Quantum Resistant Public Key Encryption Scheme RLCE

Yongge Wang

UNC Charlotte

April 13, 2018

UNC CHARLOTTE
College of Computing and Informatics

Outline

UNC CHARLOTTE
College of Computing and Informatics

Outline

Outline

UNC CHARLOTTE
College of Computing and Informatics

McEliece Scheme

McEliece Scheme (1978)

`Mc.KeySetup`: An $(n, k, 2t+1)$ linear Goppa code $\mathcal{C}$ with $k \times n$ generator matrix G_s. Public key: $G = SG_sP$. Private key: G_s Where S is random and P is permutation.

`Mc.Enc`$(G, \mathbf{m}, \mathbf{e})$. For a message $\mathbf{m} \in \{0, 1\}^k$, choose a random vector $\mathbf{e} \in \{0, 1\}^n$ of weight t. The cipher text $\mathbf{c} = \mathbf{m}G + \mathbf{e}$

`Mc.Dec`$(S, G_s, P, \mathbf{c})$. For a received ciphertext $\mathbf{c}$, first compute $\mathbf{c}' = \mathbf{c}P^{-1} = \mathbf{m}SG$. Next use an error-correction algorithm to recover $\mathbf{m}' = \mathbf{m}S$ and compute the message $\mathbf{m}$ as $\mathbf{m} = \mathbf{m}'S^{-1}$.

McEliece Security

- Broken ones: Niederreiter's scheme with Generalized Reed-Solomon Code Broken
- Broken ones: Wild Goppa code based McEliece, GRS-McEliece with random columns
- Unbroken ones: Original McEliece, MDPC/LDPC McEliece, Wang's RLCE

McEliece Security

- Broken ones: Niederreiter's scheme with Generalized Reed-Solomon Code Broken
- Broken ones: Wild Goppa code based McEliece, GRS-McEliece with random columns
- Unbroken ones: Original McEliece, MDPC/LDPC McEliece, Wang's RLCE

McEliece Security

- Broken ones: Niederreiter's scheme with Generalized Reed-Solomon Code Broken
- Broken ones: Wild Goppa code based McEliece, GRS-McEliece with random columns
- Unbroken ones: Original McEliece, MDPC/LDPC McEliece, Wang's RLCE

RLCE Key setup

`RLCE.KeySetup`. Let G_s be a $k \times n$ generator matrix for an $[n, k, d]$ linear code $\mathcal{C}$correcting at least t errors and $w \leq n$. Let $G_s P_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$ for a random permutation P_1

1. Let $G_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-w}, \mathbf{r}_0, \cdots, \mathbf{g}_{n-1}, \mathbf{r}_{w-1}]$ be a $k \times (n + w)$ matrix where $\mathbf{r}_i \in GF(q)^k$ are random
2. Let $A_i \in GF(q)^{2\times 2}$ be random 2×2 matrices. Let $A = \text{diag}[I_{n-w}, A_0, \cdots, A_{w-1}]$ be an $(n + w) \times (n + w)$ non-singular matrix.
3. The public key: $k \times (n + w)$ matrix $G = SG_1AP_2$ and the private key: (S, G_s, P_1, P_2, A) where S is random $k \times k$ matrix and P_2 is a permutation.

UNC CHARLOTTE
College of Computing and Informatics

RLCE Key setup

`RLCE.KeySetup`. Let G_s be a $k \times n$ generator matrix for an $[n, k, d]$ linear code $\mathcal{C}$correcting at least t errors and $w \le n$. Let $G_s P_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$ for a random permutation P_1

1. Let $G_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-w}, \mathbf{r}_0, \cdots, \mathbf{g}_{n-1}, \mathbf{r}_{w-1}]$ be a $k \times (n+w)$ matrix where $\mathbf{r}_i \in GF(q)^k$ are random
2. Let $A_i \in GF(q)^{2\times 2}$ be random 2×2 matrices. Let $A = \mathrm{diag}[I_{n-w}, A_0, \cdots, A_{w-1}]$ be an $(n+w) \times (n+w)$ non-singular matrix.
3. The public key: $k \times (n+w)$ matrix $G = SG_1AP_2$ and the private key: (S, G_s, P_1, P_2, A) where S is random $k \times k$ matrix and P_2 is a permutation.

