A new framework for code-based encryption and more

J.-C. Deneuville
<jean-christophe.deneuville@insa-cvl.fr>

April the 5th, 2018

Code-based Cryptography Workshop
Fort Lauderdale

Joint work with:
C. Aguilar Melchor  O. Blazy  P. Gaborit  G. Zémor
IRIT Toulouse  XLIM-DMI Limoges  University of Bordeaux
Motivations

[ME78]
Motivations

[ME78] → [Nie86]
Motivations

[ME78] → [Nie86] → RS, BCH, Goppa, RM → \[\begin{array}{c}
80's \\
\downarrow \\
00's
\end{array}\]
Motivations

Key Sizes

\[
\begin{array}{c}
\text{[ME78]} \rightarrow \text{[Nie86]} \\
\quad \text{RS} \\
\quad \text{BCH} \\
\quad \text{Goppa} \\
\quad \text{RM} \\
\end{array}
\]

\[
\begin{array}{c}
\text{80's} \\
\downarrow \\
\text{00's}
\end{array}
\]

Security reduction to a standard problem (random codes)
Motivations

Key Sizes

<table>
<thead>
<tr>
<th>[ME78] → [Nie86]</th>
<th>RS</th>
<th>BCH</th>
<th>Goppa</th>
<th>RM</th>
<th>80's</th>
<th>Other variations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>↓</td>
<td>00's</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Most of them broken</td>
</tr>
</tbody>
</table>

Security proof
Motivations

Key Sizes

- Rank Metric
- [Gab91]
- [ME78] → [Nie86] → RS, BCH, Goppa, RM
- 80's → 00's
- Other variations
- Most of them broken

Security proof
Motivations

Key Sizes

Rank Metric

[ME78] -> [Nie86] -> [Gab91]

RS BCH Goppa RM

80's 00's Other variations Most of them broken

Security proof

[Ale03]
Motivations

Key Sizes
- 80's
- 00's

Other variations
Most of them broken

Security proof

Group action

Rank Metric

Groups
- BCH
- Goppa
- RM

Metrics
- RS

Groups
- RDLC
- QC-LDPC
- Ntru-like

Rank Metric

Rank Metric

Attacks
Lack a Proof
Lack Efficiency

1980's
1990's

References
- [ME78]
- [Nie86]
- [Gab05]
- [Gab91]
- [Ale03]
- [Ove07]
- [MB09]
- [BCGO09]
- [BBC08]
- [MTSB13]
- [GMRZ13]
- [ABDGZ16]
Motivations

Key Sizes

- 80's \[\text{Nie86}\]
- 00's \[\text{Gab05}\]

Other variations

Most of them broken

Rank Metric

- RS
- BCH
- Goppa
- RM

Group action

80's \[\text{Gab91}\] \[\text{Ove07}\]

Attacks

Lack a Proof

Lack Efficiency

Security proof

80's \[\text{Ale03}\]
Motivations

Key Sizes

- 80's
- 00's

Group action

Rank Metric

- [ME78]
- [Nie86]
- [Gab91]
- [BBC08]
- QC-LDPC
- [Ove07]
- Attacks

Metric

- RS
- BCH
- Goppa
- RM

Action

80's

00's

Other variations
Most of them broken

Security proof

- [Ale03]
- [Gab05]
- [ABDGZ16]
- [Ove07]
Motivations

- **Group action** → [Gab05]

- **Rank Metric** → [Gab91]

- **Key Sizes**
  - 80's
  - 00's

- **Attacks**
  - [Ove07]

- **Security proof** → [Ale03]

- **Other variations**
  - Most of them broken

- **Group**
  - RS
  - BCH
  - Goppa
  - RM

- **Rank Metric**
  - 80's
  - 00's

- **Attacks**

- **Group action**
  - [MB09] dyadic
  - [BCGO09] alternant
  - [BBC08] QC-LDPC

- **Group**
  - Ntru-like
  - QC-MDPC
  - QC-LRPC

- **Rank Metric**
  - [MTSB13]

- **Attacks**

- **Security proof**
Motivations

- Key Sizes
  - 80's
  - 00's
  - [ME78] [Nie86]

- Group action
  - [Gab05]
  - [BBC08] QC-LDPC
  - [GMRZ13] QC-LRPC
  - [MB09] dyadic
  - [BCGO09] alternant

- Rank Metric
  - [Gab91]
  - [Ove07] Attacks
  - Ntru-like

- Metric
  - Action
    - [MB09] dyadic
    - [BCGO09] alternant

- Security proof
  - [Ale03]

- Other variations
  - Most of them broken

- Attacks
  - Bottom Line
    - Lack a Proof
    - Lack Efficiency

- Group
  - Action
    - Metric
      - Rank
    - [MC08] alternant

- Metric
  - Action
    - Group
  - Security proof

- 80's
  - 00's

- RS
  - BCH
  - Goppa
  - RM

- Ntru-like
  - [MTSB13]
  - QC-MDPC
  - QC-LRPC
Motivations

- **Group action**
  - [ME78]
  - [Nie86]
  - [Gab05]
  - [Gab91]
  - [BBC08]
  - [GMRZ13]

- **Rank Metric**
  - [ME78] → [Nie86] → RS
  - BCH
  - Goppa
  - RM

- **Key Sizes**
  - 80's
  - 00's
  - Other variations
    - Most of them broken

- **Security proof**
  - [Ale03]

- **Attacks**
  - [Ove07]
  - Ntru-like
  - QC-LRPC

- **Security**
  - Lack a Proof
  - Lack Efficiency

- **Rank Metric**
  - [MB09] dyadic
  - [BCGO09] alternant
  - QC-LDPC
  - QC-MDPC

- **Group action**
  - Ntru-like
  - [MTSB13]
Motivations

Key Sizes

80's ↓ 00's

Other variations
Most of them broken

Group action

[Nie86] → [ME78]

Rank Metric

[Gab91] → [Gab05]

[BCGO09] alternant

[MB09] dyadic

QC-LDPC

QC-MDPC

Attacks

80's ↓ 00's

Security proof

[Nie86] → [ME78]

[BCH] Goppa RM

[BCGO09] alternant

[MB09] dyadic

[Ove07] Ntrulike

[MTSB13] QC-LRPC

[ABDGZ16] Ntrulike

Bottom Line

Lack a Proof

Lack Efficiency
Motivations

Key Sizes
- 80's
- 00's

Rank Metric

Group action

[Gab05]
[Mb09] dyadic
[BCGO09] alternant

[GB08] QC-LDPC

[MB09] QC-MDPC

[MB09] QC-LRPC

[Ove07] Ntru-like

[Lack a proof]

Attacks

Other variations
Most of them broken

Bottom Line

[ME78] → [Nie86] → [Gab91]

RS
BCH
Goppa
RM

80's
00's

Security proof

[Nie86] → [Gab05]

