Outline

1. Code Based Cryptography and RLCE
   - McEliece Encryption Scheme
   - RLCE Key setup
   - RLCE Encryption/Decryption
   - Why RLCE?
   - Systematic RLCE

2. Recommended parameters and RLCE padding

3. Appendix: Security Analysis and performance
   - ISD
   - Other potential security attacks
   - Filtration attacks
   - Performance

Yongge Wang
Quantum Resistant Public Key Encryption Scheme RLCE
Outline

1. Code Based Cryptography and RLCE
   - McEliece Encryption Scheme
   - RLCE Key setup
   - RLCE Encryption/Decryption
   - Why RLCE?
   - Systematic RLCE

2. Recommended parameters and RLCE padding

3. Appendix: Security Analysis and performance
   - ISD
   - Other potential security attacks
   - Filtration attacks
   - Performance
Outline

1. Code Based Cryptography and RLCE
   - McEliece Encryption Scheme
   - RLCE Key setup
   - RLCE Encryption/Decryption
   - Why RLCE?
   - Systematic RLCE

2. Recommended parameters and RLCE padding

3. Appendix: Security Analysis and performance
   - ISD
   - Other potential security attacks
   - Filtration attacks
   - Performance
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: \(G = S G_s P\). Private key: \(G_s\)
Where \(S\) is random and \(P\) is permutation.

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

**Mc.Dec\((S, G_s, P, c)\).** For a received ciphertext \(c\), first compute \(c' = cP^{-1} = mSG\). Next use an error-correction algorithm to recover \(m' = mS\) and compute the message \(m\) as \(m = m' S^{-1}\).
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. Let $G_s$ be a $k \times n$ generator matrix for an $[n, k, d]$ linear code $C$ correcting at least $t$ errors and $w \leq n$. Let $G_sP_1 = [g_0, \cdots, g_{n-1}]$ for a random permutation $P_1$.

1. Let $G_1 = [g_0, \cdots, g_{n-w}, r_0, \cdots, g_{n-1}, r_{w-1}]$ be a $k \times (n+w)$ matrix where $r_i \in GF(q)^k$ are random.

2. Let $A_i \in GF(q)^{2 \times 2}$ be random $2 \times 2$ 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.
RLCE Key setup. Let $G_s$ be a $k \times n$ generator matrix for an $[n, k, d]$ linear code $C$ correcting at least $t$ errors and $w \leq n$. Let $G_s P_1 = [g_0, \cdots, g_{n-1}]$ for a random permutation $P_1$

1. Let $G_1 = [g_0, \cdots, g_{n-w}, r_0, \cdots, g_{n-1}, r_{w-1}]$ be a $k \times (n + w)$ matrix where $r_i \in GF(q)^k$ are random.

2. Let $A_i \in GF(q)^{2 \times 2}$ be random $2 \times 2$ 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.
RLCE Key setup

RLCE.KeySetup. Let $G_s$ be a $k \times n$ generator matrix for an $[n, k, d]$ linear code $C$ correcting at least $t$ errors and $w \leq n$. Let $G_s P_1 = [g_0, \cdots, g_{n-1}]$ for a random permutation $P_1$

1. Let $G_1 = [g_0, \cdots, g_{n-w}, r_0, \cdots, g_{n-1}, r_{w-1}]$ be a $k \times (n + w)$ matrix where $r_i \in GF(q)^k$ are random

2. Let $A_i \in GF(q)^{2 \times 2}$ be random $2 \times 2$ 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 $G = S G_1 A P_2$ 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

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

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

\[ cP_2^{-1}A^{-1} = mSG_1 + eP_2^{-1}A^{-1} = [c'_0, \ldots, c'_{n+w-1}]\].

Let }c' = [c'_0, c'_1, \ldots, c'_{n-w}, c'_{n-w+2}, \ldots, c'_{n+w-2}] \in GF(q)^n.\text{ Then }c'P_1^{-1} = mSG_s + e'\text{ for some }e' \in GF(q)^n\text{ of weight at most }t.\text{ Using an efficient decoding algorithm, one can recover }mSG_s\text{ from }c'P_1^{-1}.\text{ Let }D\text{ be a }k \times k\text{ inverse matrix of }SG_s\text{ where }G'_s\text{ is the first }k\text{ columns of }G_s.\text{ Then }m = c_1D\text{ where }c_1\text{ is the first }k\text{ elements of }mSG_s.
Why RLCE?