RLCE Key setup

`RLCE.KeySetup`. Let G_s be a $k \times n$ generator matrix for an $[n, k, d]$ linear code $\mathcal{C}$correcting at least t errors and $w \leq n$. Let $G_s P_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$ for a random permutation P_1

1. Let $G_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-w}, \mathbf{r}_0, \cdots, \mathbf{g}_{n-1}, \mathbf{r}_{w-1}]$ be a $k \times (n + w)$ matrix where $\mathbf{r}_i \in GF(q)^k$ are random
2. Let $A_i \in GF(q)^{2\times 2}$ be random 2×2 matrices. Let $A = \mathrm{diag}[I_{n-w}, A_0, \cdots, A_{w-1}]$ be an $(n + w) \times (n + w)$ non-singular matrix.
3. The public key: $k \times (n + w)$ matrix $G = SG_1AP_2$ and the private key: (S, G_s, P_1, P_2, A) where S is random $k \times k$ matrix and P_2 is a permutation.

RLCE Encryption/Decryption

`RLCE.Enc`$(G, \mathbf{m}, \mathbf{e})$. For a message $\mathbf{m} \in GF(q)^k$, choose $\mathbf{e} \in GF(q)^{n+w}$ of weight at most t. The cipher: $\mathbf{c} = \mathbf{m}G + \mathbf{e}$.

`RLCE.Dec`$(S, G_s, P_1, P_2, A, \mathbf{c})$. For a cipher text $\mathbf{c}$, compute

$$\mathbf{c}P_2^{-1}A^{-1} = \mathbf{m}SG_1 + \mathbf{e}P_2^{-1}A^{-1} = [c'_0, \ldots, c'_{n+w-1}].$$

Let $\mathbf{c}' = [c'_0, c'_1, \cdots, c'_{n-w}, c'_{n-w+2}, \cdots, c'_{n+w-2}] \in GF(q)^n$. Then $\mathbf{c}'P_1^{-1} = \mathbf{m}SG_s + \mathbf{e}'$ for some $\mathbf{e}' \in GF(q)^n$ of weight at most t. Using an efficient decoding algorithm, one can recover $\mathbf{m}SG_s$ from $\mathbf{c}'P_1^{-1}$. Let D be a $k \times k$ inverse matrix of SG'_s where G'_s is the first k columns of G_s. Then $\mathbf{m} = \mathbf{c}_1 D$ where $\mathbf{c}_1$ is the first k elements of $\mathbf{m}SG_s$.

Why RLCE?

- The problem of decoding random linear codes is **NP**-hard
- Though challenging to show that decoding RLCE is **NP**-hard, the mixed random columns could hide all structures of underlying linear code
- Goppa-McEliece assumes Goppa codes behave like random codes while RLCE does not requires such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes has wide industry experience
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece

UNC CHARLOTTE
College of Computing and Informatics

Why RLCE?

- The problem of decoding random linear codes is **NP**-hard
- Though challenging to show that decoding RLCE is **NP**-hard, the mixed random columns could hide all structures of underlying linear code
- Goppa-McEliece assumes Goppa codes behave like random codes while RLCE does not requires such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes has wide industry experience
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece

UNC CHARLOTTE
College of Computing and Informatics

Why RLCE?

- The problem of decoding random linear codes is **NP**-hard
- Though challenging to show that decoding RLCE is **NP**-hard, the mixed random columns could hide all structures of underlying linear code
- Goppa-McEliece assumes Goppa codes behave like random codes while RLCE does not requires such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes has wide industry experience
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece

UNC CHARLOTTE
College of Computing and Informatics

Why RLCE?

- The problem of decoding random linear codes is **NP**-hard
- Though challenging to show that decoding RLCE is **NP**-hard, the mixed random columns could hide all structures of underlying linear code
- Goppa-McEliece assumes Goppa codes behave like random codes while RLCE does not requires such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes has wide industry experience
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece

Why RLCE?

- The problem of decoding random linear codes is **NP**-hard
- Though challenging to show that decoding RLCE is **NP**-hard, the mixed random columns could hide all structures of underlying linear code
- Goppa-McEliece assumes Goppa codes behave like random codes while RLCE does not requires such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes has wide industry experience
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece

Why RLCE?