[Ove07] QC-LRPC

[MTSB13] Ntru-like

[ABDGZ16] HQC

[AE03]
Motivations

Key Sizes
- 80's
- 00's

Rank Metric
- [ME78]
- [Nie86]
- [Gab05]
- [Gab91]
- [Ove07]
- [MB09]
- [BBC08]
- [BCGO09]
- [BBC08]
- [GMRZ13]
- [MTSB13]
- [MTRZ13]
- [Ntru-like]
- [QC-LDPC]
- [QC-LRPC]
- [QC-MDPC]
- [Ntru-like]
- [RS]
- [BCH]
- [Goppa]
- [RM]

Group action
- dyadic
- alternant

Security proof
- [Ale03]

Attacks
- 80's
- 00's
- [Ntru-like]
- Most of them broken

Bottom Line
- Lack a Proof
- Lacks Efficiency
Motivations

Key Sizes
- 80's
  - [ME78] → [Nie86] → [BCH, Goppa, RM] → [Gab91]
- 00's
  - Attacks
    - [Ove07] QC-LRPC
    - [BBC08] QC-LDPC
    - [GMRZ13] QC-MDPC
  - Other variations
    - Most of them broken

Group action
- [Gab05]
  - Ntru-like
    - [MB09] dyadic
    - [BCGO09] alternant
  - QC-MDPC
    - [MTSB13]

Rank Metric
- [Ale03]
  - Security proof
  - [ABDGZ16] HQC
  - RQC

Bottom Line
- Lack a Proof
- Lack Efficiency

Security proof
- [ME78] → [Nie86] → [BCH, Goppa, RM] → [Gab91] → [Gab05] → Ntru-like → QC-MDPC

Group action
- [Ale03] → HQC
- RQC

Rank Metric
- [Nie86] → [Gab91] → [Gab05] → Ntru-like → QC-MDPC

Key Sizes
- 80's
  - [ME78] → [Nie86] → [BCH, Goppa, RM] → [Gab91] → [Gab05] → Ntru-like → QC-MDPC
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
2. Security
3. A few words on key exchange protocols
4. NIST’s call for standardization of post-quantum algorithms
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
   - McEliece Encryption
   - Alekhnovich
   - Hamming Quasi-Cyclic

2. Security

3. A few words on key exchange protocols

4. NIST's call for standardization of post-quantum algorithms
Code-Based Encryption: McEliece

Key Generation:
- \( C[n, k] \) linear code, generated by \( \mathbf{G} \in \mathbb{F}_q^{k \times n} \), decoding up to \( t \) errors
- \( S \leftarrow \mathbb{F}_q^{k \times k} \) invertible, \( P \leftarrow \mathbb{F}_2^{n \times n} \) permutation

\[ \rightarrow \text{pk} = (\tilde{\mathbf{G}} = SP, t), \text{sk} = (S, G, P) \]

Encryption (of \( \mu \in \mathbb{F}_q^k \)):
- \( e \leftarrow \mathbb{F}_q^n \), with \( \omega(e) = t \)

\[ \rightarrow \mathbf{c} = \mu\tilde{\mathbf{G}} + e \]

Decryption:
- \( \tilde{\mu} = \mathcal{C} \cdot \text{Decode}(\mathbf{cP}^{-1}) \)

\[ \rightarrow \tilde{\mu}S^{-1} \]
Code-Based Encryption: McEliece

Key Generation:

- $\mathcal{C}[n,k]$ linear code, generated by $G \in \mathbb{F}_q^{k \times n}$, decoding up to $t$ errors
- $S \leftarrow \mathbb{F}_q^{k \times k}$ invertible, $P \leftarrow \mathbb{F}_2^{n \times n}$ permutation

\[ \rightarrow pk = (\tilde{G} = SGP, t), sk = (S, G, P) \]

Encryption (of $\mu \in \mathbb{F}_q^k$):

- $e \leftarrow \mathbb{F}_q^n$, with $\omega(e) = t$

\[ \rightarrow c = \mu \tilde{G} + e \]

Decryption:

- $\tilde{\mu} = \mathcal{C}.\text{Decode}(cP^{-1})$

\[ \rightarrow \tilde{\mu}S^{-1} \]
Code-Based Encryption: McEliece

Key Generation:
- $C[n, k]$ linear code, generated by $G \in \mathbb{F}_q^{k \times n}$, decoding up to $t$ errors
- $S \overset{\$}{\leftarrow} \mathbb{F}_q^{k \times k}$ invertible, $P \overset{\$}{\leftarrow} \mathbb{F}_2^{n \times n}$ permutation

\[
pk = (\tilde{G} = SGP, t), \ sk = (S, G, P)
\]

Encryption (of $\mu \in \mathbb{F}_q^k$):
- $e \overset{\$}{\leftarrow} \mathbb{F}_q^n$, with $\omega(e) = t$

\[
c = \mu \tilde{G} + e
\]

Decryption:
- $\tilde{\mu} = C.\text{Decode}(cP^{-1})$

\[
\tilde{\mu}S^{-1}
\]
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
   - McEliece Encryption
   - Alekhnovich
   - Hamming Quasi-Cyclic

2. Security

3. A few words on key exchange protocols

4. NIST’s call for standardization of post-quantum algorithms
Alekhnovich first cryptosystem (1/2)

Hypothesis: Decisional Decoding Hypothesis with parameter $t$

Let $0 < R_1 < R_2 < 1$. Consider $k, n$ such that $R_1 \leq k/n \leq R_2$, $C$ a random code (generated by $G$), and a vector which is either:

(i) a uniformly random vector $u$

(ii) $c + e$ where $c \in C$ is a unif. rand. codeword and $e$ a unif. rand. error of weight $t$, ind. of $c$.

There is no polynomial-time decoding algorithm $A$ that decides between (i) and (ii) with a non-negligible advantage over random choice.

Key generation

- $A \leftarrow \mathbb{F}_2^{k \times n}$, $e \leftarrow S_t^n(\mathbb{F}_2)$
- $y \leftarrow xA + e$, for $x \leftarrow \mathbb{F}_2^n$
- $pk = H \leftarrow \begin{pmatrix} A \\ y \end{pmatrix}$
- $G = H^\perp$ generating $C$
- $sk = e$
Hypothesis: Decisional Decoding Hypothesis with parameter $t$

Let $0 < R_1 < R_2 < 1$. Consider $k, n$ such that $R_1 \leq k/n \leq R_2$, $C$ a random code (generated by $G$), and a vector which is either:

(i) a uniformly random vector $u$
(ii) $c + e$ where $c \in C$ is a unif. rand. codeword and $e$ a unif. rand. error of weight $t$, ind. of $c$.

There is no polynomial-time decoding algorithm $A$ that decides between (i) and (ii) with a non-negligible advantage over random choice.

