JEONBUK NATIONAL UNIVERSITY

kPQC 7th workshop
2023/11/14

Layered ROLLO 공격 시나리오 및 개선점 분석 (Attack scenarios and improvements on Layered ROLLO)

Chanki Kim

Jeonbuk Nat'l. Univ.

Dept. of Computer Science and Artificial Intelligence

Layered ROLLO

ROLLO

- 저랭크 패리티 체크 부호(LRPC 부호)를 활용한 고성능 KEM 기법
  - 장점: 해밍 거리가 아닌 랭크 거리에 기반한 방법론, 부호 기반 양자 내성암호 중 상대적으로 작은 키 사이즈 및 고성능의 장점을 획득
  - 단점: 여전히 격자 기반 암호보다는 동작 성능적 열위에 있음, 기반 문제에 대한 안정성이 이론적으로 완전히 보장되지는 않음

Layered-ROLLO

- LRPC 부호의 랭크 거리를 낮추어 복호 성능을 높이고, 대신 무작위 다항식 및 두 모듈러 연산의 구조를 이용하여 보안성을 확보
  - 장점: 파라미터 다변화 및 상당한 동작 성능 향상을 얻을 수 있음
  - 단점: 추가적인 연산이 필요하며, 키 사이즈가 증가함, 추가 연산 부분에 대한 보안성 검증이 필요

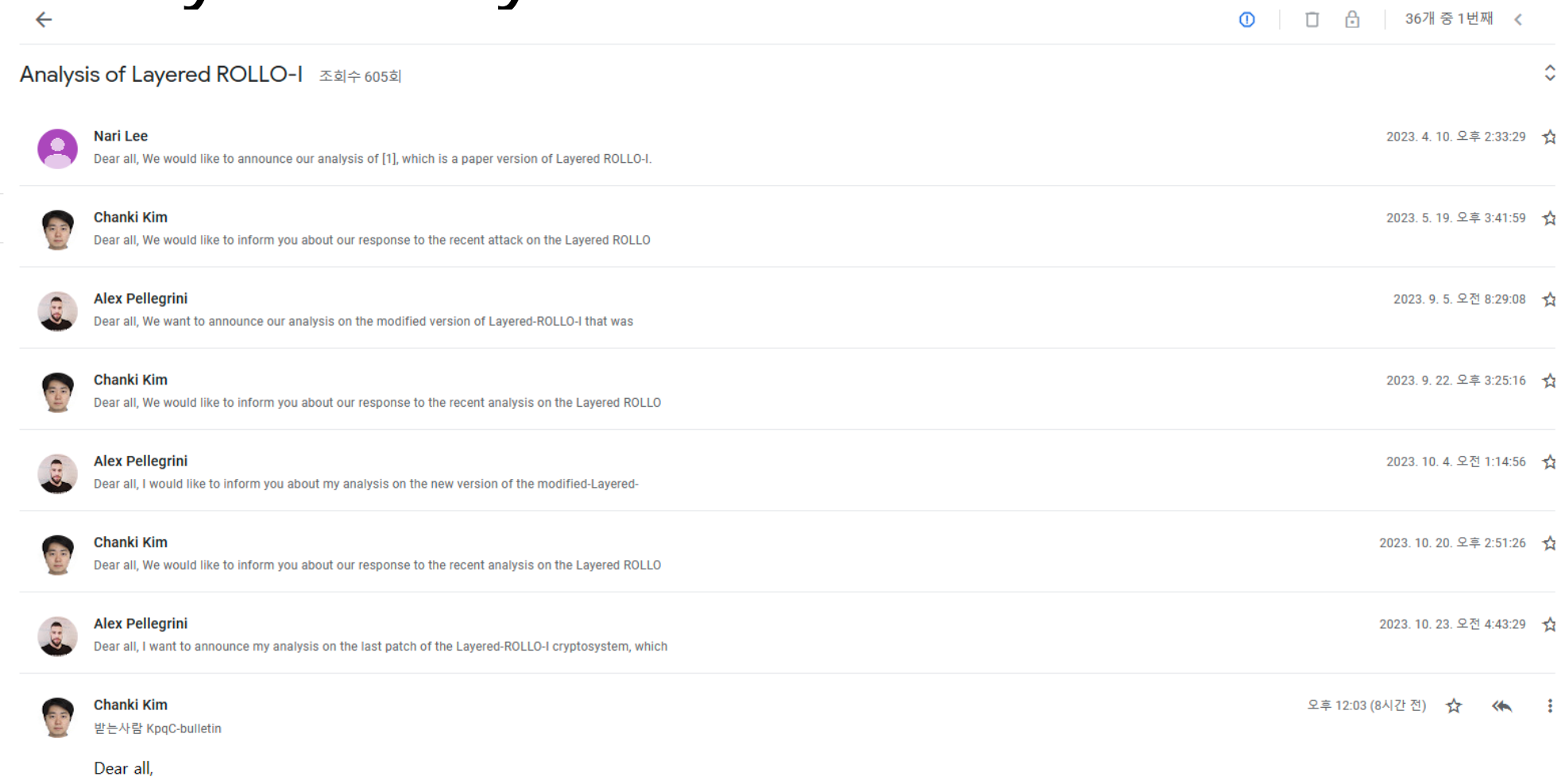
Outlook on Layered ROLLO Scheme

- 1st submission (22/10/31)

- A series of attacks and modifications are uploaded in in kPQC-bulletin
  - First attacks and the corresponding modification are uploaded (23/04/10, 23/05/19)
  - Second attacks and the corresponding modification are uploaded (23/09/05, 23/09/22)
  - Third attacks and the corresponding modification are uploaded (23/10/04, 23/10/20)
  - Fourth attacks and the corresponding modification are uploaded (23/10/22, 23/11/06)
  - Presentation on 7th KPQC workshop(23/11/14)

- 2nd submission(planned)(23/02/23)

History on Layered ROLLO Scheme



36개 중 1번째

Analysis of Layered ROLLO-I 조회수 605회

Nari Lee 2023. 4. 10. 오후 2:33:29
Dear all, We would like to announce our analysis of [1], which is a paper version of Layered ROLLO-I.

Chanki Kim 2023. 5. 19. 오후 3:41:59
Dear all, We would like to inform you about our response to the recent attack on the Layered ROLLO

Alex Pellegrini 2023. 9. 5. 오전 8:29:08
Dear all, We want to announce our analysis on the modified version of Layered-ROLLO-I that was

Chanki Kim 2023. 9. 22. 오후 3:25:16
Dear all, We would like to inform you about our response to the recent analysis on the Layered ROLLO

Alex Pellegrini 2023. 10. 4. 오전 1:14:56
Dear all, I would like to inform you about my analysis on the new version of the modified-Layered-

Chanki Kim 2023. 10. 20. 오후 2:51:26
Dear all, We would like to inform you about our response to the recent analysis on the Layered ROLLO

Alex Pellegrini 2023. 10. 23. 오전 4:43:29
Dear all, I want to announce my analysis on the last patch of the Layered-ROLLO-I cryptosystem, which

Chanki Kim 오후 12:03 (8시간 전)
받는사람 KpqC-bulletin

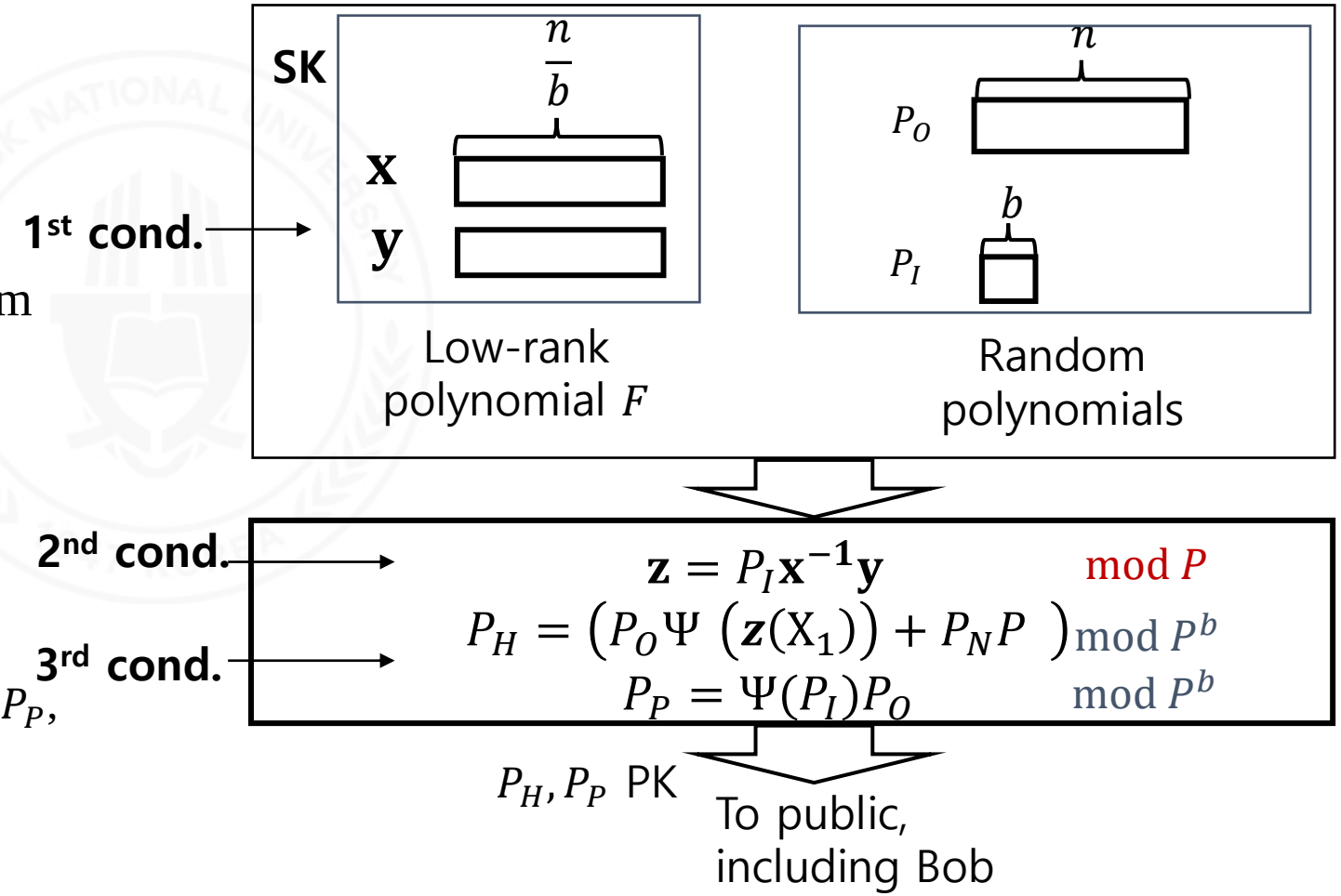
Dear all,

Layered ROLLO(1st Submission, 2022, 10)