- The problem of decoding random linear codes is **NP**-hard
- Though challenging to show that decoding RLCE is **NP**-hard, the mixed random columns could hide all structures of underlying linear code
- Goppa-McEliece assumes Goppa codes behave like random codes while RLCE does not requires such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes has wide industry experience
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece

Systematic RLCE

- Decryption for systematic RLCE could be more efficient.
- In the RLCE, one recovers $\mathbf{m}SG_s$ first.
- Let $\mathbf{m}SG_sP_1 = (d_0, \cdots, d_{n-1})$ and $\mathbf{c}_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ be a length $n + w$ vector.
- For each $i < k$ such that $d'_i = d_j$ for some $j < n - w$, we have $m_i = d_j$. Let

$$I_R = \{i : m_i \text{ is recovered via } \mathbf{m}SG_s\} \text{ and } \bar{I}_R = \{0, \cdots, k-1\} \setminus I_R.$$

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining u message symbols m_i with $i \in \bar{I}_R$.

UNC CHARLOTTE
College of Computing and Informatics

Systematic RLCE

- Decryption for systematic RLCE could be more efficient.
- In the RLCE, one recovers $\mathbf{m}SG_s$ first.
- Let $\mathbf{m}SG_sP_1 = (d_0, \cdots, d_{n-1})$ and $\mathbf{c}_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ be a length $n+w$ vector.
- For each $i < k$ such that $d'_i = d_j$ for some $j < n-w$, we have $m_i = d_j$. Let

$$I_R = \{i : m_i \text{ is recovered via } \mathbf{m}SG_s\} \text{ and } \bar{I}_R = \{0, \cdots, k-1\} \setminus I_R.$$

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining u message symbols m_i with $i \in \bar{I}_R$.

Systematic RLCE

- Decryption for systematic RLCE could be more efficient.
- In the RLCE, one recovers $\mathbf{m}SG_s$ first.
- Let $\mathbf{m}SG_sP_1 = (d_0, \cdots, d_{n-1})$ and $\mathbf{c}_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ be a length $n + w$ vector.
- For each $i < k$ such that $d'_i = d_j$ for some $j < n - w$, we have $m_i = d_j$. Let

$$I_R = \{i : m_i \text{ is recovered via } \mathbf{m}SG_s\} \text{ and } \bar{I}_R = \{0, \cdots, k-1\} \setminus I_R.$$

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining u message symbols m_i with $i \in \bar{I}_R$.

UNC CHARLOTTE
College of Computing and Informatics

Systematic RLCE

- Decryption for systematic RLCE could be more efficient.
- In the RLCE, one recovers $\mathbf{m}SG_s$ first.
- Let $\mathbf{m}SG_sP_1 = (d_0, \cdots, d_{n-1})$ and $\mathbf{c}_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ be a length $n+w$ vector.
- For each $i < k$ such that $d'_i = d_j$ for some $j < n-w$, we have $m_i = d_j$. Let

$$I_R = \{i : m_i \text{ is recovered via } \mathbf{m}SG_s\} \text{ and } \bar{I}_R = \{0, \cdots, k-1\} \setminus I_R.$$

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining u message symbols m_i with $i \in \bar{I}_R$.

Decoding algorith 1

The message symbols with indices in $\bar{I}_R$ could be recovered by solving the linear equation system

$$\mathbf{m}\left[\mathbf{g}_{i_0},\cdots,\mathbf{g}_{i_{u-1}}\right]=[d'_{i_0},\cdots,d'_{i_{u-1}}]$$

where $\mathbf{g}_{i_0},\cdots,\mathbf{g}_{i_{u-1}}$ are the corresponding columns in the public key. Choose P such that $\mathbf{m}P=(\mathbf{m}_{I_R},\mathbf{m}_{\bar{I}_R})$. Then

$$(\mathbf{m}_{I_R},\mathbf{m}_{\bar{I}_R})P^{-1}\left[\mathbf{g}_{i_0},\cdots,\mathbf{g}_{i_{u-1}}\right]=[d'_{i_0},\cdots,d'_{i_{u-1}}]$$

Let $P^{-1}\left[\mathbf{g}_{i_0},\cdots,\mathbf{g}_{i_{u-1}}\right]=\begin{pmatrix}V\\W\end{pmatrix}$. Then

$$\mathbf{m}_{\bar{I}_R}W=[d'_{i_0},\cdots,d'_{i_{u-1}}]-\mathbf{m}_{I_R}V.$$

$$\mathbf{m}_{\bar{I}_R}=\left([d'_{i_0},\cdots,d'_{i_{u-1}}]-\mathbf{m}_{I_R}V\right)W^{-1}.$$

UNC CHARLOTTE
College of Computing and Informatics

Defeating side-channel attacks

For the decoding algorithms 1, the value u is dependent on the choice of the private permutation P_2. Though the leakage of the size of u is not sufficient for the adversary to recover P_2 or to carry out other attacks against RLCE scheme, this kind of side-channel information leakage could be easily defeated by requiring u be smaller than u_0 in the following Table for selected P_2.

