QC-MDPC KEM

Philip Lafrance

ISARA Corporation
<philip.lafrance@isara.com>

April 13, 2018
What is it?

Encryption-based Key Encapsulation Mechanism:
Takes as input a public key and a secret seed.
Derives and “encapsulates” an ephemeral symmetric key $K$.
$K$ can be recovered from the ciphertext by using the secret key matching the public key used above.
Background

What is it?

- Encryption-based Key Encapsulation Mechanism:
What is it?

- Encryption-based Key Encapsulation Mechanism:
  - Takes as input a public key and a secret seed.
Background

What is it?

- Encryption-based Key Encapsulation Mechanism:
  - Takes as input a public key and a secret seed.
  - Derives and “encapsulates” an *ephemeral* symmetric key $K$. 

K can be recovered from the ciphertext by using the secret key matching the public key used above.
What is it?

- Encryption-based Key Encapsulation Mechanism:
  - Takes as input a public key and a secret seed.
  - Derives and “encapsulates” an *ephemeral* symmetric key $K$.
  - $K$ can be recovered from the ciphertext by using the secret key matching the public key used above.
What is it based on?
What is it based on?

- McEliece Encryption Scheme:
What is it based on?

- McEliece Encryption Scheme:
  - Encrypted messages are of the form: \( mG \oplus e \),
What is it based on?

- **McEliece Encryption Scheme:**
  - Encrypted messages are of the form: $mG \oplus e$,
  - where, $m$ is the message, $e$ is an error vector, and $G$ is the public-key.
What is it based on?

- **McEliece Encryption Scheme:**
  - Encrypted messages are of the form: \( mG \oplus e \),
  - where, \( m \) is the message, \( e \) is an error vector, and \( G \) is the public-key.

- **Using Quasi-Cyclic Moderate Density Parity Check codes.**
McEliece Encryption Scheme:

- Encrypted messages are of the form: \( mG \oplus e \),
- where, \( m \) is the message, \( e \) is an error vector, and \( G \) is the public-key.

Using Quasi-Cyclic Moderate Density Parity Check codes.

- \( n \) - codeword length
What is it based on?

- McEliece Encryption Scheme:
  - Encrypted messages are of the form: \( mG \oplus e \),
  - where, \( m \) is the message, \( e \) is an error vector, and \( G \) is the public-key.

- Using Quasi-Cyclic Moderate Density Parity Check codes.
  - \( n \) - codeword length
  - \( 2^k \) - cardinality of the code family
Background

What is it based on?

- McEliece Encryption Scheme:
  - Encrypted messages are of the form: \( mG \oplus e \),
  - where, \( m \) is the message, \( e \) is an error vector, and \( G \) is the public-key.

- Using Quasi-Cyclic Moderate Density Parity Check codes.
  - \( n \) - codeword length
  - \( 2^k \) - cardinality of the code family
  - \( k \), and \( r = k = n/2 \) - dimension and co-dimension
Background

What is it based on?

- McEliece Encryption Scheme:
  - Encrypted messages are of the form: \( mG \oplus e \),
  - where, \( m \) is the message, \( e \) is an error vector, and \( G \) is the public-key.

- Using Quasi-Cyclic Moderate Density Parity Check codes.
  - \( n \) - codeword length
  - \( 2^k \) - cardinality of the code family
  - \( k \), and \( r = k = n/2 \) - dimension and co-dimension
  - \( w \in \mathcal{O}(\sqrt{n \log(n)}) \) - weight of the rows of the parity-check matrix \( H \)
Background

What is it based on?

- McEliece Encryption Scheme:
  - Encrypted messages are of the form: \( mG \oplus e \),
  - where, \( m \) is the message, \( e \) is an error vector, and \( G \) is the public-key.

- Using Quasi-Cyclic Moderate Density Parity Check codes.
  - \( n \) - codeword length
  - \( 2^k \) - cardinality of the code family
  - \( k \), and \( r = k = n/2 \) - dimension and co-dimension
  - \( w \in \mathcal{O}(\sqrt{n \log(n)}) \) - weight of the rows of the parity-check matrix \( H \)
  - \( t \) - the error-correction threshold
Algorithm 1 QCMDPC.KeyGen

**Input:** Security parameter $n = 2r$, weight $w$, and co-dimension $r$.

**Output:** Public key $G$, secret key $H$.

1. Select $h_0, h_1 \in \{0, 1\}^r$, each of odd weight $w/2$.
2. Compute $H_0, H_1 \in \mathbb{F}_2^{r \times r}$ by right circular shifts of $h_0$ and $h_1$.
3. Set $H = [H_0|H_1] \in \mathbb{F}_2^{r \times n}$.
4. Calculate $Q = (H_1^{-1}H_0)^T$.
5. Set $G = [I_k|Q]$.
6. return $(G, H)$. 
Recall that an encrypted message is of the form:
\[ c = mG \oplus e. \]
Recall that an encrypted message is of the form:

$$c = mG \oplus e.$$ 

To recover $m$, a decoding algorithm is required.
Decoding Algorithms

Recall that an encrypted message is of the form:

\[ c = mG \oplus e. \]

- To recover \( m \), a *decoding algorithm* is required.
- The choice of decoder does not affect interoperability/functionality.
Recall that an encrypted message is of the form:

\[ c = mG \oplus e. \]

- To recover \( m \), a *decoding algorithm* is required.
- The choice of decoder does not affect interoperability/functionality.
- However, for security reasons, the decoding algorithm must be constant time, and preferably with as low of a decoding failure rate (DFR) as possible.
We require an error vector derivation function, as well as two key derivation functions.
We require an error vector derivation function, as well as two key derivation functions.