1. Key generation

- Generate two random vectors $\mathbf{x}$ and $\mathbf{y}$ from the low-rank $\mathbb{F}_q$-subspace $F \in \left(\mathbb{F}_{q^m}\right)^{\frac{n}{b}}$ with rank weight d
- Generate b-degree and n-degree random polynomials P_I and P_O.
- Generate $\mathbf{z}$ and P_H as
  - $\mathbf{z} = P_I\mathbf{x}^{-1}\mathbf{y} \mod P$,
  - $P_H = \left(P_O\Psi\left(\boldsymbol{z}(\mathrm{X}_1)\right) + P_N P \mod \mathrm{P}^b\right)$
- Finally, construct SK and PK as
  - **PK:** $P_H, P_P = \Psi(P_I)P_O \mod P^b$

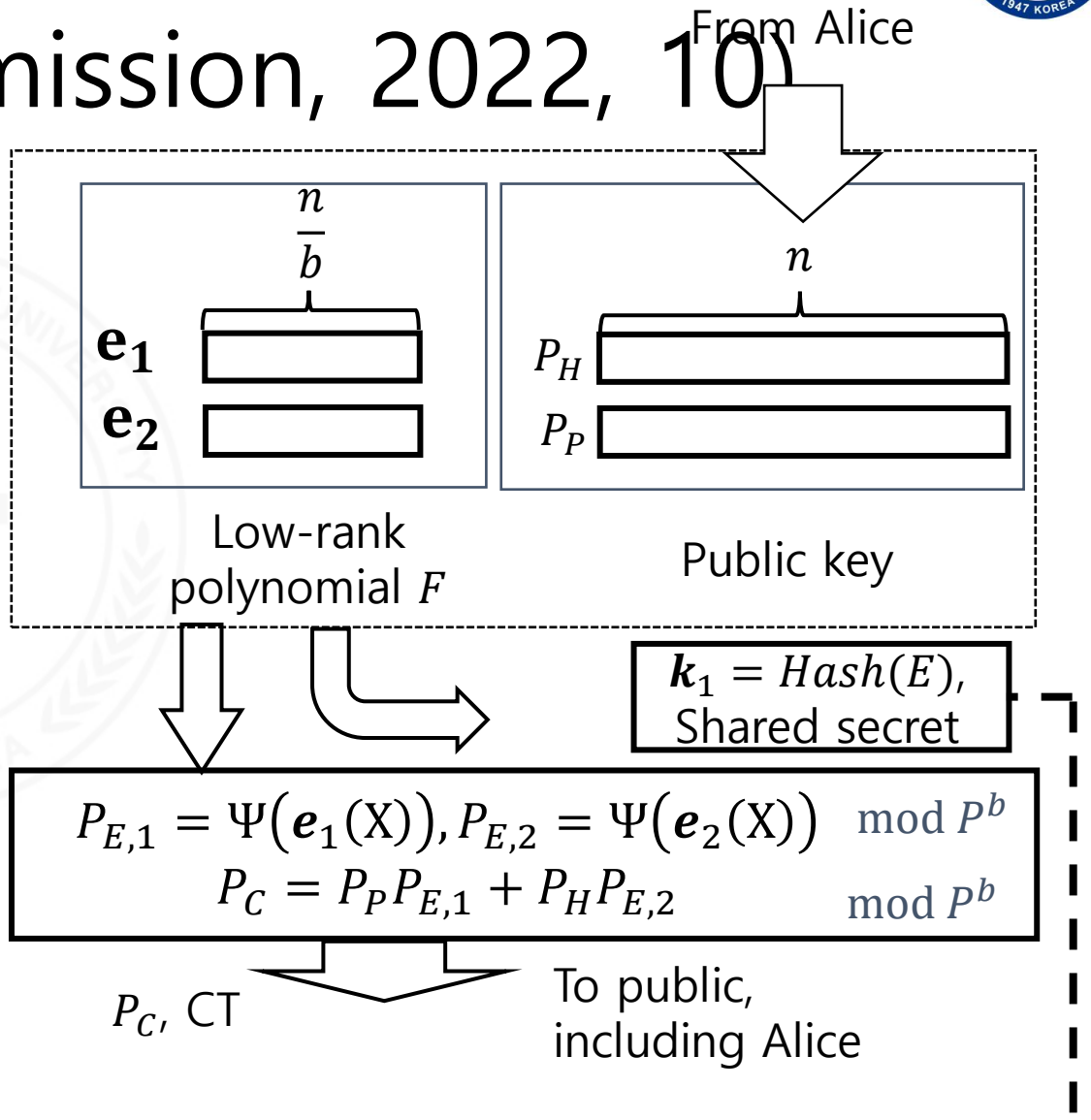
  (NOTE: We use an additional key size by P_P, which amounts to $\left\lceil \frac{n\log_2 m}{8} \right\rceil$[Byte])
  - **SK:** $\mathbf{x}, \mathbf{y}, P_O, P_I$



Layered ROLLO(1st Submission, 2022, 10)

2. Encapsulation

- Generate Low-rank $\mathbb{F}_q$-subspace $E \in \left(\mathbb{F}_{q^m}\right)^{\frac{n}{b}}$ with rank weight r with the first b zero elements
- 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)

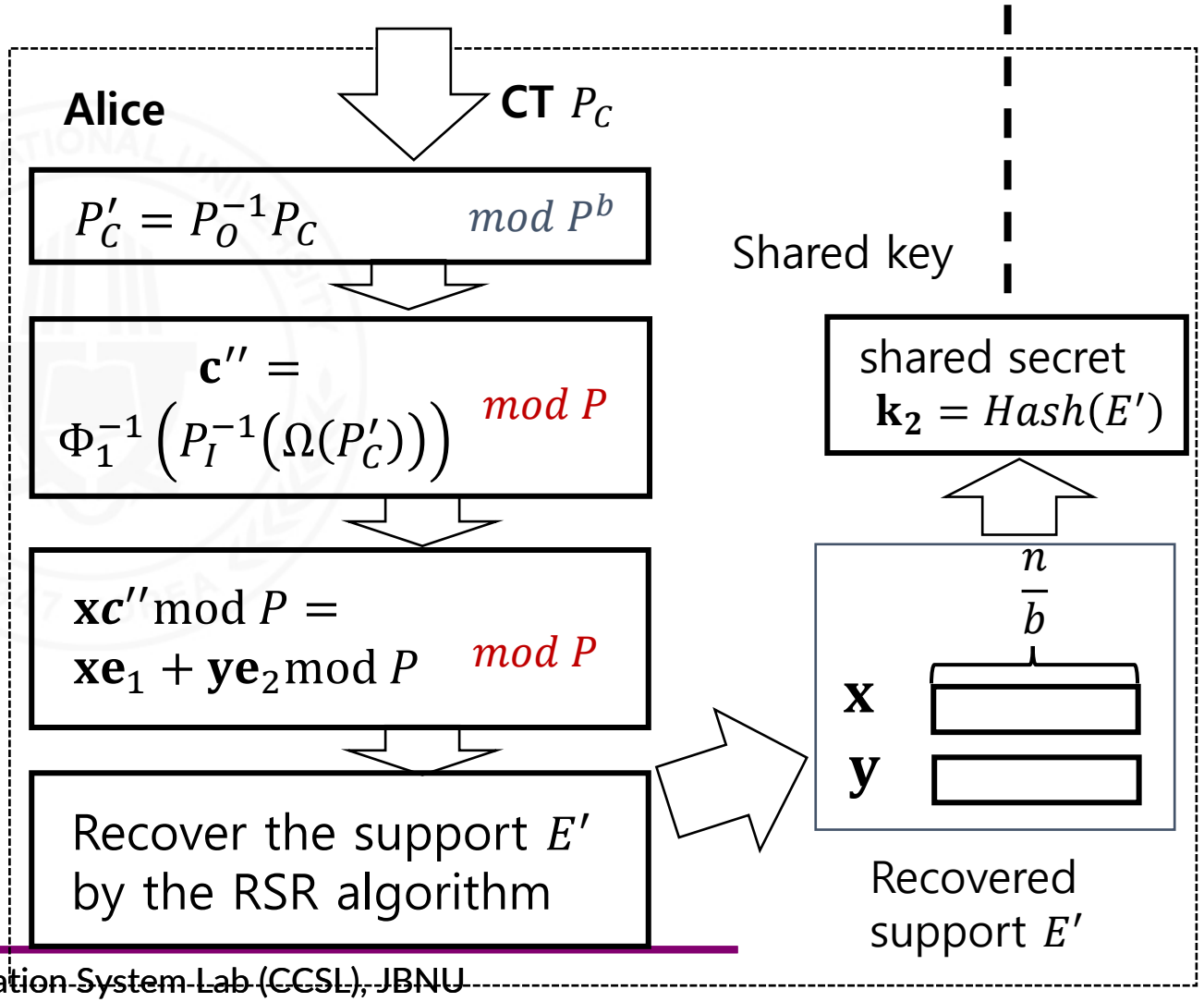


Layered ROLLO(1st Submission, 2022, 10)

3. Decapsulation

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

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



Layered ROLLO(1st Submission, 2022, 10)

- **Suggested parameter for the existing ROLLO-I**

Name	q	n	m	r	d	b	Sec. (Conv Gen)	Sec. (Conv Stru.)	DFR	Dec. Comp.	pk	ct size
ROLLO-I-128	2	83	67	**7**	8	1	207.648687	155.3233859	**-27**	**20.68079903**	696	696
ROLLO-I-192	2	97	79	**8**	8	1	278.968813	178.7110808	**-33**	**21.30378075**	958	958
ROLLO-I-256	2	113	97	**9**	9	1	383.623107	266.7602754	**-32**	**22.44953785**	1371	1371

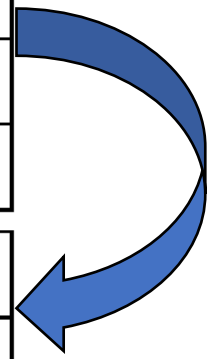
- **Suggested parameter for the existing Layered ROLLO-I**

Name	q	n	m	r	d'	P	b	Sec. (Conv,Gen)	Sec. (Conv,Stru)	Sec. (New,Comb)	Sec. (New, Alge)	DFR	Dec. Comp.	pk	ct size
Existing-128	2	74	67	**3**	2	47	2	~~138.826628~~	~~63.82662767~~	~~59.551085~~	~~66.15394273~~	**-31**	**13.236**	1240	620
Existing-192	2	86	79	**4**	3	39	2	~~199.190137~~	~~110.6901365~~	~~106.46009~~	~~89.85304835~~	**-31**	**16.0587**	1699	850
Existing-256	2	106	97	**5**	3	31	2	~~286.9835~~	~~129.4834999~~	~~174.65567~~	~~116.2798717~~	**-38**	**16.9987**	2571	1286

Layered ROLLO(1st Submission, 2022, 10)

- **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



Summary of the first attack (2023.4)

- Two different types of attacks are actually occurred by insufficiently protected polynomial P_E(low-rank message) and P_P(PK) in the modulus P^b and thus, we do following:
- As an alternatives for new attacks on Layered ROLLO-I, the modified scheme is proposed with little performance degradation, but no additional key size.