**Key generation**

- $A \leftarrow \mathbb{F}_2^{k \times n}$, $e \leftarrow \mathcal{S}_t^n(\mathbb{F}_2)$
- $y \leftarrow xA + e$, for $x \leftarrow \mathbb{F}_2^n$
- $G = H^\perp$ generating $C$
- $sk = e$
- $pk = H \leftarrow \begin{pmatrix} A \\ y \end{pmatrix}$
Alekhnovich first cryptosystem (2/2)

Encryption

- $c_0 = \text{Enc}_{pk}(0) = u \leftarrow \mathbb{F}_2^n$
- $c_1 = \text{Enc}_{pk}(1) = c + t, (c, t) \leftarrow C \times S_t^n(\mathbb{F}_2)$

Decryption

- Return $b = \langle e, c \rangle$

$(\langle e, c_1 \rangle = \langle e, c \rangle + \langle e, t \rangle = \langle e, t \rangle)$

Security reduction

Alekhnovich showed that a distinguisher between $u$ and $c + e$ yields an algorithm to decode up to $t$ errors.

→ Can be adapted to larger plaintext spaces, but results in inefficient cryptosystems...
Alekhnovich first cryptosystem (2/2)

Encryption

- $c_0 = \text{Enc}_{pk}(0) = u \leftarrow \mathbb{F}_2^n$
- $c_1 = \text{Enc}_{pk}(1) = c + t, (c, t) \leftarrow C \times S_t^n(\mathbb{F}_2)$

Decryption

- Return $b = \langle e, c \rangle$
- $(\langle e, c_1 \rangle = \langle e, c \rangle + \langle e, t \rangle = \langle e, t \rangle)$

Security reduction

Alekhnovich showed that a distinguisher between $u$ and $c + e$ yields an algorithm to decode up to $t$ errors.

→ Can be adapted to larger plaintext spaces, but results in inefficient cryptosystems...
Alekhnovich first cryptosystem (2/2)

Encryption

- \( c_0 = \text{Enc}_{pk}(0) = u \xleftarrow{\$} \mathbb{F}_2^n \)
- \( c_1 = \text{Enc}_{pk}(1) = c + t, (c, t) \xleftarrow{\$} C \times S_t^n(\mathbb{F}_2) \)

Decryption

- Return \( b = \langle e, c \rangle \)
- \( \langle e, c_1 \rangle = \langle e, c \rangle + \langle e, t \rangle = \langle e, t \rangle \)

Security reduction

Alekhnovich showed that a distinguisher between \( u \) and \( c + e \) yields an algorithm to decode up to \( t \) errors.

→ Can be adapted to larger plaintext spaces, but results in inefficient cryptosystems...
Alekhnovich first cryptosystem (2/2)

Encryption

- $c_0 = Enc_{pk}(0) = u \xleftarrow{\$} \mathbb{F}_2^n$
- $c_1 = Enc_{pk}(1) = c + t, (c, t) \xleftarrow{\$} C \times S_t^n(\mathbb{F}_2)$

Decryption

- Return $b = \langle e, c \rangle$
- $\langle e, c_1 \rangle = \langle e, c \rangle + \langle e, t \rangle = \langle e, t \rangle$

Security reduction

Alekhnovich showed that a distinguisher between $u$ and $c + e$ yields an algorithm to decode up to $t$ errors.

$\rightarrow$ Can be adapted to larger plaintext spaces, but results in inefficient cryptosystems...
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
   - McEliece Encryption
   - Alekhnovich
   - Hamming Quasi-Cyclic

2. Security

3. A few words on key exchange protocols

4. NIST’s call for standardization of post-quantum algorithms
Encryption scheme in Hamming metric, using Quasi-Cyclic Codes

- Notation: Secret data - Public data - One-time Randomness
- \( G \) is the generator matrix of some public code \( C \).

Alice

\[
\begin{align*}
\text{seed}_h & \xleftarrow{\$} \{0,1\}^\lambda, \quad h \xleftarrow{\$} \mathbb{F}_2^n \\
x, y & \xleftarrow{\$} S_w^n(\mathbb{F}_2), \quad s \leftarrow x + hy \\
\mu & \leftarrow C.\text{Decode}(\rho - vy)
\end{align*}
\]

Bob

\[
\begin{align*}
\text{seed}_h & \xleftarrow{\$} \{0,1\}^\lambda, \quad h \xleftarrow{\$} \mathbb{F}_2^n \\
r_1, r_2 & \xleftarrow{\$} S_w^n(\mathbb{F}_2), \quad \epsilon \xleftarrow{\$} S_{cw}^n(\mathbb{F}_2) \\
v & \leftarrow r_1 + hr_2, \quad \rho \leftarrow \mu G + sr_2 + \epsilon
\end{align*}
\]
HQC Encryption Scheme \([ABD^{+}18]\)

Encryption scheme in Hamming metric, using Quasi-Cyclic Codes

- **Notation**: Secret data - Public data - One-time Randomness
- **\(G\)** is the generator matrix of some public code \(C\).

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{seed}_h \xleftarrow{$} {0, 1}^\lambda, \quad h \xleftarrow{$} \mathbb{F}_2^n)</td>
<td>(r_1, r_2 \xleftarrow{$} S_w(\mathbb{F}<em>2), \quad \epsilon \xleftarrow{$} S</em>{cw}(\mathbb{F}_2))</td>
</tr>
<tr>
<td>(x, y \xleftarrow{$} S_w(\mathbb{F}_2), \quad s \leftarrow x + hy)</td>
<td>(v \leftarrow r_1 + hr_2, \quad \rho \leftarrow \mu G + sr_2 + \epsilon)</td>
</tr>
<tr>
<td>(\mu \leftarrow C.\text{Decode}(\rho - vy))</td>
<td>(\text{seed}_h, s\rightarrow \quad v, \rho\rightarrow)</td>
</tr>
</tbody>
</table>
HQC Encryption Scheme \([\text{ABD}^+18]\)

Encryption scheme in Hamming metric, using Quasi-Cyclic Codes

- **Notation**: Secret data - Public data - One-time Randomness
- **\(G\)** is the generator matrix of some public code \(C\).

\[
\begin{align*}
\text{Alice} & \quad \text{Bob} \\
seed_h & \overset{\$}{\leftarrow} \{0, 1\}^\lambda, \quad h \overset{\text{seed}_h}{\leftarrow} \mathbb{F}_2^n \\
x, y & \overset{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_2), \quad s \leftarrow x + hy
\end{align*}
\]

\[
\begin{align*}
\mu & \leftarrow C.\text{Decode}(\rho - vy) \\
r_1, r_2 & \overset{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_2), \quad \epsilon \overset{\$}{\leftarrow} \mathcal{S}_{cw}^n(\mathbb{F}_2) \\
v & \leftarrow r_1 + hr_2, \quad \rho \leftarrow \mu G + sr_2 + \epsilon
\end{align*}
\]
Encryption scheme in Hamming metric, using Quasi-Cyclic Codes

- Notation: Secret data - Public data - One-time Randomness
- $G$ is the generator matrix of some public code $C$.