- The problem of decoding random linear codes is \( \textbf{NP} \)-hard
- Though challenging to show that decoding RLCE is \( \textbf{NP} \)-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 require such kind of assumption
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes have wide industry experience
- \textbf{Limitation}: RLCE public key sizes are larger though smaller than Goppa-McEliece
Why RLCE?

- The problem of decoding random linear codes is \( \text{NP} \)-hard.
- Though challenging to show that decoding RLCE is \( \text{NP} \)-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 require such kind of assumption.
- Other McEliece variants are based on stronger assumptions that certain structured codes are hard to decode.
- Reed-Solomon codes have 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 \textbf{NP}-hard
- Though challenging to show that decoding RLCE is \textbf{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 require 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
- \textbf{Limitation}: RLCE public key sizes are larger though smaller than Goppa-McEliece
Why RLCE?

- The problem of decoding random linear codes is \( \text{NP} \)-hard.
- Though challenging to show that decoding RLCE is \( \text{NP} \)-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 require such kind of assumption.
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.

Limitation: RLCE public key sizes are larger though smaller than Goppa-McEliece.
Why RLCE?

- The problem of decoding random linear codes is \( \text{NP} \)-hard.
- Though challenging to show that decoding RLCE is \( \text{NP} \)-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 require such kind of assumption.
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes have 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 $\textbf{NP}$-hard.
- Though challenging to show that decoding RLCE is $\textbf{NP}$-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 require such kind of assumption.
- Other McEliece variants are based on stronger assumption that certain structured codes are hard to decode.
- Reed-Solomon codes have wide industry experience.
- **Limitation**: RLCE public key sizes are larger though smaller than Goppa-McEliece.
Decryption for systematic RLCE could be more efficient.

In the RLCE, one recovers $m_{SG}$ first.

Let $m_{SG}P_1 = (d_0, \cdots, d_{n-1})$ and $c_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ 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

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

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining 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 $m_{SG}$ first.

Let $m_{SG}P_1 = (d_0, \cdots, d_{n-1})$ and $c_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ 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

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

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

Yongge Wang
Systematic RLCE

- Decryption for systematic RLCE could be more efficient.
- In the RLCE, one recovers $m_{SG_s}$ first.

Let $m_{SG_s} P_1 = (d_0, \cdots, d_{n-1})$ and $c_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp) P_2$ 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

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

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining message symbols $m_i$ with $i \in \bar{I}_R$. 
Decryption for systematic RLCE could be more efficient.

In the RLCE, one recovers $mSG_s$ first.

Let $mSG_sP_1 = (d_0, \cdots, d_{n-1})$ and $c_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ 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 $I_R = \{i : m_i \text{ is recovered via } mSG_s\}$ and $\bar{I}_R = \{0, \cdots, k-1\}\backslash I_R$.

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining $u$ message symbols $m_i$ with $i \in \bar{I}_R$. 
Decoding algorithm 1

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

$$m \begin{bmatrix} g_{i_0}, & \cdots, & g_{i_{u-1}} \end{bmatrix} = \begin{bmatrix} d'_0, & \cdots, & d'_{i_{u-1}} \end{bmatrix}$$

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

$$(m_{I_R}, m_{\bar{I}_R})P^{-1} \begin{bmatrix} g_{i_0}, & \cdots, & g_{i_{u-1}} \end{bmatrix} = \begin{bmatrix} d'_0, & \cdots, & d'_{i_{u-1}} \end{bmatrix}$$

Let $P^{-1} \begin{bmatrix} g_{i_0}, & \cdots, & g_{i_{u-1}} \end{bmatrix} = \begin{bmatrix} V & W \end{bmatrix}$. Then

$$m_{\bar{I}_R} W = \begin{bmatrix} d'_0, & \cdots, & d'_{i_{u-1}} \end{bmatrix} - m_{I_R} V.$$ 

$$m_{\bar{I}_R} = \left( \begin{bmatrix} d'_0, & \cdots, & d'_{i_{u-1}} \end{bmatrix} - m_{I_R} V \right) W^{-1}.$$
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 $P_2$.

<table>
<thead>
<tr>
<th>RLCE ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_0$</td>
<td>200</td>
<td>123</td>
<td>303</td>
<td>190</td>
<td>482</td>
<td>309</td>
<td>7</td>
</tr>
</tbody>
</table>
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.
## Recommended parameters

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

\[ \begin{align*}
  k_1 & \quad m & \quad H_1(m, r, e_0) & \quad k_2 & \quad H_2(r, e_0) & \quad k_3 & \quad r \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\
  & \downarrow & & & \downarrow & & \\  \end{align*} \]
Questions?