Comment No.	Comments	Modification
1.1-1.3	Attacks on E over ciphertext (SL reduction)	Avoiding attack **by increasing value of r** (Blocking the attack or SL enhancement)
1.4	Errors on security analysis (SL reduction)	Increasing $\boldsymbol{P_I}$ **degrees** (SL enhancement)
2	Direct attack is possible (broken)	Avoiding attack by using generalized modulus $P^{(2)}$ (Blocking the attack method)

Summary of the first attack(2023.4)

1.1 Key recovery instance

1.1 Instance

A ciphertext in Layered ROLLO-I is generated as follows.

$$\mathbf{c} = \Phi_2^{-1}(P_P P_{E,1} + P_H P_{E,2} \mod P^b),$$

where the vector $(\Phi_2^{-1}(P_{E,1}), \Phi_2^{-1}(P_{E,2})) \in \mathbb{F}_{q^m}^{2n}$ is of rank weight r.

Notice that the ciphertext $\mathbf{c}$ is a syndrome calculated by the parity check matrix

$$H_1 \quad [\mathcal{I}(\Phi_2^{-1}(P_P, P^b)) \,|\, \mathcal{I}(\Phi_2^{-1}(P_H, P^b))] \quad H_2$$

- Here, E is generated from **low-rank support**. Since the ciphertext can be considered as a syndrome.
- Low-rank property of E can be exploited to attack the cryptosystem

Summary of the first attack(2023.4)

1.2 Combinatorial attack

- The attack on the low-rank support E can be performed from the existing generic attack on modulus P^b
- Increasing rank distance on E can protect the security

1.2 Combinatorial attack

The complexity of the combinatorial attack for an $[tn, n]$ ideal code over $\mathbb{F}_{q^m}$ is suggested in [3] as follows.

$$\mathcal{O}\left(((t-1)nm)^{\omega} q^{r\left\lceil \frac{m(n+1)}{tn} \right\rceil - m}\right).$$

The value for ω is suggested to be taken as $\omega = 2$ in [3], but we will also consider $\omega = 3$ to show that the parameters of Layered ROLLO-I fall considerably short of the security level in any cases. The costs of the combinatorial attacks are shown on the table below.

Security Level	q	n	m	r	t	Complexity: $\omega = 2$	Complexity: $\omega = 3$
128	2	74	67	3	2	2^{60}	2^{72}
192	2	86	79	4	2	2^{106}	2^{119}
256	2	106	97	5	2	2^{175}	2^{188}

Summary of the first attack(2023.4)

1.3 Algebraic attack

- The attack can be performed from the existing generic attack on the modulus P^b
- Over-determined or super-determined cases are not applied for high rank weight r

- Super-overdetermined case:

$$\mathcal{O}\left(m\binom{n-p-1}{r}\binom{2n-p}{r}^{\omega-1}\right)$$

The calculated results are shown in the table below. We used $\omega = 2.81$ as suggested in [1].

Security Level	q	n	m	r	Over. Case	p	Super-over. Case
128	2	74	67	3	2^{56}	47	2^{49}
192	2	86	79	4	2^{73}	39	2^{66}
256	2	106	97	5	2^{90}	31	2^{86}

1.3 Algebraic attack

Algebraic attacks consider the RSD instance and use a system of equations. If the system is solved, it can be said that a solution is found for the RSD instance.

The algebraic attack demonstrated in [1] considers two cases comparing the number of equations and those of variables. When the number of equations is greater than or equal to those of variables, it is called *an overdetermined case*. Otherwise, it is called *an underdetermined case*.

For an instance of the RSD problem with an $[n, k]$ code over $\mathbb{F}_{q^m}$ and an error vector with rank weight r, it is an overdetermined case if the inequality (1) is satisfied.

$$m\binom{n-k-1}{r} \geq \binom{n}{r}. \tag{1}$$

The complexity in this case is as follows.

$$\mathcal{O}\left(m\binom{n-k-1}{r}\binom{n}{r}^{\omega-1}\right). \tag{2}$$

For an overdetermined case, if we can find a nonzero p satisfying the inequality (3), it is called *a super-overdetermined case*.

$$m\binom{n-p-k-1}{r} \geq \binom{n-p}{r}. \tag{3}$$

For the maximal value of p satisfying (3), we can reduce the attack complexity (2) as follows.

Summary of the first attack(2023.4)

1.4 Structural attacks

- Some typo errors from the calculation of the security level exists.
- Rearranging the parameters can increase the security level.
- For example, increasing the size of degree of inner polynomial can increase the complexity by the brute-force algorithm, which increase overall performance

1.4 Structural attack

The complexity of a structural attack using the structure of BII-LRPC codes is suggested in [2] as follows.

$$S_S = \left(\frac{n}{b}\right)^2 m^3 q^{(b-1)m+d\lceil\frac{m}{2}\rceil-m-\frac{n}{b}}. \quad (4)$$

It seems that there is a typo in (4) because it is said in [2] that S_S will be the same to $S_{S_{ROLLO}}$ if $b = 1$, but it does not.

$$S_{S_{ROLLO}} = n^3 m^3 q^{d\lceil\frac{m}{2}\rceil-m-n}.$$

Thus we used the following complexity instead of (4), and obtained the cost of the attack in the table below.

$$S'_S = \left(\frac{n}{b}\right)^3 m^3 q^{(b-1)m+d\lceil\frac{m}{2}\rceil-m-\frac{n}{b}}.$$

Security Level	q	n	m	d	b	S'_S
128	2	74	67	2	2	2^{65}
192	2	86	79	3	2	2^{112}
256	2	106	97	3	2	2^{131}

Summary of the first attack(2023.4)

2. Direct attack

$$\begin{aligned} P_P^{-1}P_H \bmod P^b &= (P_O\Psi(P_I))^{-1}(P_O\Psi(\mathbf{z}) + P_N P) && \bmod P^b \\ &= \Psi(P_I)^{-1}\Psi(\mathbf{z}) + P_P^{-1}P_N P && \bmod P^b. \end{aligned}$$

Since P^b is a multiple of P, we can take mod$P$ to the equation above. Then, using $\Psi(\mathbf{z}) \bmod P = P_I\mathbf{x}^{-1}\mathbf{y} \bmod P$, we have the following equation:

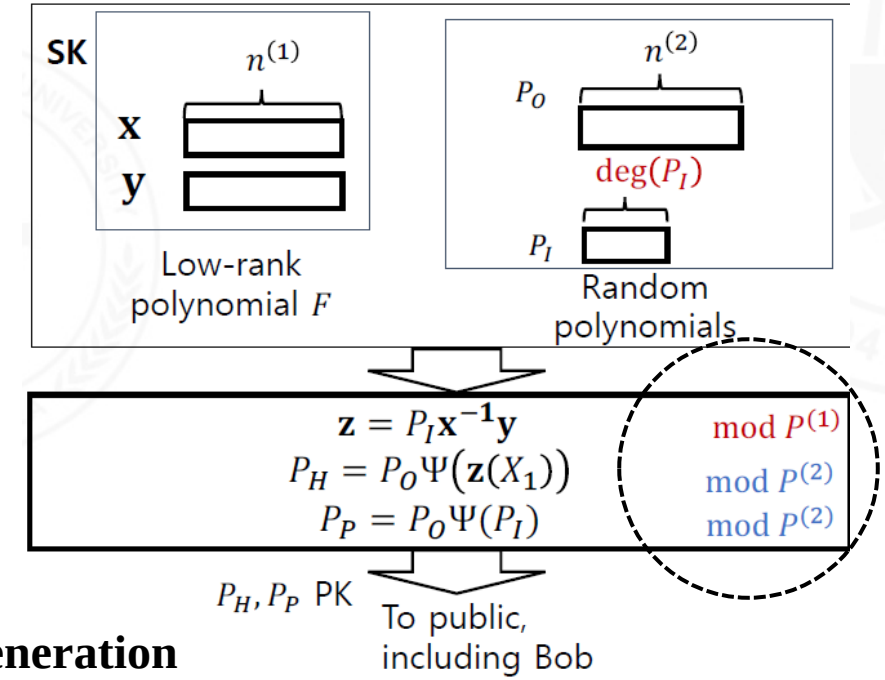
$$\begin{aligned} [P_P^{-1}P_H \bmod P^b] \bmod P &= [\Psi(P_I)^{-1}\Psi(\mathbf{z}) + P_P^{-1}P_N P \bmod P^b] \bmod P \\ &= \boxed{\Psi(P_I)^{-1}P_I}\mathbf{x}^{-1}\mathbf{y} && \bmod P \\ &= \mathbf{x}^{-1}\mathbf{y} && \bmod P. \end{aligned}$$

- After P reduction for the inverse operation over P^b modulus $\Psi(P_I)^{-1}$, the result can P reduction.
- Solution: The following attack can be avoided by the following method
  - The problem can be solved by changing the **modulus.** In specific, we can avoid the direct attack by modifying the moduli of P and P^b to those of $P^{(1)}$ and $P^{(2)}$

Improvements on Layered ROLLO(2023.5)

Summary of modification

1. Avoiding attack by using generalized modulus $P^{(2)}$
2. Increasing rank weight of error vector r



Key generation

- **Suggested parameter for the ex**

Name	q	n	m	r	'	P	b
Existing-128	2	74	67	**3**	2	47	2
Existing-192	2	86	79	**4**	3	39	2
Existing-256	2	106	97	**5**	3	31	2

- **Suggested parameter for the Mo**

1	q	$n^{(1)}$	$n^{(2)}$	m	r	d	Deg(PI
Modified-128	2	37	61	67	6	2	11
Modified-192	2	43	71	79	7	3	15
Modified-256	2	53	103	97	7	3	20

Suggest parameter

Summary of the second attack(2023.9)

- Two different types of attacks are efficient by insufficiently protected polynomial P_E(low-rank message) and P_P(PK) and thus, we do the following:

Comment No.	Comments	Modification
1.	Improved attack on RSD can r educe the SL	Rearranging the code parameters with the increased r
2	The scheme can be reduced to the initial ROLLO problem (broken)	Modify the structure of PK and SK, which masks P_P polynomials in PK

Summary of the second attack(2023.9)

1. Improved Attacks on RSD

- **By the overdetermined case, the modified parameter is still low for obtaining sufficient SL by variants on RSD**
  (e.g S_W by Wiederman Alg.)
  ➔ **Solution:** Increasing r more

- **Recall that influence on large r is relatively small** because the decryption complexity has a procedure of multiplying P_O, and P_I and recovering s **x** and **y** with low rank weight d.