Alice

\[
\begin{align*}
\text{seed}_h & \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda, \quad h \stackrel{\text{seed}_h}{\leftarrow} \mathbb{F}_2^n \\
x, y & \stackrel{\$}{\leftarrow} S_w^n(\mathbb{F}_2), \quad s \leftarrow x + hy \\
\mu & \leftarrow C.\text{Decode}(\rho - vy)
\end{align*}
\]

Bob

\[
\begin{align*}
r_1, r_2 & \stackrel{\$}{\leftarrow} S_w^n(\mathbb{F}_2), \quad \epsilon \stackrel{\$}{\leftarrow} S_{cw}^n(\mathbb{F}_2) \\
v & \leftarrow r_1 + hr_2, \quad \rho \leftarrow \mu G + sr_2 + \epsilon
\end{align*}
\]
Correctness

Correctness Property

$$\text{Decrypt}(sk, \text{Encrypt}(pk, \mu, \theta)) = \mu$$

C. Decode correctly decodes $\rho - v \cdot y$ whenever

- the error term is not too big
  $$\omega(s \cdot r_2 - v \cdot y + \epsilon) \leq \delta$$
  $$\omega((x + h \cdot y) \cdot r_2 - (r_1 + h \cdot r_2) \cdot y + \epsilon) \leq \delta$$
  $$\omega(x \cdot r_2 - r_1 \cdot y + \epsilon) \leq \delta$$

Error distribution analysis $\rightarrow$ Decryption failure probability better understood
Correctness

Correctness Property

\[ \text{Decrypt} (sk, \text{Encrypt} (pk, \mu, \theta)) = \mu \]

\[ C. \text{Decode correctly decodes } \rho - \mathbf{v} \cdot \mathbf{y} \text{ whenever} \]

the error term is \textbf{not too big}

\[ \omega (s \cdot r_2 - \mathbf{v} \cdot \mathbf{y} + \epsilon) \leq \delta \]

\[ \omega ((x + h \cdot y) \cdot r_2 - (r_1 + h \cdot r_2) \cdot y + \epsilon) \leq \delta \]

\[ \omega (x \cdot r_2 - r_1 \cdot y + \epsilon) \leq \delta \]

Error distribution analysis → Decryption failure probability better understood
Correctness

Correctness Property

Decrypt \((sk, \text{Encrypt}(pk, \mu, \theta))\) = \mu

C. Decode correctly decodes \(\rho - \mathbf{v} \cdot \mathbf{y}\) whenever

- the error term is not too big
  \[ \omega (s \cdot r_2 - \mathbf{v} \cdot \mathbf{y} + \epsilon) \leq \delta \]
  \[ \omega ((x + h \cdot y) \cdot r_2 - (r_1 + h \cdot r_2) \cdot y + \epsilon) \leq \delta \]
  \[ \omega (x \cdot r_2 - r_1 \cdot y + \epsilon) \leq \delta \]

Error distribution analysis → Decryption failure probability better understood
Decryption failure rate

- **In red**: Theoretical DFR with a reasonable assumption on the error distribution.
- **In black**: Observed/Empirical DFR, obtained by running $10^5$ encryptions/decryptions over $10^3$ codes with $n_1 = 766$, $k_1 = 256$, $\delta_1 = 57$, $w = 67$, $w_r = 77$ and varying $n_2$. 

\[ \log_2(DFR) \]
## Parameters

<table>
<thead>
<tr>
<th>Instance</th>
<th>$n_1$</th>
<th>$n_2$</th>
<th>$n \approx n_1 n_2$</th>
<th>$k_1$</th>
<th>$\delta_1$</th>
<th>$w$</th>
<th>$w_r = w_e$</th>
<th>security</th>
<th>$p_{\text{fail}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic-I</td>
<td>766</td>
<td>29</td>
<td>22,229</td>
<td>256</td>
<td>57</td>
<td>67</td>
<td>77</td>
<td>128</td>
<td>$&lt; 2^{-64}$</td>
</tr>
<tr>
<td>Basic-II</td>
<td>766</td>
<td>31</td>
<td>23,747</td>
<td>256</td>
<td>57</td>
<td>67</td>
<td>77</td>
<td>128</td>
<td>$&lt; 2^{-96}$</td>
</tr>
<tr>
<td>Basic-III</td>
<td>796</td>
<td>31</td>
<td>24,677</td>
<td>256</td>
<td>57</td>
<td>67</td>
<td>77</td>
<td>128</td>
<td>$&lt; 2^{-128}$</td>
</tr>
<tr>
<td>Advanced-I</td>
<td>796</td>
<td>51</td>
<td>40,597</td>
<td>256</td>
<td>60</td>
<td>101</td>
<td>117</td>
<td>192</td>
<td>$&lt; 2^{-64}$</td>
</tr>
<tr>
<td>Advanced-II</td>
<td>766</td>
<td>57</td>
<td>43,669</td>
<td>256</td>
<td>57</td>
<td>101</td>
<td>117</td>
<td>192</td>
<td>$&lt; 2^{-128}$</td>
</tr>
<tr>
<td>Advanced-III</td>
<td>766</td>
<td>61</td>
<td>46,747</td>
<td>256</td>
<td>57</td>
<td>101</td>
<td>117</td>
<td>192</td>
<td>$&lt; 2^{-192}$</td>
</tr>
<tr>
<td>Paranoiac-I</td>
<td>766</td>
<td>77</td>
<td>59,011</td>
<td>256</td>
<td>57</td>
<td>133</td>
<td>153</td>
<td>256</td>
<td>$&lt; 2^{-64}$</td>
</tr>
<tr>
<td>Paranoiac-II</td>
<td>766</td>
<td>83</td>
<td>63,587</td>
<td>256</td>
<td>57</td>
<td>133</td>
<td>153</td>
<td>256</td>
<td>$&lt; 2^{-128}$</td>
</tr>
<tr>
<td>Paranoiac-III</td>
<td>796</td>
<td>85</td>
<td>67,699</td>
<td>256</td>
<td>60</td>
<td>133</td>
<td>153</td>
<td>256</td>
<td>$&lt; 2^{-192}$</td>
</tr>
<tr>
<td>Paranoiac-IV</td>
<td>796</td>
<td>89</td>
<td>70,853</td>
<td>256</td>
<td>60</td>
<td>133</td>
<td>153</td>
<td>256</td>
<td>$&lt; 2^{-256}$</td>
</tr>
</tbody>
</table>
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
2. Security
   - Security Model and Hybrid Argument
   - HQC Security
3. A few words on key exchange protocols
4. NIST’s call for standardization of post-quantum algorithms
Security Model and Hybrid Argument

- **Key exchange as an encryption scheme**
  - Same as Ding et al. [Din12, DXL12], Peikert’s [Pei14], BCNS [BCNS15] and **NEWHOPE** [ADPS16]

- **Usual game:**

\[
\text{Exp}^{\text{ind} - b}_{\mathcal{E}, \mathcal{A}}(\lambda) \\
1. \text{param} \leftarrow \text{Setup}(1^\lambda) \\
2. (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{param}) \\
3. (\epsilon_0, \epsilon_1) \leftarrow \mathcal{A}(\text{FIND} : \text{pk}) \\
4. c^* \leftarrow \text{Encrypt}(\text{pk}, \epsilon_b, \theta) \\
5. b' \leftarrow \mathcal{A}(\text{GUESS} : c^*) \\
6. \text{RETURN} \ b'
\]