- \( \nu : \{0, 1\}^* \rightarrow \{0, 1\}^n \) – an efficient, deterministic, pseudorandom, one-way function with weight \( t \) outputs.
We require an error vector derivation function, as well as two key derivation functions.

- \( \nu : \{0, 1\}^* \rightarrow \{0, 1\}^n \) – an efficient, deterministic, pseudorandom, one-way function with weight \( t \) outputs.
- \( \text{KDF}_1 : \{0, 1\}^* \rightarrow \{0, 1\}^k \), and
We require an error vector derivation function, as well as two key derivation functions.

- $\nu : \{0,1\}^* \rightarrow \{0,1\}^n$ – an efficient, deterministic, pseudorandom, one-way function with weight $t$ outputs.
- $\text{KDF}_1 : \{0,1\}^* \rightarrow \{0,1\}^k$, and
- $\text{KDF}_2 : \{0,1\}^* \rightarrow \{0,1\}^{256+\ell}$ – where $\ell$ is the desired key length.
Algorithm 2 QCMDPC.Encap

**Input:** Public key $G$, and random seed $s \in \mathbb{F}_2^k$.

**Output:** Symmetric key $K \in \{0, 1\}^m$.

**Output:** Ciphertext $C = (C_1, C_2) \in \mathbb{F}_2^{256} \times \mathbb{F}_2^\ell$.

1: $e \leftarrow \nu(s)$ \hspace{1cm} \text{▷ Compute $n$-bit error vector}
2: $y \leftarrow KDF_1(e)$ \hspace{1cm} \text{▷ Compute $k$-bit masking value}
3: $x \leftarrow s \oplus y$ \hspace{1cm} \text{▷ Obtain $k$-bit plain text}
4: $C_1 \leftarrow xG \oplus e$ \hspace{1cm} \text{▷ Encrypt $x$ with $e$}
5: $C_2 \leftarrow K \| KDF_2(s)$
6: return $(K, C = (C_1, C_2))$
Decapsulation

Algorithm 3 QC-MDPC.Decap

Input: Secret key $H$, ciphertext $(C_1, C_2) \in \mathbb{F}_2^{256} \times \mathbb{F}_2^{\ell}$, and dimension $k$.
Output: Symmetric key $K \in \{0, 1\}^\ell$ or a decapsulation failure $\bot$.

1: $((x, e), d_{err}) \leftarrow$ QCMDPC.Decrypt($H, C_1$).
2: $y \leftarrow KDF_1(e)$
3: $s \leftarrow x \oplus y$
4: $e' \leftarrow \nu(s)$.
5: $C_2' || K \leftarrow KDF_2(s)$.
6: if $e' = e$ and $C_2' = C_2$ and $d_{err} = \text{False}$ then
7: return $K$
8: else
9: return $\bot$
10: end if
What attacks were considered?
What attacks were considered?

- State-of-the-art key distinguishing, key recovery, and decoding attacks.
What attacks were considered?

- State-of-the-art key distinguishing, key recovery, and decoding attacks.
- ISD,
What attacks were considered?

- State-of-the-art key distinguishing, key recovery, and decoding attacks.
  - ISD,
  - Prange + Grover, MMT + Quantum Walks (QISD),
What attacks were considered?

- State-of-the-art key distinguishing, key recovery, and decoding attacks.
  - ISD,
  - Prange + Grover, MMT + Quantum Walks (QISD),
- GJS
What attacks were considered?

- State-of-the-art key distinguishing, key recovery, and decoding attacks.
  - ISD,
  - Prange + Grover, MMT + Quantum Walks (QISD),
- GJS
- IND-CPA reduction
<table>
<thead>
<tr>
<th>Security</th>
<th>Classical</th>
<th>Quantum</th>
<th>n</th>
<th>r</th>
<th>w</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>80</td>
<td>58</td>
<td>9602</td>
<td>4801</td>
<td>90</td>
<td>84</td>
</tr>
<tr>
<td>Quantum</td>
<td>128</td>
<td>86</td>
<td>19714</td>
<td>9857</td>
<td>142</td>
<td>134</td>
</tr>
<tr>
<td>Classical</td>
<td>256</td>
<td>154</td>
<td>65542</td>
<td>32771</td>
<td>274</td>
<td>264</td>
</tr>
</tbody>
</table>

**Table:** Parameter sets for classical and quantum security\(^1\).

Using the \((65542, 32771, 274, 264)\) parameter set:

<table>
<thead>
<tr>
<th>Security</th>
<th>Classical</th>
<th>Quantum</th>
<th>Public key</th>
<th>Private Key</th>
<th>Ciphertext</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>256</td>
<td>154</td>
<td>4097</td>
<td>548</td>
<td>8226</td>
</tr>
</tbody>
</table>

**Table**: Data sizes in bytes.
Thank You.