In our work, we computed the costs of RSD attacks on modified Layered-ROLLO-I considering also improvements discussed in https://link.springer.com/chapter/10.1007/978-3-030-64837-4_17

What we found is that the parameters provided in your slides for the modified version are still not satisfying the corresponding security parameters. The costs of the attacks are reported in the following table

Security	BBC+
128	93.72
192	105.90
256	114.10

This shows that the suggested parameters are still too low to fight off RSD attacks.

$$S_W = \mathcal{O}\left(D_b A_b^2\right), \text{ where}$$

$$A_b = \sum_{j\in[1,b]} \binom{n}{r}\binom{mn+1}{j},$$

$$B_b = \sum_{j\in[1,b]} m\binom{n-1}{r}\binom{mn+1}{j},$$

$$C_b = \sum_{j\in[1,b]}\sum_{i\in[1,j]} (-1)^{i+1}\binom{2n}{r+i}\binom{m+i-1}{i}\binom{mn+1}{j-i},$$

$$D_b = \frac{B_b\binom{n+r+1}{r} + C_b(mn+1)(r+1)}{B_b + C_b}.$$

Summary of the second attack(2023.9)

2. For the previous direct attack,

In the first comment in 23/04/10

$$\begin{aligned} P_P^{-1}P_H \bmod P^b &= (P_O\Psi(P_I))^{-1}(P_O\Psi(\mathbf{z}) + P_N P) \quad \bmod P^b \\ &= \Psi(P_I)^{-1}\Psi(\mathbf{z}) + P_P^{-1}P_N P \quad \bmod P^b. \end{aligned}$$

Since P^b is a multiple of P, we can take mod$P$ to the equation above. Then, using $\Psi(\mathbf{z}) \bmod P = P_I\mathbf{x}^{-1}\mathbf{y} \bmod P$, we have the following equation:

$$\begin{aligned} [P_P^{-1}P_H \bmod P^b] \bmod P &= [\Psi(P_I)^{-1}\Psi(\mathbf{z}) + P_P^{-1}P_N P \bmod P^b] \bmod P \\ &= \Psi(P_I)^{-1}P_I\mathbf{x}^{-1}\mathbf{y} \quad \bmod P \\ &= \mathbf{x}^{-1}\mathbf{y} \end{aligned}$$

$\left(\boldsymbol{x^{-1}y} \bmod P^{(1)}\right)$

- By the new modulus $P^{(1)}$ and $P^{(2)}$, the inverse value of $\left(\left(P_P^{-1}P_H \bmod P^{(2)}\right) \bmod P^{(1)}\right)$ does not match $\mathbf{x^{-1}y} \bmod P^{(1)}$
- Direct attack can be avoided from this

Summary of k-PQC forum comments

For a new modification on direct attack

- By modification, we showed that

$$R = P_P^{-1} P_H$$

$$= \Psi(P_I^{-1})(P_I \mathbf{x^{-1}y} \bmod P^{(1)}) \bmod P^{(2)}$$

does not represent $\mathbf{z} = \mathbf{x^{-1}y} \bmod P^{(1)}$ directly

- However, we have the equation

$\Psi(P_I) M_R \bmod P^{(2)} = \Psi(\mathbf{z}) \bmod P^{(2)}$ for a matrix M_R

- The proposed attack tries to recover the unknown from matrix form M_R by using low-degree polynomial P_I and $\mathbf{z}$

Furthermore, we managed to reduce any instance of the modified Layered-ROLLO-I to an instance of ROLLO-I which uses a smaller code. Also, the reduction recovers the PO and PI parts of the private key, showing a leakage in the system. We do this by computing R=PH/PP and considering the system of equations ₩Psi(PI)RM = ₩Psi(z), where RM is the matrix of the multiplication by R mod P2.

The choices of degrees of PI and z ensure that the system has as a solution the representatives of PI and z modulo P1. Actually, we recover k*PI, where k is a field constant, by computing the left kernel of the matrix consisting of the last deg(PI)+1 columns of RM. Compute now PO/k from PP/(k*PI) and k*z from PH/(PO/k). Finally, compute y/x from (k*z)/(k*PI).

The attached SAGE script computes k*PI on a Linux Mint VM in less than 0.05 seconds for security 128, around 0.12 seconds for security 192 and around 0.36 seconds for security 256.

Recall: Polynomial Masking Method

- Here, let the *polynomial masking* be an operation to $P'_O = P'_O + P_N P^{(1)}$ for a small-degree polynomial P'_O, small-degree modulus $P^{(1)}$, and random polynomial P_N which was used to hide the structure of small structure of codes or scheme(mainly degree) from the PK
- We removed masking in the modified form. However, the related operation are remained as remarks in the implementation codes.

i) Key generation: Let F be a set of $\frac{n}{b}$-tuple vectors with rank weight d. Alice selects two $\frac{n}{b}$-tuple random vectors $\mathbf{x}, \mathbf{y} \in F$. Also, we denote the random degree-$(b-1)$ primitive polynomial $P_I \in \mathbb{F}_{q^m}[X]/ < P >$ and degree-n polynomials $P_O, P_N \in \mathbb{F}_{q^m}[X]/ < P^b >$. For $\mathbf{x}$ and $\mathbf{y}$, we derive the $\frac{n}{b}$-tuple vector $\mathbf{z} = P_I\mathbf{x}^{-1}\mathbf{y}$ mod P.
Finally, Alice constructs a PK as two polynomials of $P_P = P_O\Psi(P_I) \mod P^b, P_H = P_O\Psi(\mathbf{z}(X_1)) + P_N P$ mod P^b for the SK as the two vectors $\mathbf{x}, \mathbf{y}$ and two polynomials P_I, and P_O.

- Generate $\mathbf{z}$ and P_H as
  - $\mathbf{z} = \left(P_I\mathbf{x}^{-1}\mathbf{y}\right) \bmod P^{(1)}$
  - $P_H = P_O\left(\Psi\left(\mathbf{z}(X_1)\right) \bmod P^{(2)}\right)$
- Finally, construct SK and PK as
  - **PK:** P_H, $P_P = P_O(\Psi(P_I)) \bmod P^{(2)}$

(NOTE: We use an additional key size by P_P, whic

Initial Layered-ROLLO-I(22.10.31)
P_H was masked. i.e. $P_H = P_O\Psi(\mathbf{z}) \rightarrow P_O\Psi(\mathbf{z}) + P_N P$

Modified Layered-ROLLO-I(23.5.19)
Both P_P, P_H were not masked
(by the guess that coprimality between moduli cannot be solved generally..)

Improvements on Layered ROLLO(2023.9)

Summary of modification

1. Rearranging the code parameters with the increased r
2. Modify the structure of PK and SK, which consider $P^{(1)}$ as SK and masks P_P polynomial in PK

- **Suggested parameter for the first modified Laye**

Inst	q	$n^{(1)}$	$n^{(2)}$	m	r	d	Deg(Pl)	Sec. (Conv.Gen)	Sec. (Conv.
Modified-128	2	37	61	67	6	2	11	377.826628	197.826
Modified-192	2	43	71	79	7	3	15	480.190137	268.690
Modified-256	2	53	103	97	7	3	20	580.9835	323.483

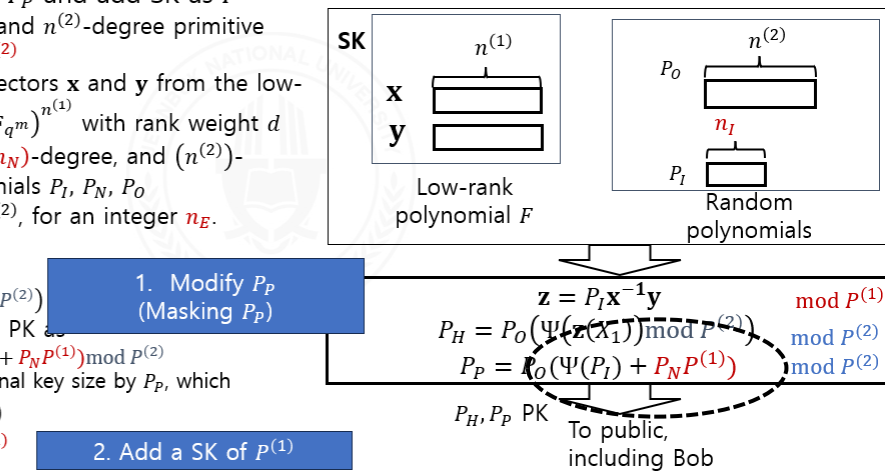
- **Suggested parameter for the new Layered ROLLO**

Name	q	$n^{(1)}$	$n^{(2)}$	n_N	n_I	m	r	d	(S_R
New-128	2	37	61	1	4	67	7	2	
New-192	2	43	71	2	4	79	9	2	
New-256	2	53	103	4	4	97	12	2	

Suggest parameter

Modified Layered ROLLO-I: Procedures

1. **Key generation: Modify** P_P and add SK as $P^{(1)}$
   - Select two $n^{(1)}$ -degree and $n^{(2)}$-degree primitive polynomials $P^{(1)}$, and $P^{(2)}$
   - Generate two random vectors $\mathbf{x}$ and $\mathbf{y}$ from the low-rank $\mathbb{F}_q$-subspace $F \in (\mathbb{F}_{q^m})^{n^{(1)}}$ with rank weight d
   - Generate (n_I)-degree, (n_N)-degree, and $(n^{(2)})$-degree random polynomials P_I, P_N, P_O with $n^{(1)} + n_E + n_N < n^{(2)}$, for an integer n_E.
   - Generate $\mathbf{z}$ and P_H as
     - $\mathbf{z} = (P_I\mathbf{x}^{-1}\mathbf{y}) \bmod P^{(1)}$
     - $P_H = P_O(\Psi(\mathbf{z}(X_1)) \bmod P^{(2)})$
   - Finally, construct SK and PK as
     - **PK:** P_H, $P_P = P_O(\Psi(P_I) + P_N P^{(1)}) \bmod P^{(2)}$
       (NOTE: We use an additional key size by P_P, which amounts to $\lceil \frac{n\log_2 m}{8} \rceil$[Byte])
     - **SK:** $\mathbf{x}, \mathbf{y}$, P_O, P_I, and $P^{(1)}$



Key generation

Summary of the third and fourth attack(2023.10)

- Third attacks are based on the fixed location of error polynomial $P_{E,1}$ and $P_{E,2}$ on the ciphertext

Comment No.	Comments	Modification
1.	new attacks by fixed nonzero location on $P_{E,1}$,$P_{E,2}$	Polynomial masking on CT Setting the different degrees on $P_{E,1}$ and