- **Hybrid argument:**
  - Construct a sequence of games transitioning from Enc(\epsilon_0) to Enc(\epsilon_1)
  - Prove they are indistinguishable one from another
Security Model and Hybrid Argument

- Key exchange as an encryption scheme
- Same as Ding et al. [Din12, DXL12], Peikert’s [Pei14], BCNS [BCNS15] and NewHope [ADPS16]

Usual game:

\[ \text{Exp}^{\text{ind}-b} (\lambda) \]
1. param $\leftarrow \text{Setup}(1^\lambda)$
2. (pk, sk) $\leftarrow \text{KeyGen}(\text{param})$
3. (\epsilon_0, \epsilon_1) $\leftarrow \mathcal{A}(\text{FIND} : \text{pk})$
4. $c^* \leftarrow \text{Encrypt}(\text{pk}, \epsilon_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\text{GUESS} : c^*)$
6. RETURN $b'$

Hybrid argument:
- Construct a sequence of games transitioning from Enc($\epsilon_0$) to Enc($\epsilon_1$)
- Prove they are indistinguishable one from another
Security Model and Hybrid Argument

- Key exchange as an encryption scheme
- Same as Ding et al. [Din12, DXL12], Peikert’s [Pei14], BCNS [BCNS15] and NewHope [ADPS16]
- Usual game:

\[
\text{Exp}_{\mathcal{A}}^{\text{ind}}(\lambda)
\]
1. param $\leftarrow$ Setup$(1^\lambda)$
2. $(pk, sk) \leftarrow$ KeyGen$(\text{param})$
3. $(\epsilon_0, \epsilon_1) \leftarrow \mathcal{A}(\text{FIND} : pk)$
4. $c^* \leftarrow \text{Encrypt}(pk, \epsilon_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\text{GUESS} : c^*)$
6. RETURN $b'$

- Hybrid argument:
  - Construct a sequence of games transitioning from Enc($\epsilon_0$) to Enc($\epsilon_1$)
  - Prove they are indistinguishable one from another
Security Model and Hybrid Argument

- Key exchange as an encryption scheme
- Same as Ding et al. [Din12, DXL12], Peikert’s [Pei14], BCNS [BCNS15] and NewHope [ADPS16]

Usual game:

1. param $\leftarrow$ Setup($1^\lambda$)
2. (pk, sk) $\leftarrow$ KeyGen(param)
3. ($\epsilon_0, \epsilon_1$) $\leftarrow$ $\mathcal{A}$(FIND : pk)
4. $c^* \leftarrow$ Encrypt(pk, $\epsilon_b, \theta$)
5. $b' \leftarrow$ $\mathcal{A}$(GUESS : $c^*$)
6. RETURN $b'$

Hybrid argument:

- Construct a sequence of games transitioning from Enc($\epsilon_0$) to Enc($\epsilon_1$)
- Prove they are indistinguishable one from another
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
2. Security
   - Security Model and Hybrid Argument
   - HQC Security
3. A few words on key exchange protocols
4. NIST’s call for standardization of post-quantum algorithms
Security

Definition (SD Distribution)

For positive integers $n$, $k$, and $w$, the $SD(n, k, w)$ Distribution chooses $H \leftarrow \mathbb{F}^{(n-k) \times n}$ and $x \leftarrow \mathbb{F}^n$ such that $\omega(x) = w$, and outputs $(H, Hx^T)$.

Definition (Decisional $s$-QCSD Problem)

For positive integers $n$, $k$, $w$, $s$, a random parity check matrix $H$ of a QC code $C$ and $y \leftarrow \mathbb{F}^n$, the Decisional $s$-Quasi-Cyclic SD Problem $s$-DQCSD$(n, k, w)$ asks to decide with non-negligible advantage whether $(H, y^T)$ came from the $s$-QCSD$(n, k, w)$ distribution or the uniform distribution over $\mathbb{F}^{(n-k) \times n} \times \mathbb{F}^{n-k}$.

Theorem

HQC is IND-CPA under the 2-DQCSD and 3-DQCSD assumptions. → sketch of proof
**Security**

**Definition (SD Distribution)**

For positive integers, \( n, k, \) and \( w, \) the \( SD(n, k, w) \) Distribution chooses \( H \leftarrow \mathbb{F}^{(n-k) \times n} \) and \( x \leftarrow \mathbb{F}^n \) such that \( \omega(x) = w, \) and outputs \((H, Hx^\top)\).

**Definition (Decisional s-QCSD Problem)**

For positive integers \( n, k, w, s, \) a random parity check matrix \( H \) of a QC code \( C \) and \( y \leftarrow \mathbb{F}^n, \) the Decisional \( s\)-Quasi-Cyclic SD Problem \( s\text{-DQCSD}(n, k, w) \) asks to decide with non-negligible advantage whether \((H, y^\top)\) came from the \( s\)-QCSD\((n, k, w)\) distribution or the uniform distribution over \( \mathbb{F}^{(n-k) \times n} \times \mathbb{F}^{n-k} \).

**Theorem**

\( HQC \) is IND-CPA under the 2-DQCSD and 3-DQCSD assumptions. → sketch of proof
Security

Definition (SD Distribution)

For positive integers, \( n, k, \) and \( w \), the \( SD(n, k, w) \) Distribution chooses \( H \leftarrow F^{(n-k)\times n} \) and \( x \leftarrow F^n \) such that \( \omega(x) = w \), and outputs \((H, Hx^\top)\).

Definition (Decisional s-QCSD Problem)

For positive integers \( n, k, w, s \), a random parity check matrix \( H \) of a QC code \( C \) and \( y \leftarrow F^n \), the Decisional s-Quasi-Cyclic SD Problem \( s-DQCSD(n, k, w) \) asks to decide with non-negligible advantage whether \((H, y^\top)\) came from the s-QCSD\((n, k, w)\) distribution or the uniform distribution over \( F^{(n-k)\times n} \times F^{n-k} \).

Theorem

HQC is IND-CPA under the 2-DQCSD and 3-DQCSD assumptions. \( \rightarrow \) sketch of proof
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
2. Security
3. A few words on key exchange protocols
   - Cyclic Error Decoding
   - BitFlipping algorithm
   - Parameters
4. NIST’s call for standardization of post-quantum algorithms
A particular decoding

- HQC requires $x \cdot r_2 - r_1 \cdot y + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error [DGZ17]

**Cyclic Error Decoding (CED) Problem**

- Let $x, y, r_1, r_2 \overset{\$}{\leftarrow} S_n^w(F_2)$ with $w = O(\sqrt{n})$, and $e \overset{\$}{\leftarrow} S_{cw}(F_2)$ a random error vector.
- Given $(x, y) \in (S_n^w(F_2))^2$ and $e_c \leftarrow x r_2 - y r_1 + e$ such that $\omega(r_1) = \omega(r_2) = w$, find $(r_1, r_2)$.