Yongge Wang

Quantum Resistant Public Key Encryption Scheme RLCE
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.
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.
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.
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.
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 \( f : \{0, 1\}^l \rightarrow \{0, 1\} \) 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 \( \frac{\pi |C_f|}{4} \sqrt{2^l} \).
Quantum ISD

- For a function $f : \{0, 1\}^l \rightarrow \{0, 1\}$ 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 $\frac{\pi |C_f|}{4} \sqrt{2^l}$.
Thus Grover’s quantum algorithm requires approximately

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

steps for the simple ISD algorithm against RLCE encryption scheme.
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.
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

$$
\gamma_I = \max_{1 \leq i \leq t} \left\{ \frac{\binom{k-I}{k-i}}{q^i \binom{k}{i}} \right\}
$$

The RLCE produces an insecure ciphertext in case that the ciphertext contains at most $I$ errors within the first $k$ components of the ciphertext and $\gamma_I > 2^{-\kappa_c}$ where $\kappa_c$ 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

$$\gamma_l = \max_{l \leq i \leq t} \left\{ \frac{(k-l)_i}{q^i \binom{k}{i}} \right\}$$

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 $\kappa_c$ 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.
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.
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.
For two codes $C_1$ and $C_2$ of length $n$, the star product code $C_1 \ast C_2$ is the vector space spanned by $a \ast b$ for all pairs $(a, b) \in C_1 \times C_2$ where $a \ast b = [a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}]$.

For the square code $C^2 = C \ast C$ of $C$, we have $\dim C^2 \leq \min \{n, (\dim C + 1)\}$.

For an $[n, k]$ GRS code $C$, let $a, b \in \text{GRS}_k(x, y)$ where
\[
a = (y_0 p_1(x_0), \cdots, y_{n-1} p_1(x_{n-1})) \quad \text{and} \quad b = (y_0 p_2(x_0), \cdots, y_{n-1} p_2(x_{n-1})).
\]
Then $a \ast b = (y_0^2 p_1(x_0)p_2(x_0), \cdots, y_{n-1}^2 p_1(x_{n-1})p_2(x_{n-1}))$. Thus $\text{GRS}_k(x, y)^2 \subseteq \text{GRS}_{2k-1}(x, y \ast y)$ where we assume $2k - 1 \leq n$. 
Filtration attacks

- For two codes $C_1$ and $C_2$ of length $n$, the star product code $C_1 * C_2$ is the vector space spanned by $a * b$ for all pairs $(a, b) \in C_1 \times C_2$ where $a * b = [a_0b_0, a_1b_1, \cdots, a_{n-1}b_{n-1}]$.

- For the square code $C^2 = C * C$ of $C$, we have $\dim C^2 \leq \min \left\{ n, \left( \frac{\dim C + 1}{2} \right) \right\}$.

- For an $[n, k]$ GRS code $C$, let $a, b \in \text{GRS}_k(x, y)$ where $a = (y_0p_1(x_0), \cdots, y_{n-1}p_1(x_{n-1}))$ and $b = (y_0p_2(x_0), \cdots, y_{n-1}p_2(x_{n-1}))$. Then $a * b = (y_0^2p_1(x_0)p_2(x_0), \cdots, y_{n-1}^2p_1(x_{n-1})p_2(x_{n-1}))$. Thus $\text{GRS}_k(x, y)^2 \subseteq \text{GRS}_{2k-1}(x, y * y)$ where we assume $2k - 1 \leq n$. 
Filtration attacks

- For two codes $C_1$ and $C_2$ of length $n$, the star product code $C_1 \ast C_2$ is the vector space spanned by $a \ast b$ for all pairs $(a, b) \in C_1 \times C_2$ where $a \ast b = [a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}]$.

- For the square code $C^2 = C \ast C$ of $C$, we have $\dim C^2 \leq \min \left\{ n, \binom{\dim C + 1}{2} \right\}$.

- For an $[n, k]$ GRS code $C$, let $a, b \in \text{GRS}_k(x, y)$ where $a = (y_0 p_1(x_0), \cdots, y_{n-1} p_1(x_{n-1}))$ and $b = (y_0 p_2(x_0), \cdots, y_{n-1} p_2(x_{n-1}))$. Then $a \ast b = (y_0^2 p_1(x_0) p_2(x_0), \cdots, y_{n-1}^2 p_1(x_{n-1}) p_2(x_{n-1}))$. Thus $\text{GRS}_k(x, y)^2 \subseteq \text{GRS}_{2k-1}(x, y \ast y)$ where we assume $2k - 1 \leq n$. 
Filtration attacks against GRS-RLCE