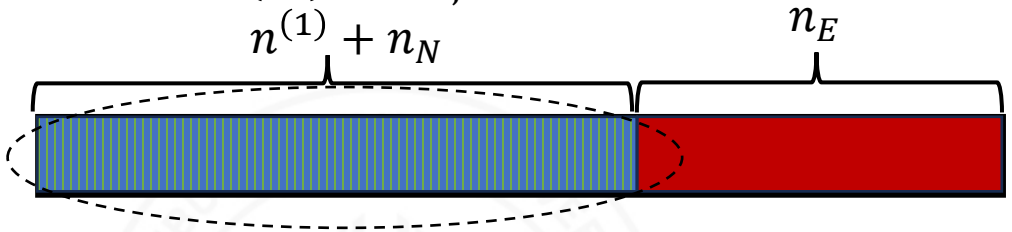
- Fourth attacks by combining three linear equations by each PK and CT

Comment No.	Comments	Modification
1.	New attacks by combining t hree linear equations by ea ch PK and CT	Modifying the code parameter of LROLLO-128 and LROLLO-192 where the proposed attack is not applied

Summary of the third attack(2023.10)

$$P_P = \Psi(P_I) + P_{N,A}P^{(1)}$$

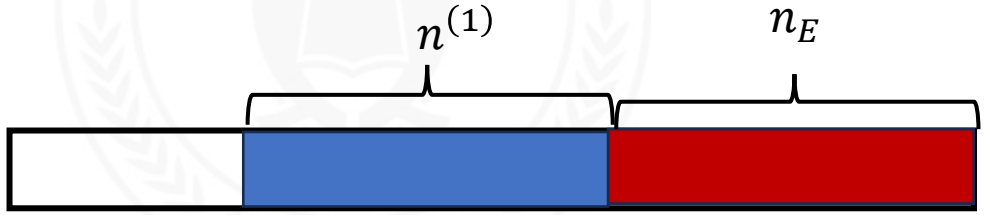
$$P_P P_{E,1} = P_O \times$$



$$P_{E,1}$$

$$+ \quad \Psi(\mathbf{z}(X_1)) \quad \text{Masked part}$$

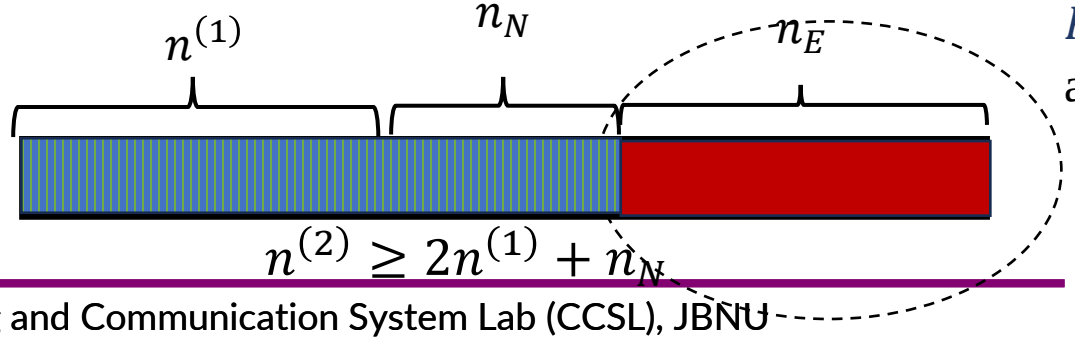
$$P_H(P_{E,2}) = P_O \times$$



$$P_{E,2}$$

$$+ \quad =$$

$$= \quad P_C \quad P_O \times$$



The attacker can exploit the location from the equation $P_H^{-1}P_C = P_P P_H^{-1} P_{E,1} + P_{E,2}$ and Prange algorithm

Improvements on Layered ROLLO(2023.10)

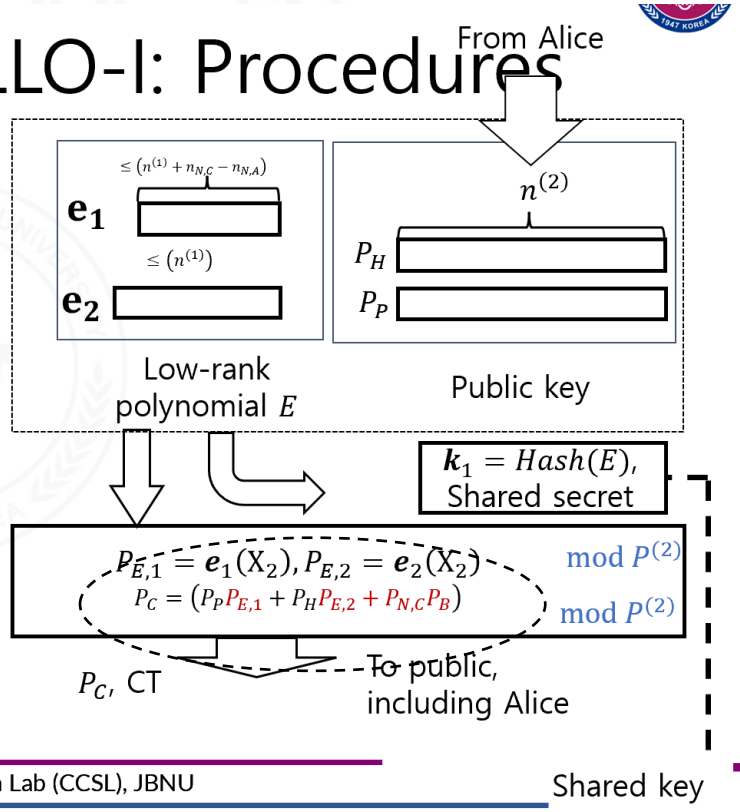
Summary of modification

1. Adding a new public key $P_B = P_{N,B}P^{(1)}$, which is used for generating CT in encapuslation.

Modified Layered ROLLO-I: Procedures

2. Encapsulation: Adding polynomial masking on the error vector $P_{N,C}P_B$

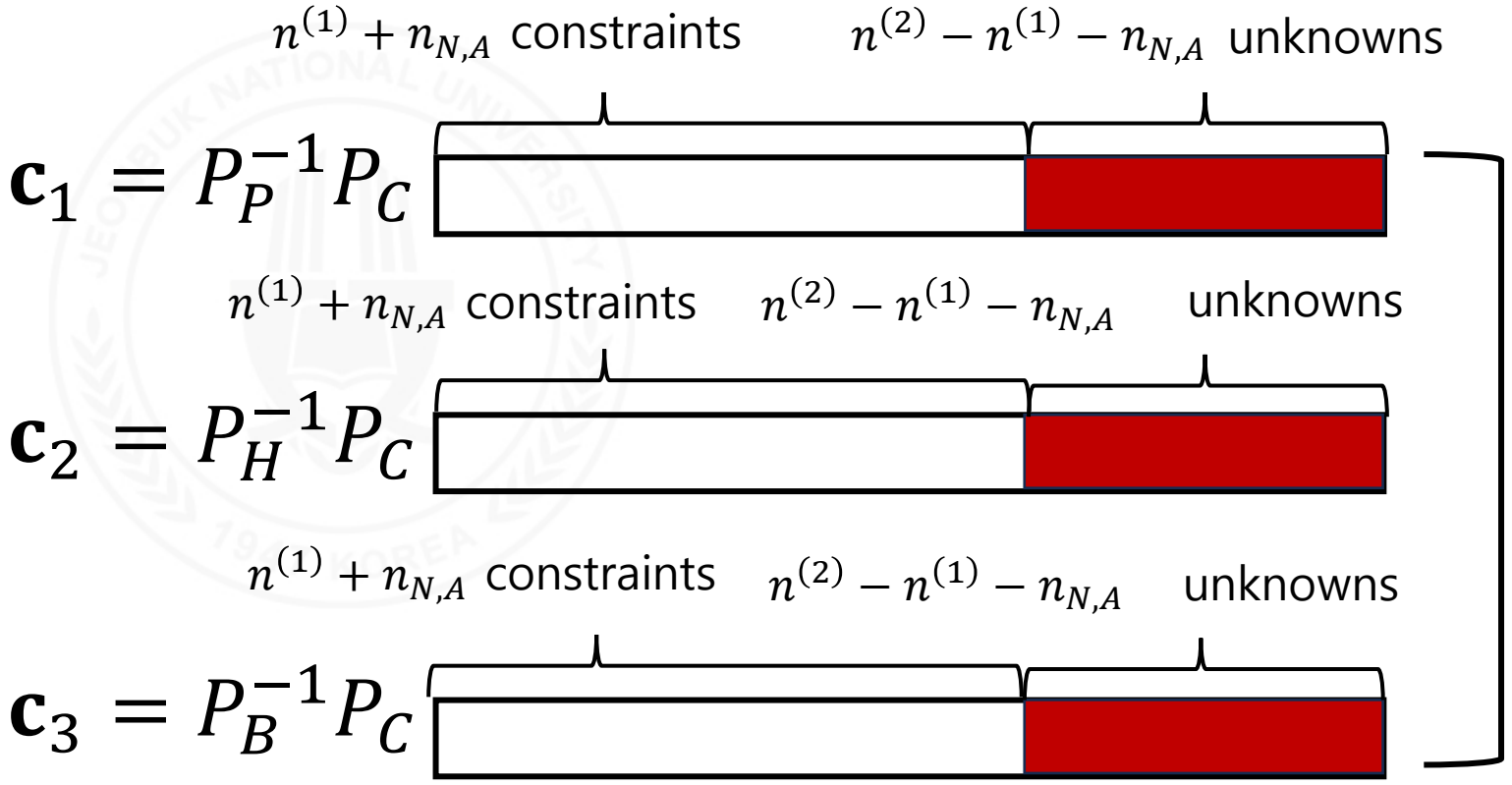
- Generate low-rank $\mathbb{F}_q$-subspace $E \in \left(\mathbb{F}_{q^m}\right)^{n^{(2)}}$ with rank weight r with the last $n^{(2)} - n^{(1)} + n_{N,A}$ nonzero elements
- Generate two error vectors $\boldsymbol{e}_1, \boldsymbol{e}_2 \in E$ and corresponding $(n^{(2)} - n^{(1)} - n_{N,A})$-degree polynomials
- $P_{E,1} = \mathbf{e}_1(X_2)$ and $P_{E,2} = \mathbf{e}_2(X_2)$
- Obtain CT polynomial P_C as $P_C = (P_P P_{E,1} + P_H P_{E,2} + P_{N,C}P_B) \bmod P^{(2)}$ for $(n^{(2)} - n^{(1)} - n_{N,A})$ −degree polynomials $P_{N,C}$
- Obtain $\mathbf{k}_1 = Hash(E)$ to have a shared secret (SS)



Summary of the fourth attack(2023.10)

From the linear combinations between the three CT, $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$.