RLCE ID	0	1	2	3	4	5	6
u_0	200	123	303	190	482	309	7

UNC CHARLOTTE
College of Computing and Informatics

Two groups of parameters

- Group 1: $w < n - w$: This group is insecure due to the recent analysis by Alain Couvreur, Matthieu Lequesne, and Jean-Pierre Till
- Group 2: $w = n - k$: This one should be used

Two groups of parameters

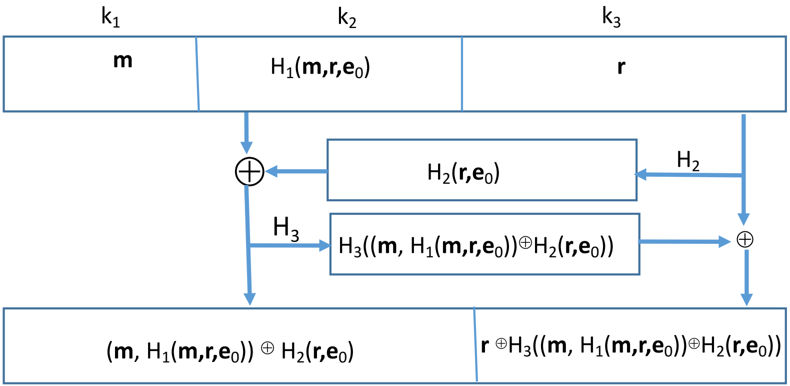
- Group 1: $w < n - w$: This group is insecure due to the recent analysis by Alain Couvreur, Matthieu Lequesne, and Jean-Pierre Till
- Group 2: $w = n - k$: This one should be used

UNC CHARLOTTE
College of Computing and Informatics

Recommended parameters

ID	κ_c, κ_q	LD	n	k	t	w	m	sk	cipher	pk
0	**128, 80**	$\perp$	630	470	80	160	**10**	310116	988	188001
2	**192,110**	$\perp$	1000	764	118	236	**10**	747393	1545	450761
4	**256,144**	$\perp$	1360	800	280	560	**11**	1773271	2640	1232001
6	22,22	$\perp$	40	20	10	5	**10**	1059	57	626
7	**128, 80**	(13,6663,14)	612	466	76	146	**10**	284636	948	170091
9	**192, 110**	(11,9317,12)	1000	790	108	210	**10**	703371	1513	414751
11	**256,144**	(26,23350,34)	1200	700	280	500	**11**	1382314	2338	926501
13	24,24	(3, 68,4)	40	20	11	5	**10**	1059	57	626
14	25,25	(10, 262,14)	40	20	12	5	**10**	1059	57	626

RLCE Padding: RLCEpad



Questions

Questions?

UNC CHARLOTTE
College of Computing and Informatics

Information-set decoding (ISD)

- Information-set decoding (ISD) is one of the most important message recovery attacks on McEliece encryption schemes.
- For the RLCE encryption scheme, the ISD attack is based on the number of columns in the public key G instead of the number of columns in the private key G_s.
- The cost of ISD attack on an $[n, k, t; w]$-RLCE scheme is equivalent to the cost of ISD attack on an $[n + w, k; t]$-McEliece scheme.

UNC CHARLOTTE
College of Computing and Informatics

Information-set decoding (ISD)

- Information-set decoding (ISD) is one of the most important message recovery attacks on McEliece encryption schemes.
- For the RLCE encryption scheme, the ISD attack is based on the number of columns in the public key G instead of the number of columns in the private key G_s.
- The cost of ISD attack on an $[n, k, t; w]$-RLCE scheme is equivalent to the cost of ISD attack on an $[n + w, k; t]$-McEliece scheme.

UNC CHARLOTTE
College of Computing and Informatics

Information-set decoding (ISD)

- Information-set decoding (ISD) is one of the most important message recovery attacks on McEliece encryption schemes.
- For the RLCE encryption scheme, the ISD attack is based on the number of columns in the public key G instead of the number of columns in the private key G_s.
- The cost of ISD attack on an $[n, k, t; w]$-RLCE scheme is equivalent to the cost of ISD attack on an $[n+w, k; t]$-McEliece scheme.

UNC CHARLOTTE
College of Computing and Informatics

Naive ISD

- Uniformly selects k columns from the public key and checks whether it is invertible.
- If it is invertible, one multiplies the inverse with the corresponding ciphertext values in these coordinates that correspond to the k columns of the public key.
- If these coordinates contain no errors in the ciphertext, one recovers the plain text.