- This is essentially a *noisy* SD problem
A particular decoding

- HQC requires $x \cdot r_2 - r_1 \cdot y + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error [DGZ17]

Cyclic Error Decoding (CED) Problem

- Let $x, y, r_1, r_2 \leftarrow S_n^w(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $e \leftarrow S_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(x, y) \in (S_n^w(\mathbb{F}_2))^2$ and $e_c \leftarrow xr_2 - yr_1 + e$ such that $\omega(r_1) = \omega(r_2) = w$, find $(r_1, r_2)$.

- This is essentially a noisy SD problem
A particular decoding

- HQC requires $x \cdot r_2 - r_1 \cdot y + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error [DGZ17]

**Cyclic Error Decoding (CED) Problem**

- Let $x, y, r_1, r_2 \leftarrow S_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $e \leftarrow S_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(x, y) \in (S_w^n(\mathbb{F}_2))^2$ and $e_c \leftarrow xr_2 - yr_1 + e$ such that $\omega(r_1) = \omega(r_2) = w$, find $(r_1, r_2)$.

- This is essentially a noisy SD problem
A particular decoding

- HQC requires $x \cdot r_2 - r_1 \cdot y + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error [DGZ17]

Cyclic Error Decoding (CED) Problem

- Let $x, y, r_1, r_2 \leftarrow S_w^n(F_2)$ with $w = O(\sqrt{n})$, and $e \leftarrow S_{cw}^n(F_2)$ a random error vector.
- Given $(x, y) \in (S^n_w(F_2))^2$ and $e_c \leftarrow x r_2 - y r_1 + e$ such that $\omega(r_1) = \omega(r_2) = w$, find $(r_1, r_2)$.

- This is essentially a noisy SD problem

\[ x \xrightarrow[]{} -y \xrightarrow[]{} \]
A particular decoding

- HQC requires $x \cdot r_2 - r_1 \cdot y + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error [DGZ17]

Cyclic Error Decoding (CED) Problem

- Let $x, y, r_1, r_2 \leftarrow S^n_w(F_2)$ with $w = O(\sqrt{n})$, and $e \leftarrow S^n_{cw}(F_2)$ a random error vector.
- Given $(x, y) \in (S^n_w(F_2))^2$ and $e_c \leftarrow xr_2 - yr_1 + e$ such that $\omega(r_1) = \omega(r_2) = w$, find $(r_1, r_2)$.

- This is essentially a noisy SD problem
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
2. Security
3. A few words on key exchange protocols
   - Cyclic Error Decoding
   - BitFlipping algorithm
   - Parameters
4. NIST’s call for standardization of post-quantum algorithms
Hard Decision Decoding: BitFlipping

- Iterative decoding for Low Density Parity Check codes [Gal62]
- Decoding capacity increase linearly with the code length (for LDPC)

Intuition

- Compute the number of unsatisfied parity-check equations for each bit of the message
- If this number is greater than some threshold, flip the bit and go to 1.
- Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand, implement, and natively pretty efficient
- The threshold value is crucial [CS16]

Ouroboros (aka. BIKE-3)

The BitFlipping algorithm can be modified to handle noisy syndrome (for almost free!).
Hard Decision Decoding: BitFlipping

- Iterative decoding for **Low Density Parity Check** codes [Gal62]
- Decoding capacity increase linearly with the code length (for LDPC)

**Intuition**

1. Compute the number of unsatisfied parity-check equations for each bit of the message
2. If this number is greater than some *threshold*, flip the bit and go to 1.
3. Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand, implement, and natively pretty efficient
- The threshold value is crucial [CS16]

Ouroboros (aka. BIKE-3)
The BitFlipping algorithm can be modified to handle noisy syndrome (for almost free!).
Hard Decision Decoding: BitFlipping

- Iterative decoding for **Low Density Parity Check** codes [Gal62]
- Decoding capacity increase linearly with the code length (for LDPC)

**Intuition**

1. Compute the number of unsatisfied parity-check equations for each bit of the message.
2. If this number is greater than some *threshold*, flip the bit and go to 1.
3. Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand, implement, and natively pretty efficient
- The threshold value is crucial [CS16]

**Ouroboros (aka. BIKE-3)**

The BitFlipping algorithm can be modified to handle noisy syndrome (for almost free!).
Hard Decision Decoding: BitFlipping

- Iterative decoding for Low Density Parity Check codes [Gal62]
- Decoding capacity increase linearly with the code length (for LDPC)

Intuition

1. Compute the number of unsatisfied parity-check equations for each bit of the message
2. If this number is greater than some threshold, flip the bit and go to 1.
3. Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand, implement, and natively pretty efficient
- The threshold value is crucial [CS16]

Ouroboros (aka. BIKE-3)

The BitFlipping algorithm can be modified to handle noisy syndrome (for almost free!).

---

J.-C. Deneuville

A new framework for code-based encryption and more.

April the 5th, 2018 21 / 31
Hard Decision Decoding: BitFlipping

- Iterative decoding for Low Density Parity Check codes [Gal62]
- Decoding capacity increase linearly with the code length (for LDPC)

Intuition

1. Compute the number of unsatisfied parity-check equations for each bit of the message.
2. If this number is greater than some threshold, flip the bit and go to 1.
3. Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand, implement, and natively pretty efficient
- The threshold value is crucial [CS16]

Ouroboros (aka. BIKE-3)

The BitFlipping algorithm can be modified to handle noisy syndrome (for almost free!).
Hard Decision Decoding: BitFlipping

- Iterative decoding for Low Density Parity Check codes [Gal62]
- Decoding capacity increase linearly with the code length (for LDPC)

Intuition

1. Compute the number of unsatisfied parity-check equations for each bit of the message
2. If this number is greater than some threshold, flip the bit and go to 1.
3. Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand, implement, and natively pretty efficient
- The threshold value is crucial [CS16]

Ouroboros (aka. BIKE-3)
The BitFlipping algorithm can be modified to handle noisy syndrome (for almost free!).
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes

2. Security

3. A few words on key exchange protocols
   - Cyclic Error Decoding
   - BitFlipping algorithm
   - Parameters

4. NIST’s call for standardization of post-quantum algorithms
## Reduction Compliant Parameters

<table>
<thead>
<tr>
<th>Instance</th>
<th>( n )</th>
<th>( w )</th>
<th>( w_e )</th>
<th>threshold</th>
<th>security</th>
<th>DFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-I</td>
<td>5,851</td>
<td>47</td>
<td>94</td>
<td>30</td>
<td>80</td>
<td>( 0.92 \cdot 10^{-5} )</td>
</tr>
<tr>
<td>Low-II</td>
<td>5,923</td>
<td>47</td>
<td>94</td>
<td>30</td>
<td>80</td>
<td>( 2.3 \cdot 10^{-6} )</td>
</tr>
<tr>
<td>Medium-I</td>
<td>13,691</td>
<td>75</td>
<td>150</td>
<td>45</td>
<td>128</td>
<td>( 0.96 \cdot 10^{-5} )</td>
</tr>
<tr>
<td>Medium-II</td>
<td>14,243</td>
<td>75</td>
<td>150</td>
<td>45</td>
<td>128</td>
<td>( 1.09 \cdot 10^{-6} )</td>
</tr>
<tr>
<td>Strong-I</td>
<td>40,013</td>
<td>147</td>
<td>294</td>
<td>85</td>
<td>256</td>
<td>( 4.20 \cdot 10^{-5} )</td>
</tr>
<tr>
<td>Strong-II</td>
<td>40,973</td>
<td>147</td>
<td>294</td>
<td>85</td>
<td>256</td>
<td>( &lt; 10^{-6} )</td>
</tr>
</tbody>
</table>