- The attacker may make a three overdetermined equations with unknowns $3\left(n^{(2)} - n^{(1)} - n_{N,A}\right)$ and constraints $3\left(n^{(1)} + n_{N,A}\right)$.
- Actually, CT has at most $n^{(2)}$ linearly independent equations and thus, the equation is determined when $3\left(n^{(2)} - n^{(1)} - n_{N,A}\right) < n^{(2)}$
- Otherwise, the attack should require an additional complexity by guessing if $3\left(n^{(2)} - n^{(1)} - n_{N,A}\right) > n^{(2)}$



Improvements on Layered ROLLO(2023.11)

Summary of modification

- Size of $n^{(2)}$ is increased in order to prevent the fourth attack (for LROLLO-128 and LROLLO-192)
- KEM scheme and parameters of LROLLO-256 are unchanged

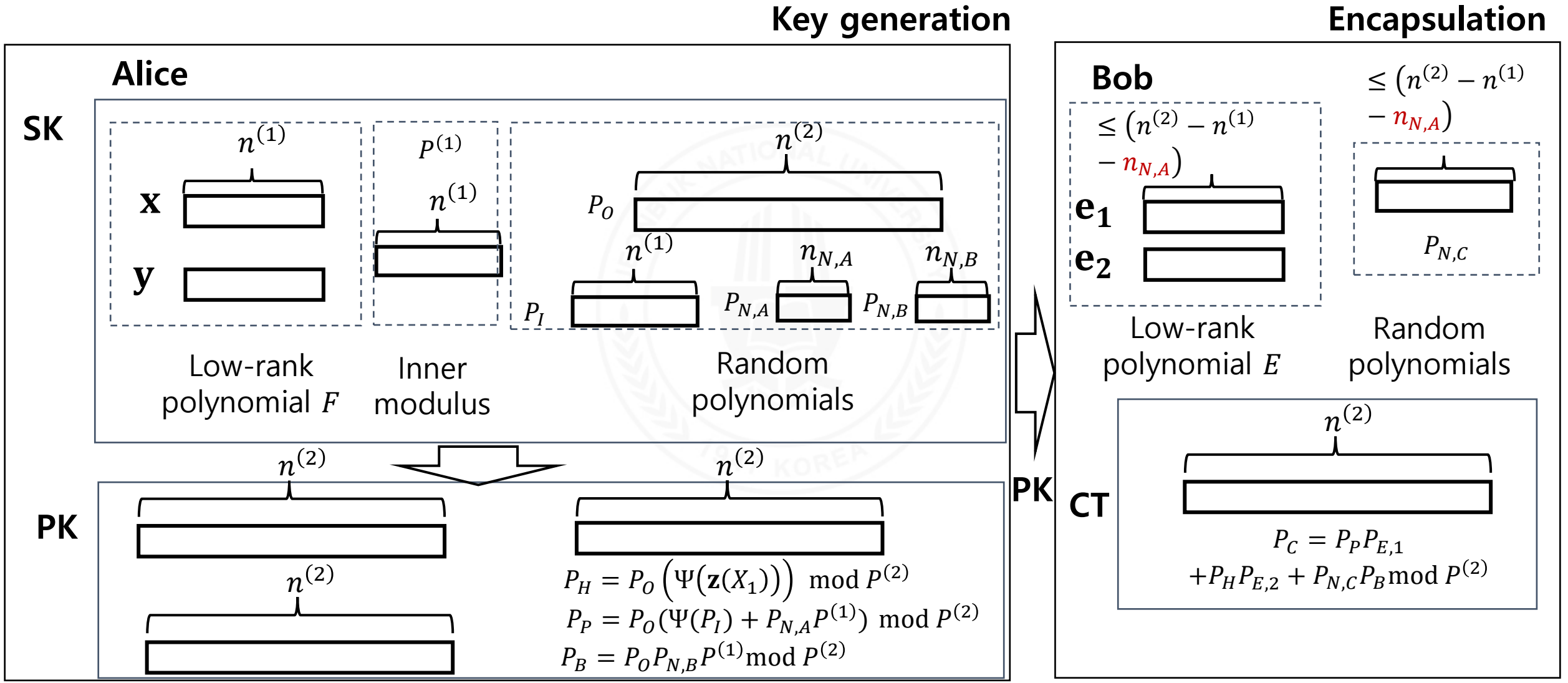
- **Suggested parameter for the new Layered ROLLO-I(2023.10),**

Name	q	$n^{(1)}$	$n^{(2)}$	$n_{N,A}$	m	r	d	SL. (S_R)	SL. (S_W)	SL. (S_C)	DFR	Pk size	ct size
New-128	2	37	61	4	67	7	2	154.6	≥268	1608	-23	1533	511
New-192	2	43	71	4	79	9	2	199.68	≥ 316	2212	-25	2106	702
New-256	2	53	103	4	97	12	2	273.4	≥ 388	4850	-29	3750	1250

- **Suggested parameter for the new Layered ROLLO-I(2023.11),**

Name	q	$n^{(1)}$	$n^{(2)}$	$n_{N,A}$	m	r	d	SL (S_R)	SL. (S_{D1})	SL. (S_{D2})	SL. (S_{D3})	DFR	Pk size	ct size
New-128	2	37	71	4	67	7	2	154.6	469	1943	1072	-23	1755	585
New-192	2	43	83	4	79	9	2	199.68	553	2765	1738	-25	2460	820
New-256	2	53	103	4	97	12	2	273.4	679	4365	3104	-29	3750	1250

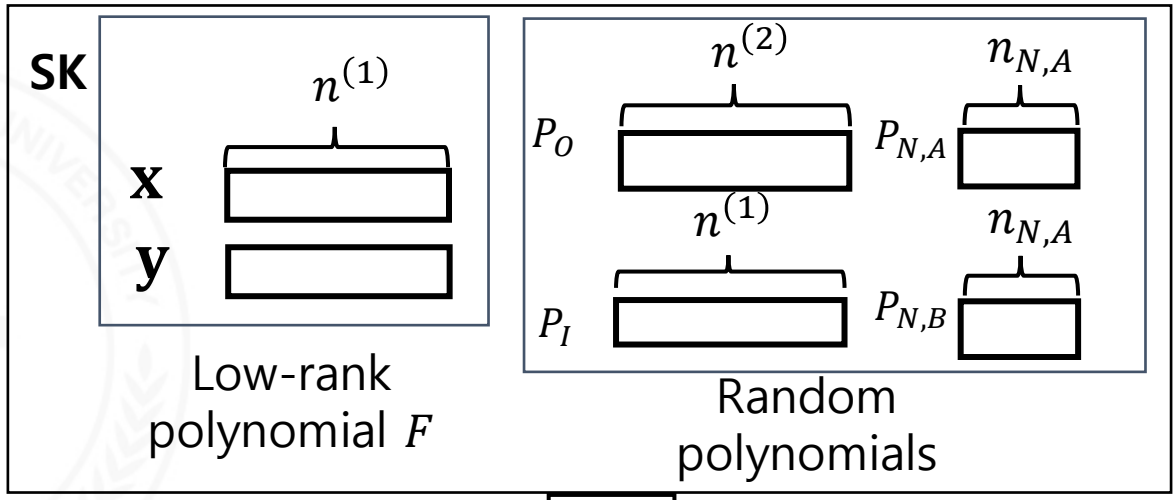
New Layered ROLLO-I: Major notation



New Layered ROLLO-I: Procedures

1. Key generation

- Select two $n^{(1)}$ -degree and $n^{(2)}$-degree primitive polynomials $P^{(1)}$, and $P^{(2)}$
- Generate two random vectors $\mathbf{x}$ and $\mathbf{y}$ from the low-rank $\mathbb{F}_q$-subspace $F \in (\mathbb{F}_{q^m})^{n^{(1)}}$ with rank weight d
- Generate $(n^{(1)})$-degree, $(n_{N,A})$-degree, $(n_{N,B} = n_{N,A})$-degree and $(n^{(2)})$-degree random polynomials P_I, $P_{N,A}$, $P_{N,B}$, P_O
- Generate $\mathbf{z}$ and P_H as
  - $\mathbf{z} = (P_I \mathbf{x}^{-1} \mathbf{y}) \bmod P^{(1)}$
  - $P_H = \left(P_O \Psi(\mathbf{z}(X_1))\right) \bmod P^{(2)}$
- Finally, construct SK and PK as
  - **PK:** P_H, $P_P = P_O(\Psi(P_I) + P_{N,A}P^{(1)}) \bmod P^{(2)}$, $P_B = P_O P_{N,B} P^{(1)}$ (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, and $P^{(1)}$



$$\mathbf{z} = P_I \mathbf{x}^{-1} \mathbf{y} \quad \bmod P^{(1)}$$
$$P_H = P_O\left(\Psi(\mathbf{z}(X_1))\right) \quad \bmod P^{(2)}$$
$$P_P = P_O(\Psi(P_I) + P_{N,A}P^{(1)}) \quad \bmod P^{(2)}$$
$$P_B = P_O P_{N,B} P^{(1)} \quad \bmod P^{(2)}$$