UNC CHARLOTTE
College of Computing and Informatics

Naive ISD

- Uniformly selects k columns from the public key and checks whether it is invertible.
- If it is invertible, one multiplies the inverse with the corresponding ciphertext values in these coordinates that correspond to the k columns of the public key.
- If these coordinates contain no errors in the ciphertext, one recovers the plain text.

Naive ISD

- Uniformly selects k columns from the public key and checks whether it is invertible.
- If it is invertible, one multiplies the inverse with the corresponding ciphertext values in these coordinates that correspond to the k columns of the public key.
- If these coordinates contain no errors in the ciphertext, one recovers the plain text.

Quantum ISD

- For a function $f : \{0,1\}^l \to \{0,1\}$ with the property that there is an $x_0 \in \{0,1\}^l$ such that $f(x_0) = 1$ and $f(x) = 0$ for all $x \neq x_0$, Grover's algorithm finds the value x_0 using $\frac{\pi}{4}\sqrt{2^l}$ Grover iterations and $O(l)$ qubits.
- Grover's algorithm converts the function f to a reversible circuit C_f and calculates

$$|x\rangle \xrightarrow{C_f} (-1)^{f(x)}|x\rangle$$

in each of the Grover iterations. Thus the total steps for Grover's algorithm is bounded by $\frac{\pi|C_f|}{4}\sqrt{2^l}$.

UNC CHARLOTTE
College of Computing and Informatics

Quantum ISD

- For a function $f : \{0,1\}^l \to \{0,1\}$ with the property that there is an $x_0 \in \{0,1\}^l$ such that $f(x_0) = 1$ and $f(x) = 0$ for all $x \neq x_0$, Grover's algorithm finds the value x_0 using $\frac{\pi}{4}\sqrt{2^l}$ Grover iterations and $O(l)$ qubits.
- Grover's algorithm converts the function f to a reversible circuit C_f and calculates

$$|x\rangle \xrightarrow{C_f} (-1)^{f(x)}|x\rangle$$

in each of the Grover iterations. Thus the total steps for Grover's algorithm is bounded by $\frac{\pi|C_f|}{4}\sqrt{2^l}$.

UNC CHARLOTTE
College of Computing and Informatics

Quantum ISD against RLCE

Thus Grover's quantum algorithm requires approximately

$$7\left((n+w)k + k^{2.807} + k^2\right)(\log_2 q)^{1.585}\sqrt{\frac{\binom{n+w}{k}}{\binom{n+w-t}{k}}}$$

steps for the simple ISD algorithm against RLCE encryption scheme.

ISD for systematic RLCE schemes

- One uniformly selects $k = k_1 + k_2$ columns from the public key where k_1 columns are from the first k columns of the public key.
- Assume that first k_1 columns have no error. Simplify the computation process for ISD

UNC CHARLOTTE
College of Computing and Informatics

ISD for systematic RLCE schemes

- One uniformly selects $k = k_1 + k_2$ columns from the public key where k_1 columns are from the first k columns of the public key.
- Assume that first k_1 columns have no error. Simplify the computation process for ISD

Insecure ciphertexts for systematic RLCE schemes

- For a systematic RLCE, if a small number of errors were added to the first k components of the ciphertext, one may be able to exhaustively search these errors.
- Let

$$\gamma_l = \max_{l \le i \le t} \left\{ \frac{\binom{k-l}{k-i}}{q^i \binom{k}{i}} \right\}$$

The RLCE produces an insecure ciphertext in case that the ciphertext contains at most l errors within the first k components of the ciphertext and $\gamma_l > 2^{-\kappa_c}$ where κ_c is the security parameter.

Insecure ciphertexts for systematic RLCE schemes

- For a systematic RLCE, if a small number of errors were added to the first k components of the ciphertext, one may be able to exhaustively search these errors.
- Let
$$\gamma_l = \max_{l \le i \le t} \left\{ \frac{\binom{k-l}{k-i}}{q^i \binom{k}{i}} \right\}$$
The RLCE produces an insecure ciphertext in case that the ciphertext contains at most l errors within the first k components of the ciphertext and $\gamma_l > 2^{-\kappa_c}$ where κ_c is the security parameter.

Sidelnikov-Shestakov's attack

- If $w \geq n - k$, not enough equations for Sidelnikov-Shestakov's attack
- If $w < n - k$, one need to guess some values to establish enough equations. The guess space is normally too big to be successful.