**Table:** Parameter sets for Ouroboros
## Optimized Parameters (wrt. Best Known Attacks)

<table>
<thead>
<tr>
<th>Instance</th>
<th>$n$</th>
<th>$w$</th>
<th>$w_e$</th>
<th>threshold</th>
<th>security</th>
<th>DFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-I</td>
<td>4,813</td>
<td>41</td>
<td>123</td>
<td>27</td>
<td>80</td>
<td>$2.23 \cdot 10^{-5}$</td>
</tr>
<tr>
<td>Low-II</td>
<td>5,003</td>
<td>41</td>
<td>123</td>
<td>27</td>
<td>80</td>
<td>$2.60 \cdot 10^{-6}$</td>
</tr>
<tr>
<td>Medium-I</td>
<td>10,301</td>
<td>67</td>
<td>201</td>
<td>42</td>
<td>128</td>
<td>$1.01 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>Medium-II</td>
<td>10,837</td>
<td>67</td>
<td>201</td>
<td>42</td>
<td>128</td>
<td>$&lt; 10^{-7}$</td>
</tr>
<tr>
<td>Strong-I</td>
<td>32,771</td>
<td>131</td>
<td>393</td>
<td>77</td>
<td>256</td>
<td>$&lt; 10^{-4}$</td>
</tr>
<tr>
<td>Strong-II</td>
<td>33,997</td>
<td>131</td>
<td>393</td>
<td>77</td>
<td>256</td>
<td>$&lt; 10^{-7}$</td>
</tr>
</tbody>
</table>

**Table:** Optimized parameter sets for Ouroboros in Hamming metric
Outline

1. HQC: Efficient encryption from random quasi-cyclic codes
2. Security
3. A few words on key exchange protocols
4. NIST’s call for standardization of post-quantum algorithms
NIST’s call for standardization of post-quantum algorithms

- 3rd call for standardization
- Asks for post-quantum cryptographic algorithms
- 3 categories:
  - Encryption
  - Key exchange
  - Signature
- Many candidates:
  - Error correcting codes,
  - Lattices,
  - Multivariate,
  - Hash functions,
  - ...

- November 2016: announcement
- November 2017: submission deadline (82 submissions)
- December 2017: 1st round (02-20-18: 66 concurrents)
- April 2018: 1st standardization conference
- ...
- In 3 to 5 years, several algorithms will eventually be standardized
NIST’s call for standardization of post-quantum algorithms

- 3rd call for standardization
- Asks for post-quantum cryptographic algorithms
- 3 categories:
  - Encryption
  - Key exchange
  - Signature
- Many candidates:
  - Error correcting codes,
  - Lattices,
  - Multivariate,
  - Hash functions,
  - ...

- November 2016: announcement
- November 2017: submission deadline (82 submissions)
- December 2017: 1st round (02-20-18: 66 concurrents)
- April 2018: 1st standardization conference
- ...
- In 3 to 5 years, several algorithms will eventually be standardized
NIST’s call for standardization of post-quantum algorithms

NIST
National Institute of Standards and Technologies

- 3rd call for standardization
- Asks for post-quantum cryptographic algorithms
- 3 categories:
  - Encryption
  - Key exchange
  - Signature
- Many candidates:
  - Error correcting codes,
  - Lattices,
  - Multivariate,
  - Hash functions,
  - ...

- November 2016: announcement
- November 2017: submission deadline (82 submissions)
- December 2017: 1st round (02-20-18: 66 concurrents)
- April 2018: 1st standardization conference
- ...
- In 3 to 5 years, several algorithms will eventually be standardized
Overview

HQC (Hamming Quasi-Cyclic) is a code-based public key encryption scheme designed to provide security against attacks by both classical and quantum computers. It relies on quasi-cyclic codes as well as BCH codes. HQC has been submitted to the NIST’s Post-Quantum Cryptography Standardization Project.

Submitters (alphabetical order)

- Carlos Aguilar Melchor, University of Toulouse (FR)
- Nicolas Aragon, University of Limoges (FR)
- Slim Bettaieb, Worldline (FR)
- Loïc Bidoux, Worldline (FR)
- Olivier Blazy, University of Limoges (FR)
- Jean-Christophe Deneuville, INSA-CVL & University of Limoges (FR)
- Philippe Gaborit, University of Limoges (FR)
- Edoardo Persichetti, Florida Atlantic University (US)
- Gilles Zémor, University of Bordeaux (FR)

https://pqc-hqc.org/
Ouroboros aka. BIKE-3

BIKE - Bit Flipping Key Encapsulation

Welcome to the BIKE Website

This website will be used by the BIKE team as its official communication media.

BIKE is a code-based key encapsulation suite based on QC-MDPC (Quasi-Cyclic Moderate Density Parity-Check) codes, which was submitted to the NIST standardization process on post-quantum cryptography. The BIKE suite consists of three variants: BIKE-1, BIKE-2 and BIKE-3. Each variant offers different performance trade-offs.

Timeline

- 12/20/2017 - NIST accepts BIKE as a "complete and proper" submission.
- 11/30/2017 - BIKE is submitted to the NIST standardization process.

Specification Document

The specification document of BIKE-1, BIKE-2 and BIKE-3 can be found here.

http://bikesuite.org/
Other submissions

- Encryption schemes:
  - RQC (rank metric version of HQC) → https://pqc-rqc.org/
  - LOCKER (uses LRPC codes), no (public) website yet

- Key exchange protocols:
  - Ouroboros-R (rank metric version of Ouroboros) → https://pqc-ouroborosr.org/
  - LAKE (uses LRPC codes), no (public) website yet

All 1\textsuperscript{st} round submissions available here.
Conclusion

In this talk

- **HQC**: _efficient_ encryption from random quasi-cyclic codes, _without masking_ the underlying structure
- Ouroboros: a _secure, simple, and efficient_ code-based key exchange protocol
- Competitive parameters

Further Improvements
Conclusion

In this talk

- **HQC:** *efficient* encryption from random quasi-cyclic codes, *without masking* the underlying structure
- **Ouroboros:** a *secure, simple, and efficient* code-based key exchange protocol
- Competitive parameters