To public, including Bob P_H, P_P

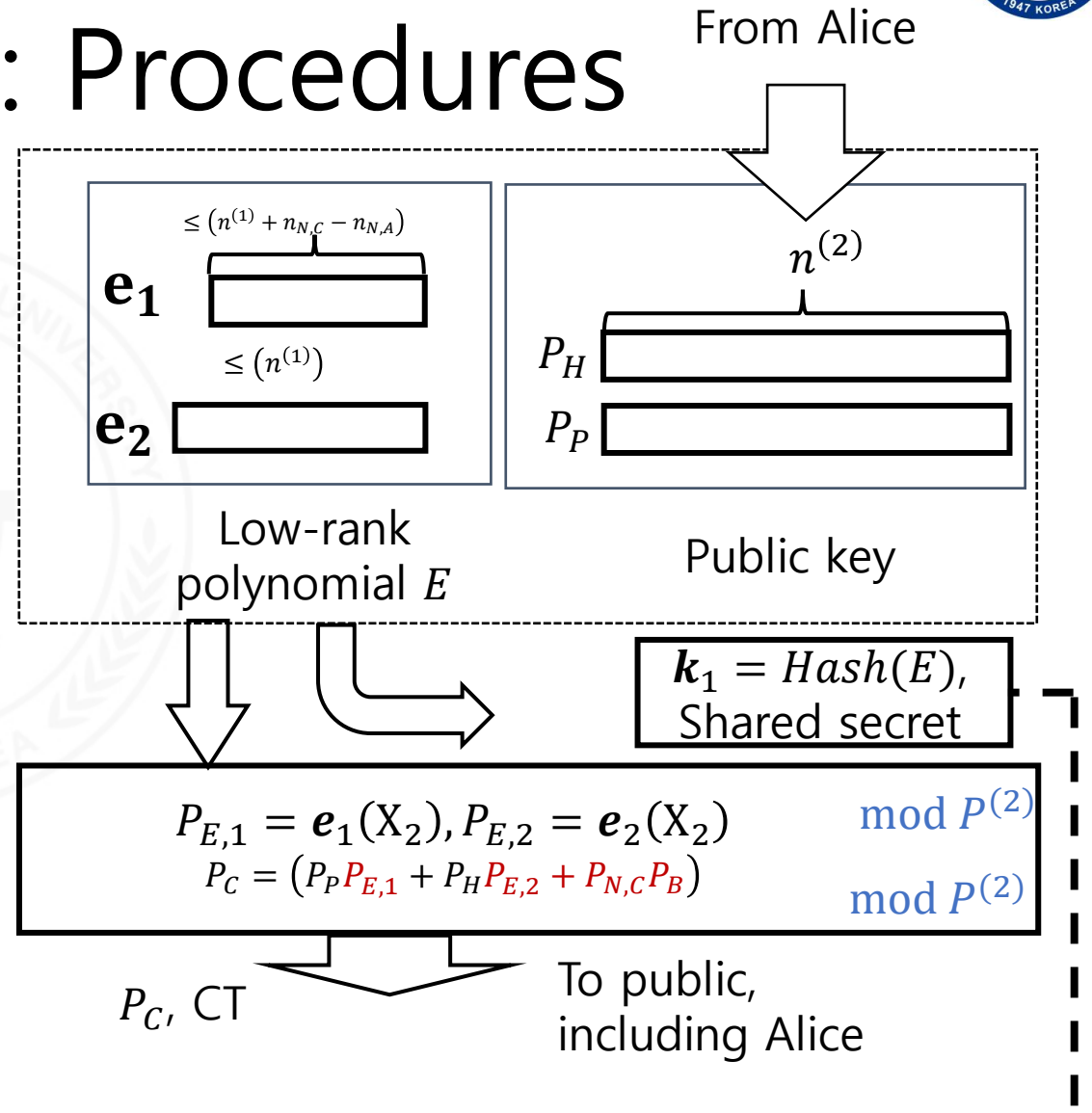
New Layered ROLLO-I: Procedures

2. Encapsulation

- Generate low-rank $\mathbb{F}_q$-subspace $E \in \left(\mathbb{F}_{q^m}\right)^{n^{(2)}}$ with rank weight r with the last $n^{(2)} - n^{(1)} + n_{N,A}$ nonzero elements
- Generate two error vectors $\boldsymbol{e}_1, \boldsymbol{e}_2 \in E$ and corresponding $\left(n^{(2)} - n^{(1)} - n_{N,A}\right)$-degree polynomials
- $P_{E,1} = \mathbf{e}_1(X_2)$ and $P_{E,2} = \mathbf{e}_2(X_2)$
- Obtain CT polynomial P_C as

$P_C = \left(P_P P_{E,1} + P_H P_{E,2} + P_{N,C} P_B\right) \bmod P^{(2)}$ for $(n^{(2)} - n^{(1)} - n_{N,A})$ −degree polynomials $P_{N,C}$

- Obtain $\mathbf{k}_1 = Hash(E)$ to have a shared secret (SS)

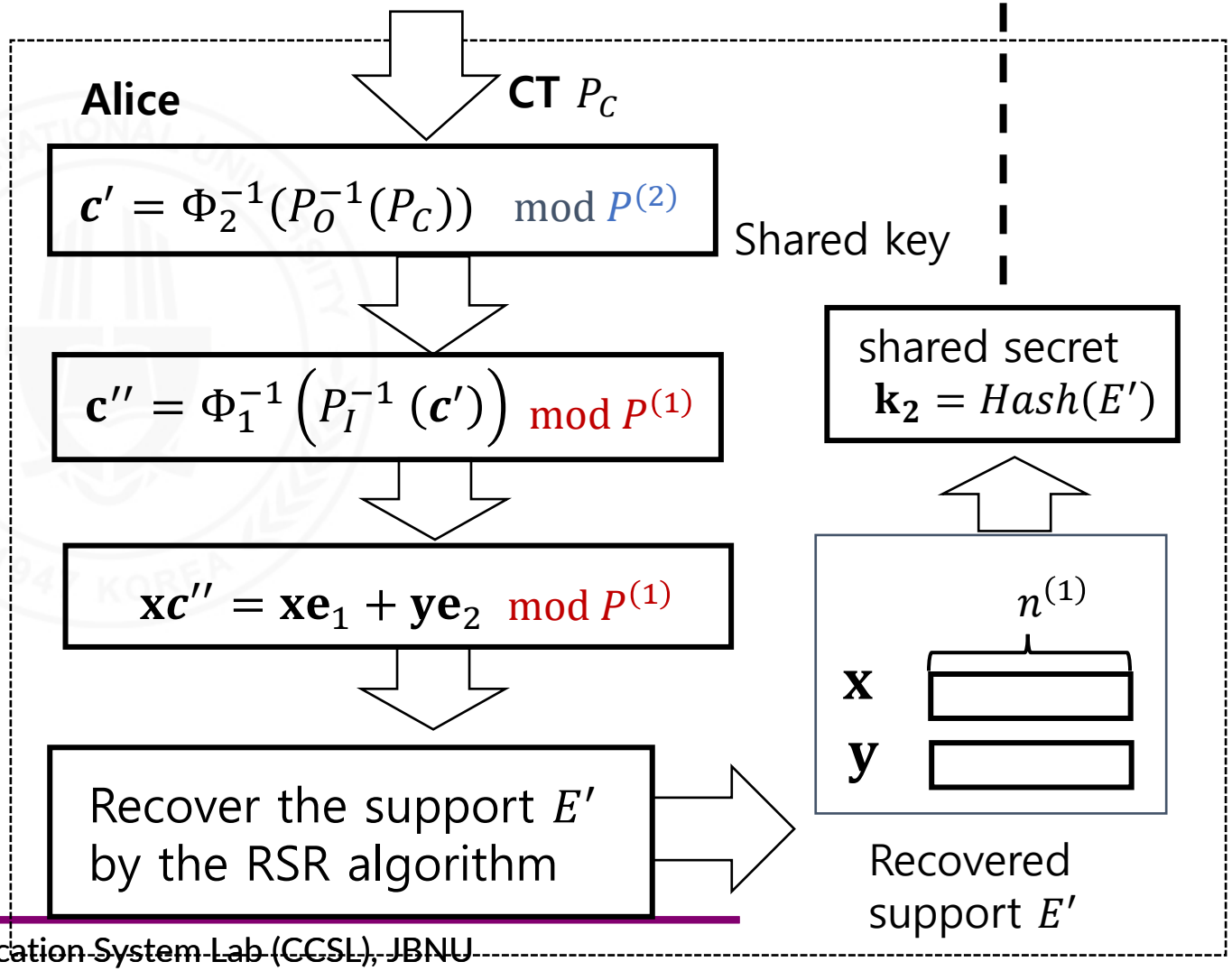


New Layered ROLLO-I: Procedures

3. Decapsulation

- Obtain the codeword $\mathbf{xc}'' = \mathbf{xe}_1 + \mathbf{ye}_2 \bmod P^{(1)}$ from P_C by
  - $\boldsymbol{c}' = \Phi_2^{-1}(P_O^{-1}(P_C)) \bmod P^{(2)}$
  - $\boldsymbol{c}'' = \Phi_1^{-1}\left(P_I^{-1}(\boldsymbol{c}') \bmod P^{(1)}\right)$
  - $\mathbf{xc}'' \bmod P^{(1)} = \mathbf{xe}_1 + \mathbf{ye}_2 \bmod P^{(1)}$

- From $\mathbf{xc}'' \bmod P^{(1)}$, recover the support E' by the RSR algorithm
- Derive shared key $\mathbf{k_2} = Hash(E')$



Security Analysis for new-LROLLO

In summary, the new attack scenarios have SL either of

1) Generic attacks by rank support decoding S_R (by the first and second attacks)
2) The first direct attack S_{D1} (Using two PKs, P_H, P_P, and other pairs, by the second attacks)

$$S_{D1} = O\left(q^{(2n^{(1)}-n^{(2)}+n_{N,A})m}\right)$$

3) Second direct attack S_R(Using two PKs (P_H, P_P and other pairs) and CT(P_C), by the third attacks)

$$S_{D2} = O\left(q^{(n^{(2)}-n^{(1)}-n_{N,A}-1)m}\right)$$

4) Third direct attack(Using three PKs and CT, P_H, P_P, P_B and CT(P_C), by the fourth attacks)

$$S_{D3} = O\left(q^{(3(n^{(2)}-n^{(1)}-n_{N,A}-1)-n^{(2)})m}\right)$$

New Suggested Parameters

- **Suggested parameter for the existing ROLLO-I**

Name	q	n	m	r	d	SL (Conv. Gen)	SL (Conv. Stru.)	DFR	Pk size	ct size
ROLLO-I-128	2	83	67	**7**	8	207.648687	155.3233859	-27	696	696
ROLLO-I-192	2	97	79	**8**	8	278.968813	178.7110808	-33	958	958
ROLLO-I-256	2	113	97	**9**	9	383.623107	266.7602754	-32	1371	1371

- **Suggested parameter for the first submission(2022.10)**

Name	q	$n^{(1)}$	$n^{(2)}$	n_N	n_I	m	r	d	SL. (S_R)	SL. (S_C)	DFR	Pk size	ct size
LROLLO-128	2	37	**74**	N.A.	**2**	67	**2**	2	27.89	Broken	-31	1240	620
LROLLO-192	2	43	**86**	N.A.	**2**	79	**3**	2	42.38	Broken	-35	1857	979
LROLLO-256	2	53	**106**	N.A	**2**	97	**3**	2	44.37	Broken	-38	2571	1286

- **Suggested parameter for the new Layered ROLLO-I(2023.11),**

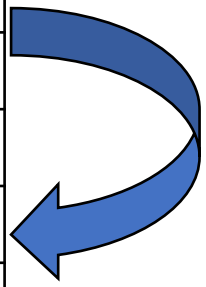
Name	q	$n^{(1)}$	$n^{(2)}$	$n_{N,A}$	m	r	d	SL (S_R)	SL. (S_{D1})	SL. (S_{D2})	SL. (S_{D3})	DFR	Pk size	ct size
New-128	2	37	**71**	**4**	67	**7**	2	**154.6**	**469**	**1943**	**1072**	**-23**	1755	585
New-192	2	43	**83**	**4**	79	**9**	2	**199.68**	**553**	**2765**	**1738**	**-25**	2460	820
New-256	2	53	**103**	**4**	97	**12**	2	**273.4**	**679**	**4365**	**3104**	**-29**	3750	1250

- From the larger $n^{(2)}$, the PK and CT size becomes larger than the first submission.