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

\[
2k - 1 \leq \dim C^2 \leq \min\{2k - 1, n - w\} + 2w
\]

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 $C$ be the code generated by the rows of $G$.
- Let $D_1$ be the code with a generator matrix $D_1$ obtained from $G$ by replacing the randomized $2w$ columns with all-zero columns and let $D_2$ be the code with a generator matrix $D_2$ obtained from $G$ by replacing the $n - w$ non-randomized columns with zero columns.
- Since $C \subset D_1 + D_2$ and the pair $(D_1, D_2)$ is an orthogonal pair, we have $C^2 \subset D_1^2 + D_2^2$. It follows that

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

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 $C$ be the code generated by the rows of $G$.
- Let $D_1$ be the code with a generator matrix $D_1$ obtained from $G$ by replacing the randomized $2w$ columns with all-zero columns and let $D_2$ be the code with a generator matrix $D_2$ obtained from $G$ by replacing the $n - w$ non-randomized columns with zero columns.
- Since $C \subset D_1 + D_2$ and the pair $(D_1, D_2)$ is an orthogonal pair, we have $C^2 \subset D_1^2 + D_2^2$. It follows that

\[
2k - 1 \leq \dim C^2 \leq \min\{2k - 1, n - w\} + 2w
\]

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

---

Yongge Wang
Quantum Resistant Public Key Encryption Scheme RLCE
Filtration attacks against GRS-RLCE

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

$$2k - 1 \leq \dim C^2 \leq \min\{2k - 1, n - w\} + 2w \tag{1}$$

where we assume that $2w \leq k^2$. 
Filtration attacks against GRS-RLCE: \( k \geq n - w \)

- Assume that the \( 2w \) randomized columns in \( D_2 \) behave like random columns in the filtration attacks
- We have \( \dim C^2 = D_1^2 + D_2^2 = n - w + D_2^2 = n + w \).
- For any code \( C' \) of length \( n' \) that is obtained from \( C \) using code puncturing and code shortening, we have \( \dim C'^2 = n' \).
- Thus filtration techniques could not be used to recover any non-randomized columns in \( D_1 \).
Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $D_2$ behave like random columns in the filtration attacks.
- We have $\dim C^2 = D_1^2 + D_2^2 = n - w + D_2^2 = n + w$.
- For any code $C'$ of length $n'$ that is obtained from $C$ using code puncturing and code shortening, we have $\dim C'^2 = n'$.
- Thus filtration techniques could not be used to recover any non-randomized columns in $D_1$. 
Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $D_2$ behave like random columns in the filtration attacks.
- We have $\dim C^2 = D_1^2 + D_2^2 = n - w + D_2^2 = n + w$.
- For any code $C'$ of length $n'$ that is obtained from $C$ using code puncturing and code shortening, we have $\dim C'^2 = n'$.
- Thus filtration techniques could not be used to recover any non-randomized columns in $D_1$. 
Filtration attacks against GRS-RLCE: $k \geq n - w$

- Assume that the $2w$ randomized columns in $D_2$ behave like random columns in the filtration attacks.
- We have $\dim C^2 = D_1^2 + D_2^2 = n - w + D_2^2 = n + w$.
- For any code $C'$ of length $n'$ that is obtained from $C$ using code puncturing and code shortening, we have $\dim C'^2 = n'$.
- 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)

<table>
<thead>
<tr>
<th>ID</th>
<th>key</th>
<th>encryption</th>
<th>decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RLCEspad</td>
<td>RLCEpad</td>
</tr>
<tr>
<td>0</td>
<td>340.616</td>
<td>0.565</td>
<td>0.538</td>
</tr>
<tr>
<td>2</td>
<td>1253.926</td>
<td>1.255</td>
<td>1.166</td>
</tr>
<tr>
<td>4</td>
<td>3215.791</td>
<td>2.836</td>
<td>2.796</td>
</tr>
</tbody>
</table>
## RLCE CPU cycles

<table>
<thead>
<tr>
<th>ID</th>
<th>key generation</th>
<th>encryption</th>
<th>decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1011071617</td>
<td>1805010</td>
<td>4646941</td>
</tr>
<tr>
<td>2</td>
<td>3829675407</td>
<td>3331234</td>
<td>8668186</td>
</tr>
<tr>
<td>4</td>
<td>9612380645</td>
<td>8184051</td>
<td>36705481</td>
</tr>
</tbody>
</table>
### RLCE peak memory usage (bytes)

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

Yongge Wang
Quantum Resistant Public Key Encryption Scheme RLCE