UNC CHARLOTTE
College of Computing and Informatics

Sidelnikov-Shestakov's attack

- If $w \geq n-k$, not enough equations for Sidelnikov-Shestakov's attack
- If $w < n-k$, one need to guess some values to establish enough equations. The guess space is normally too big to be successful.

Known non-randomized column attack

- What happens if the positions of non-randomized $n-w$ GRS columns are known to the adversary?
- Possibility one: guess the remaining w columns of the GRS generator matrix. Search space too big
- Use Sidelnikov-Shestakov attack to calculate a private key for the punctured $[n-w, k]$ GRS_k code consisting of the non-randomized GRS columns and then list-decode the punctured $[n-w, k]$ GRS_k code.

Known non-randomized column attack

- What happens if the positions of non-randomized $n-w$ GRS columns are known to the adversary?
- Possibility one: guess the remaining w columns of the GRS generator matrix. Search space too big
- Use Sidelnikov-Shestakov attack to calculate a private key for the punctured $[n-w, k]$ GRS_k code consisting of the non-randomized GRS columns and then list-decode the punctured $[n-w, k]$ GRS_k code.

UNC CHARLOTTE
College of Computing and Informatics

Known non-randomized column attack

- What happens if the positions of non-randomized $n-w$ GRS columns are known to the adversary?
- Possibility one: guess the remaining w columns of the GRS generator matrix. Search space too big
- Use Sidelnikov-Shestakov attack to calculate a private key for the punctured $[n-w, k]$ GRS_k code consisting of the non-randomized GRS columns and then list-decode the punctured $[n-w, k]$ GRS_k code.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks

- For two codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length n, the star product code $\mathcal{C}_1 * \mathcal{C}_2$ is the vector space spanned by $\mathbf{a} * \mathbf{b}$ for all pairs $(\mathbf{a}, \mathbf{b}) \in \mathcal{C}_1 \times \mathcal{C}_2$ where $\mathbf{a} * \mathbf{b} = [a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}]$.
- For the square code $\mathcal{C}^2 = \mathcal{C} * \mathcal{C}$ of $\mathcal{C}$, we have $\dim \mathcal{C}^2 \le \min\left\{n, \binom{\dim \mathcal{C}+1}{2}\right\}$.
- For an $[n, k]$ GRS code $\mathcal{C}$, let $\mathbf{a}, \mathbf{b} \in \mathrm{GRS}_k(\mathbf{x}, \mathbf{y})$ where $\mathbf{a} = (y_0 p_1(x_0), \cdots, y_{n-1} p_1(x_{n-1}))$ and $\mathbf{b} = (y_0 p_2(x_0), \cdots, y_{n-1} p_2(x_{n-1}))$. Then $\mathbf{a} * \mathbf{b} = (y_0^2 p_1(x_0) p_2(x_0), \cdots, y_{n-1}^2 p_1(x_{n-1}) p_2(x_{n-1}))$. Thus $\mathrm{GRS}_k(\mathbf{x}, \mathbf{y})^2 \subseteq \mathrm{GRS}_{2k-1}(\mathbf{x}, \mathbf{y} * \mathbf{y})$ where we assume $2k - 1 \le n$.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks

- For two codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length n, the star product code $\mathcal{C}_1 * \mathcal{C}_2$ is the vector space spanned by $\mathbf{a} * \mathbf{b}$ for all pairs $(\mathbf{a}, \mathbf{b}) \in \mathcal{C}_1 \times \mathcal{C}_2$ where $\mathbf{a} * \mathbf{b} = [a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}]$.
- For the square code $\mathcal{C}^2 = \mathcal{C} * \mathcal{C}$ of $\mathcal{C}$, we have $\dim \mathcal{C}^2 \le \min \left\{ n, \binom{\dim \mathcal{C}+1}{2} \right\}$.
- For an $[n, k]$ GRS code $\mathcal{C}$, let $\mathbf{a}, \mathbf{b} \in \mathrm{GRS}_k(\mathbf{x}, \mathbf{y})$ where $\mathbf{a} = (y_0 p_1(x_0), \cdots, y_{n-1} p_1(x_{n-1}))$ and $\mathbf{b} = (y_0 p_2(x_0), \cdots, y_{n-1} p_2(x_{n-1}))$. Then $\mathbf{a} * \mathbf{b} = (y_0^2 p_1(x_0) p_2(x_0), \cdots, y_{n-1}^2 p_1(x_{n-1}) p_2(x_{n-1}))$. Thus $\mathrm{GRS}_k(\mathbf{x}, \mathbf{y})^2 \subseteq \mathrm{GRS}_{2k-1}(\mathbf{x}, \mathbf{y} * \mathbf{y})$ where we assume $2k - 1 \le n$.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks

- For two codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length n, the star product code $\mathcal{C}_1 * \mathcal{C}_2$ is the vector space spanned by $\mathbf{a} * \mathbf{b}$ for all pairs $(\mathbf{a}, \mathbf{b}) \in \mathcal{C}_1 \times \mathcal{C}_2$ where $\mathbf{a} * \mathbf{b} = [a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}]$.
- For the square code $\mathcal{C}^2 = \mathcal{C} * \mathcal{C}$ of $\mathcal{C}$, we have $\dim \mathcal{C}^2 \le \min\left\{n, \binom{\dim \mathcal{C}+1}{2}\right\}$.
- For an $[n, k]$ GRS code $\mathcal{C}$, let $\mathbf{a}, \mathbf{b} \in \mathrm{GRS}_k(\mathbf{x}, \mathbf{y})$ where $\mathbf{a} = (y_0 p_1(x_0), \cdots, y_{n-1} p_1(x_{n-1}))$ and $\mathbf{b} = (y_0 p_2(x_0), \cdots, y_{n-1} p_2(x_{n-1}))$. Then $\mathbf{a} * \mathbf{b} = (y_0^2 p_1(x_0) p_2(x_0), \cdots, y_{n-1}^2 p_1(x_{n-1}) p_2(x_{n-1}))$. Thus $\mathrm{GRS}_k(\mathbf{x}, \mathbf{y})^2 \subseteq \mathrm{GRS}_{2k-1}(\mathbf{x}, \mathbf{y} * \mathbf{y})$ where we assume $2k - 1 \le n$.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks against GRS-RLCE

- G is public key for an (n, k, d, t, w) GRS-RLCE scheme.
- Let $\mathcal{C}$ be the code generated by the rows of G.
- Let $\mathcal{D}_1$ be the code with a generator matrix D_1 obtained from G by replacing the randomized $2w$ columns with all-zero columns and let $\mathcal{D}_2$ be the code with a generator matrix D_2 obtained from G by replacing the $n - w$ non-randomized columns with zero columns.
- Since $\mathcal{C} \subset \mathcal{D}_1 + \mathcal{D}_2$ and the pair $(\mathcal{D}_1, \mathcal{D}_2)$ is an orthogonal pair, we have $\mathcal{C}^2 \subset \mathcal{D}_1^2 + \mathcal{D}_2^2$. It follows that

$$2k - 1 \leq \dim \mathcal{C}^2 \leq \min\{2k - 1, n - w\} + 2w \qquad (1)$$

where we assume that $2w \leq k^2$.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks against GRS-RLCE

- G is public key for an (n, k, d, t, w) GRS-RLCE scheme.
- Let $\mathcal{C}$ be the code generated by the rows of G.
- Let $\mathcal{D}_1$ be the code with a generator matrix D_1 obtained from G by replacing the randomized $2w$ columns with all-zero columns and let $\mathcal{D}_2$ be the code with a generator matrix D_2 obtained from G by replacing the $n - w$ non-randomized columns with zero columns.
- Since $\mathcal{C} \subset \mathcal{D}_1 + \mathcal{D}_2$ and the pair $(\mathcal{D}_1, \mathcal{D}_2)$ is an orthogonal pair, we have $\mathcal{C}^2 \subset \mathcal{D}_1^2 + \mathcal{D}_2^2$. It follows that

$$2k - 1 \leq \dim \mathcal{C}^2 \leq \min\{2k - 1, n - w\} + 2w \qquad (1)$$

where we assume that $2w \leq k^2$.

Filtration attacks against GRS-RLCE

- G is public key for an (n, k, d, t, w) GRS-RLCE scheme.
- Let $\mathcal{C}$ be the code generated by the rows of G.
- Let $\mathcal{D}_1$ be the code with a generator matrix D_1 obtained from G by replacing the randomized $2w$ columns with all-zero columns and let $\mathcal{D}_2$ be the code with a generator matrix D_2 obtained from G by replacing the $n - w$ non-randomized columns with zero columns.
- Since $\mathcal{C} \subset \mathcal{D}_1 + \mathcal{D}_2$ and the pair $(\mathcal{D}_1, \mathcal{D}_2)$ is an orthogonal pair, we have $\mathcal{C}^2 \subset \mathcal{D}_1^2 + \mathcal{D}_2^2$. It follows that

$$2k - 1 \le \dim \mathcal{C}^2 \le \min\{2k - 1, n - w\} + 2w \quad (1)$$

where we assume that $2w \le k^2$.