Code performance analysis

- **Performance measure:** The number of CPU processing cycle for key generation, encapsulation, and decapsulation on the simulation environments on CPU
  - CPU: Intel(R) Xeon(R) w7-3465X, 256GB, x4 4800GB/s, Hynix DDR5

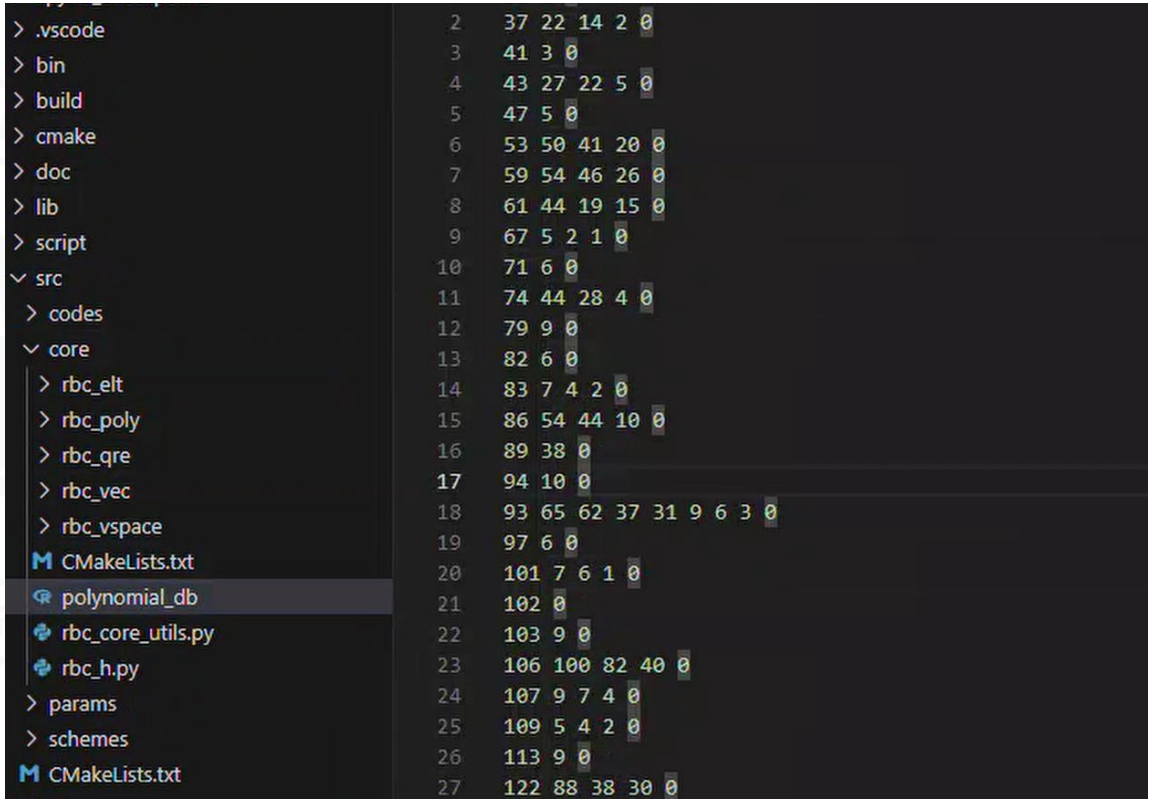
Instance	Key generation	Encapsulation	Decapsulation	Total
ROLLO-I-128	2,331,728	327,392	4,737,172	7,396,292
ROLLO-I-192	2,537,876	546,316	6,248,864	9,333,056
ROLLO-I-256	3,436,412	428,696	10,493,536	14,358,644
New-128	1,417,548	495,412	2,440,012	4,352,972
New-192	1,848,076	823,500	2,789,864	5,461,440
New-256	2,224,488	767,080	4,263,728	7,255,296



30-50% cycle reduction

On the implementation of modulus $P^{(1)}$ operation

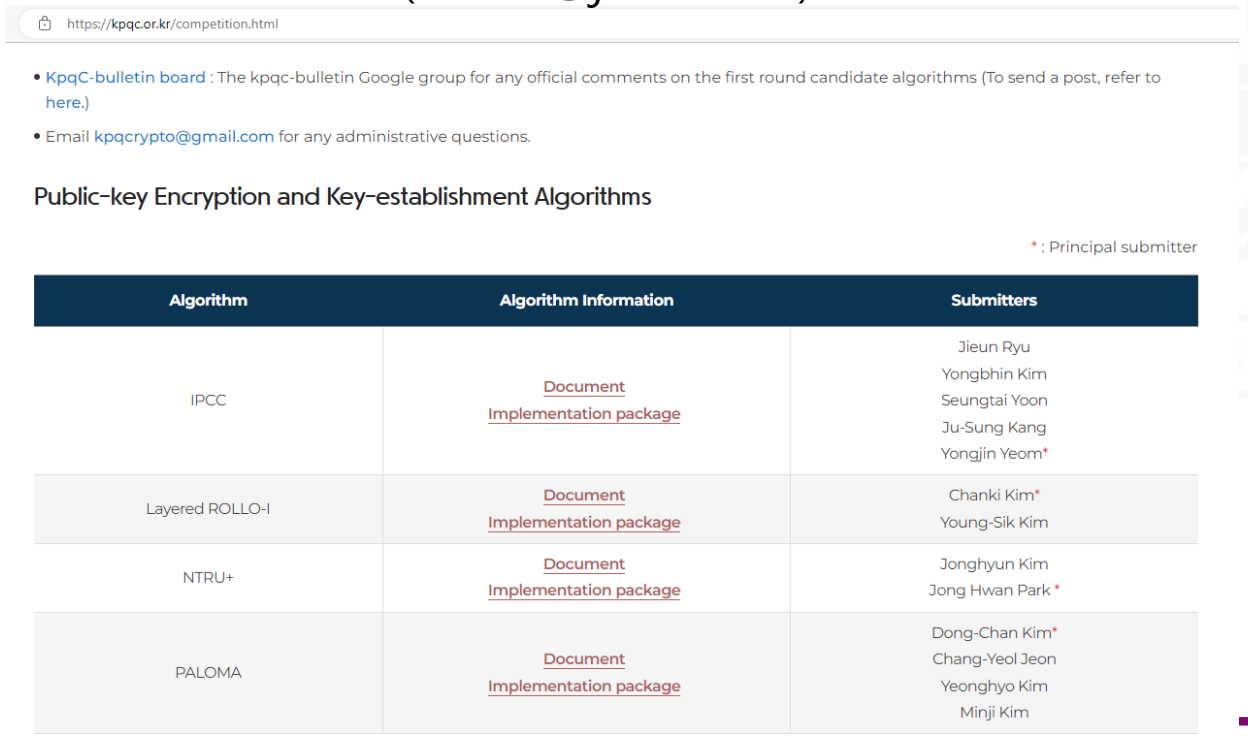
- From the key generation phase, the inner modulus polynomial $P^{(1)}$ is classified as a class of SK, which can be a fixed or random value for each KEM operation.
- Note that the corresponding SL can be lowered from the low-degree polynomial $\frac{P_B}{P^{(1)}}$ when modulus polynomial $P^{(1)}$ can be found by attacker.
  - For the fixed case, a new attack for recovering $P^{(1)}$ can be devised by collecting prior PKs or CTs and thus, using random values for $P^{(1)}$ is desirable for security.
- Still, $P^{(1)}$ in the corresponding C code is implemented as a fixed polynomial, which follows the initial source code RBC(Rank-based cryptography) in the ROLLO-I.
- We are trying to modify it with minimizing the additional processing speed (i.e. processing cycle) until the end of 2nd submission. There seems to be some options to be applied for performance optimization.



List of fixed primitive polynomial in the implementation codes

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)
- Kpqc-Bulletin: https://groups.google.com/g/kpqc-bulletin/
- Layered-ROLLO-I Homepage(https://sites.google.com/view/ccsl-jbnu/research/layered-rollo) or contact me (carisis@jbnu.ac.kr)



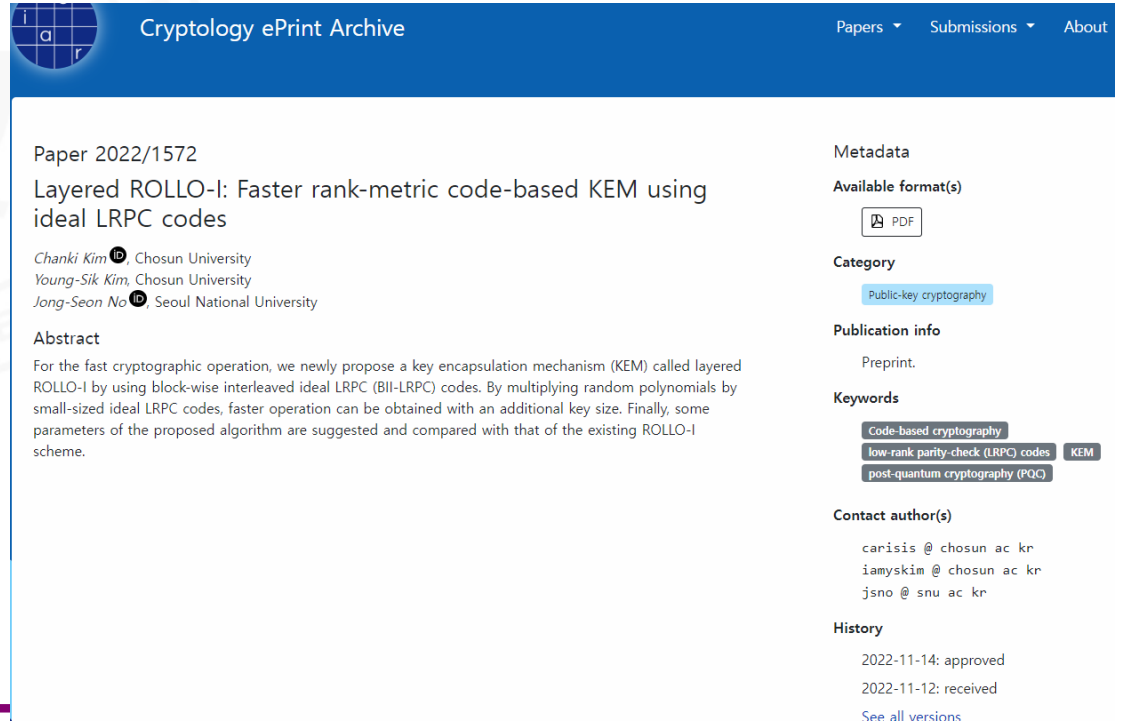
https://kpqc.or.kr/competition.html

- 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

Preliminaries: Polynomial Ring Arithmetic

- **Polynomial (quotient) ring** $\mathbb{F}_{q^m}[X]/< P >$
  - For positive integer $n, m,$ 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

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

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

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

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

Vector-vector multiplication

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

Polynomial-vector multiplication

Preliminaries: Polynomial Ring Arithmetic

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

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

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

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

- **Equivalent circulant 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 $\mathcal{I}(\mathbf{u})\mathbf{u}'$

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

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 $\mathbf{v} = (v_1, v_2, \ldots, v_n) \in \left(\mathbb{F}_{q^m}\right)^n$
- (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 $C = \left\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{q^k}\right\} \subset \left(\mathbb{F}_{q^m}\right)^n$
- **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 $\mathrm{w}_H(C) = d$