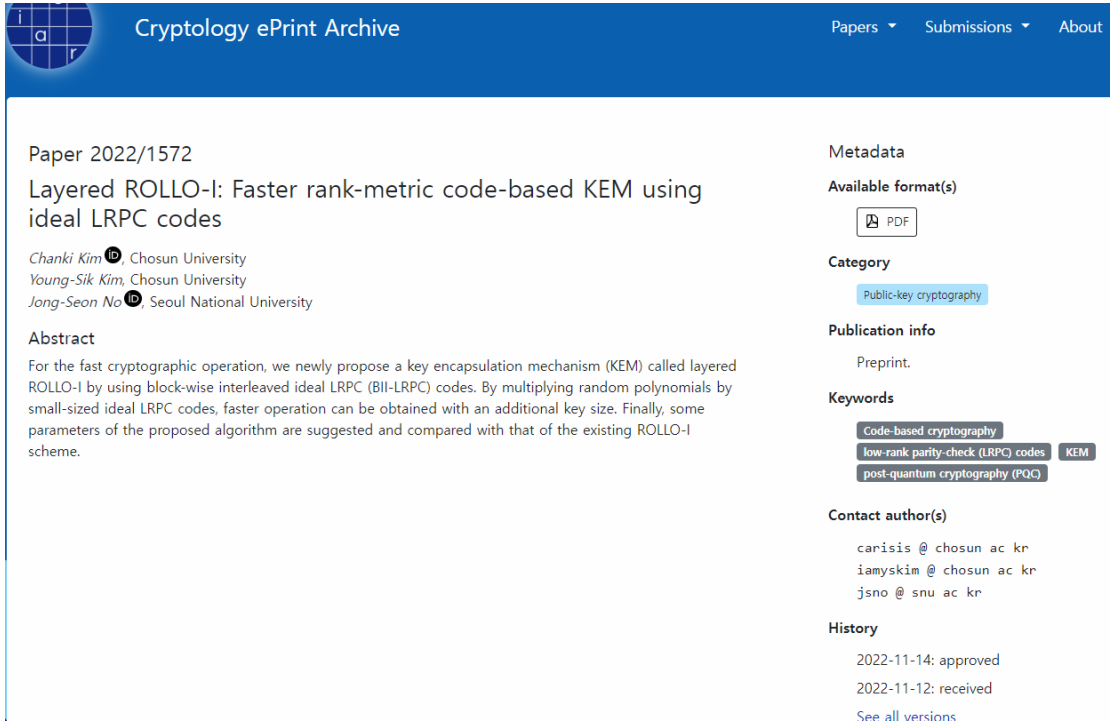
• KpqC-bulletin board : The kpqc-bulletin Google group for any official comments on the first round candidate algorithms (To send a post, refer to here.)

• Email kpqcrypto@gmail.com for any administrative questions.

Public-key Encryption and Key-establishment Algorithms

* : Principal submitter

Algorithm	Algorithm Information	Submitters
IPCC	Document Implementation package	Jieun Ryu Yongbhin Kim Seungtai Yoon Ju-Sung Kang Yongjin Yeom*
Layered ROLLO-I	Document Implementation package	Chanki Kim* Young-Sik Kim
NTRU+	Document Implementation package	Jonghyun Kim Jong Hwan Park *
PALOMA	Document Implementation package	Dong-Chan Kim* Chang-Yeol Jeon Yeonghyo Kim Minji Kim


Cryptology ePrint Archive

Papers ▾ Submissions ▾ About

Paper 2022/1572

Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes

Chanki Kim, Chosun University
Young-Sik Kim, Chosun University
Jong-Seon No, Seoul National University

Abstract

For the fast cryptographic operation, we newly propose a key encapsulation mechanism (KEM) called layered ROLLO-I by using block-wise interleaved ideal LRPC (BII-LRPC) codes. By multiplying random polynomials by small-sized ideal LRPC codes, faster operation can be obtained with an additional key size. Finally, some parameters of the proposed algorithm are suggested and compared with that of the existing ROLLO-I scheme.

Metadata

Available format(s)

PDF

Category

Public-key cryptography

Publication info

Preprint.

Keywords

Code-based cryptography
low-rank parity-check (LRPC) codes
KEM
post-quantum cryptography (PQC)

Contact author(s)

carisis @ chosun ac kr
iamyskim @ chosun ac kr
jsno @ snu ac kr

History

2022-11-14: approved
2022-11-12: received

See all versions

Thank you