Filtration attacks against GRS-RLCE

- G is public key for an (n, k, d, t, w) GRS-RLCE scheme.
- Let $\mathcal{C}$ be the code generated by the rows of G.
- Let $\mathcal{D}_1$ be the code with a generator matrix D_1 obtained from G by replacing the randomized $2w$ columns with all-zero columns and let $\mathcal{D}_2$ be the code with a generator matrix D_2 obtained from G by replacing the $n-w$ non-randomized columns with zero columns.
- Since $\mathcal{C} \subset \mathcal{D}_1 + \mathcal{D}_2$ and the pair $(\mathcal{D}_1, \mathcal{D}_2)$ is an orthogonal pair, we have $\mathcal{C}^2 \subset \mathcal{D}_1^2 + \mathcal{D}_2^2$. It follows that

$$2k-1 \le \dim \mathcal{C}^2 \le \min\{2k-1, n-w\} + 2w \qquad (1)$$

where we assume that $2w \le k^2$.

Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $\mathcal{D}_2$ behave like random columns in the filtration attacks
- We have $\dim \mathcal{C}^2 = \mathcal{D}_1^2 + \mathcal{D}_2^2 = n - w + \mathcal{D}_2^2 = n + w$.
- For any code $\mathcal{C}'$ of length n' that is obtained from $\mathcal{C}$ using code puncturing and code shortening, we have $\dim \mathcal{C}'^2 = n'$.
- Thus filtration techniques could not be used to recover any non-randomized columns in D_1.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $\mathcal{D}_2$ behave like random columns in the filtration attacks
- We have $\dim \mathcal{C}^2 = \mathcal{D}_1^2 + \mathcal{D}_2^2 = n - w + \mathcal{D}_2^2 = n + w$.
- For any code $\mathcal{C}'$ of length n' that is obtained from $\mathcal{C}$ using code puncturing and code shortening, we have $\dim \mathcal{C}'^2 = n'$.
- Thus filtration techniques could not be used to recover any non-randomized columns in D_1.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $\mathcal{D}_2$ behave like random columns in the filtration attacks
- We have $\dim \mathcal{C}^2 = \mathcal{D}_1^2 + \mathcal{D}_2^2 = n - w + \mathcal{D}_2^2 = n + w$.
- For any code $\mathcal{C}'$ of length n' that is obtained from $\mathcal{C}$ using code puncturing and code shortening, we have $\dim \mathcal{C}'^2 = n'$.
- Thus filtration techniques could not be used to recover any non-randomized columns in D_1.

UNC CHARLOTTE
College of Computing and Informatics

Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $\mathcal{D}_2$ behave like random columns in the filtration attacks
- We have $\dim \mathcal{C}^2 = \mathcal{D}_1^2 + \mathcal{D}_2^2 = n - w + \mathcal{D}_2^2 = n + w$.
- For any code $\mathcal{C}'$ of length n' that is obtained from $\mathcal{C}$ using code puncturing and code shortening, we have $\dim \mathcal{C}'^2 = n'$.
- Thus filtration techniques could not be used to recover any non-randomized columns in D_1.

Running times for RLCE with Decoding Algorithm 1 (in milliseconds)

ID	key	encryption		decryption	
		RLCEspad	RLCEpad	RLCEspad	RLCEpad
0	340.616	0.565	0.538	1.574	1.509
2	1253.926	1.255	1.166	3.034	2.937
4	3215.791	2.836	2.796	13.092	12.925

UNC CHARLOTTE
College of Computing and Informatics

RLCE CPU cycles

ID	key generation	encryption	decryption
0	1011071617	1805010	4646941
2	3829675407	3331234	8668186
4	9612380645	8184051	36705481

UNC CHARLOTTE
College of Computing and Informatics

RLCE peak memory usage (bytes)

ID	Mul. Table	key generation	encryption	decryption
0	N	2,536,704	798,288	1,335,280
0	Y	4,648,656	2,437,320	2,856,584
2	N	6,178,744	1,906,576	3,178,688
2	Y	8,287,312	2,865,400	3,825,112
4	N	11,561,352	4,829,968	7,010,368
4	Y	19,975,040	10,258,112	12,227,384

Questions

Questions?

UNC CHARLOTTE
College of Computing and Informatics