Further Improvements
Conclusion

In this talk

- **HQC**: efficient encryption from random quasi-cyclic codes, *without masking* the underlying structure
- **Ouroboros**: a *secure, simple, and efficient* code-based key exchange protocol
- Competitive parameters

Further Improvements
Conclusion

In this talk

- HQC: **efficient** encryption from random quasi-cyclic codes, *without masking* the underlying structure
- Ouroboros: a **secure**, **simple**, and **efficient** code-based key exchange protocol
- Competitive parameters

Further Improvements

- Improve BitFlipping threshold [CS16]
- Switching to Rank metric drastically improves parameters! → interlude?
- Optimize implementation
- More fancy code combinations for HQC
- Other codes? Other metrics? Lattices?
Conclusion

In this talk

- **HQC**: *efficient* encryption from random quasi-cyclic codes, *without masking* the underlying structure
- **Ouroboros**: a *secure, simple, and efficient* code-based key exchange protocol
- Competitive parameters

Further Improvements

- Improve BitFlipping threshold [CS16]
- Switching to Rank metric drastically improves parameters! → interlude?
- Optimize implementation
- More fancy code combinations for HQC
- Other codes? Other metrics? Lattices?
In this talk

- HQC: efficient encryption from random quasi-cyclic codes, without masking the underlying structure
- Ouroboros: a secure, simple, and efficient code-based key exchange protocol
- Competitive parameters

Further Improvements

- Improve BitFlipping threshold [CS16]
- Switching to Rank metric drastically improves parameters! → interlude?
- Optimize implementation
- More fancy code combinations for HQC
- Other codes? Other metrics? Lattices?
In this talk

- **HQC**: *efficient* encryption from random quasi-cyclic codes, *without masking* the underlying structure
- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Competitive parameters

Further Improvements

- Improve BitFlipping threshold [CS16]
- Switching to Rank metric drastically improves parameters! → interlude?
- Optimize implementation
  - More fancy code combinations for HQC
  - Other codes? Other metrics? Lattices?
Conclusion

In this talk

- **HQC**: *efficient* encryption from random quasi-cyclic codes, *without masking* the underlying structure
- **Ouroboros**: a *secure, simple, and efficient* code-based key exchange protocol
- Competitive parameters

Further Improvements

- Improve BitFlipping threshold [CS16]
- Switching to Rank metric drastically improves parameters! → interlude?
- Optimize implementation
- More fancy code combinations for HQC
- Other codes? Other metrics? Lattices?
Conclusion

In this talk

- HQC: efficient encryption from random quasi-cyclic codes, *without masking* the underlying structure
- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Competitive parameters

Further Improvements

- Improve BitFlipping threshold [CS16]
- Switching to Rank metric drastically improves parameters! → interlude?
- Optimize implementation
- More fancy code combinations for HQC
- Other codes? Other metrics? Lattices?
Thanks!

HQC (& RQC) available @ http://unil.im/HQC-RQC

Ouroboros available @ http://unil.im/ouroboros
Thanks!


Jintai Ding, Xiang Xie, and Xiaodong Lin.


HQC (& RQC) available @ http://unil.im/HQC-RQC

Ouroboros available @ http://unil.im/ouroboros
Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields
- $\mathbb{F}_q$ a finite field with $q$ a power of a prime.
- $\mathbb{F}_{q^m}$ an extension of degree $m$ of $\mathbb{F}_q$.
- $\mathbb{F}_{q^m}$ can be seen as a vector space on $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, \ldots, b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

Let $\mathbf{v} = (v_1, \ldots, v_n)$ be a word of length $n$ in $\mathbb{F}_{q^m}$.

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

Rank weight of word $\mathbf{v}$ has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is $r$.

Equivalently $\text{rank}(\mathbf{v}) = r \iff v_j \in V_r \subseteq \mathbb{F}_{q^m}^n$ with $\text{dim}(V_r) = r$. 
Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \( \mathbb{F}_q \) a finite field with \( q \) a power of a prime.
- \( \mathbb{F}_{q^m} \) an extension of degree \( m \) of \( \mathbb{F}_q \).
- \( \mathbb{F}_{q^m} \) can be seen as a vector space on \( \mathbb{F}_q \).
- \( B = (b_1, \ldots, b_m) \) a basis of \( \mathbb{F}_{q^m} \) over \( \mathbb{F}_q \).

Let \( \mathbf{v} = (v_1, \ldots, v_n) \) be a word of length \( n \) in \( \mathbb{F}_{q^m} \).

Any coordinate \( v_j = \sum_{i=1}^{m} v_{ij} b_i \) with \( v_{ij} \in \mathbb{F}_q \).

\[ \mathbf{v} = (v_1, \ldots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{pmatrix} \]

Rank weight of word

\( \mathbf{v} \) has rank \( r = \text{rank}(\mathbf{v}) \) iff the rank of \( \mathbf{V} = (v_{ij})_{ij} \) is \( r \).

Equivalently \( \text{rank}(\mathbf{v}) = r \iff v_j \in \mathbf{V}_r \subset \mathbb{F}_{q^m}^n \) with \( \dim(\mathbf{V}_r) = r \).
Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \( \mathbb{F}_q \) a finite field with \( q \) a power of a prime.
- \( \mathbb{F}_{q^m} \) an extension of degree \( m \) of \( \mathbb{F}_q \).
- \( \mathbb{F}_{q^m} \) can be seen as a vector space on \( \mathbb{F}_q \).
- \( \mathcal{B} = (b_1, \ldots, b_m) \) a basis of \( \mathbb{F}_{q^m} \) over \( \mathbb{F}_q \).

Let \( \mathbf{v} = (v_1, \ldots, v_n) \) be a word of length \( n \) in \( \mathbb{F}_{q^m} \).

Any coordinate \( v_j = \sum_{i=1}^m v_{ij} b_i \) with \( v_{ij} \in \mathbb{F}_q \).

\[
\mathbf{v} = (v_1, \ldots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix}
  v_{11} & v_{12} & \cdots & v_{1n} \\
  v_{21} & v_{22} & \cdots & v_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  v_{m1} & v_{m2} & \cdots & v_{mn}
\end{pmatrix}
\]

Rank weight of word

\( \mathbf{v} \) has rank \( r = \text{rank}(\mathbf{v}) \) iff the rank of \( \mathbf{V} = (v_{ij})_{ij} \) is \( r \).

Equivalently \( \text{rank}(\mathbf{v}) = r \iff v_j \in V_r \subseteq \mathbb{F}_{q^m}^n \) with \( \text{dim}(V_r) = r \).
Rank Metric Interlude (2/2)

- Best Known Attacks have worse complexity in rank metric \(2^{O(n^2)}\) than in Hamming metric \(2^{O(n)}\).
- Consequence: worse attacks \(\Rightarrow\) better parameters.

<table>
<thead>
<tr>
<th>Instance</th>
<th>key size (bits)</th>
<th>(n)</th>
<th>(m)</th>
<th>(q)</th>
<th>(w)</th>
<th>security</th>
<th>decoding failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ouroboros-R-I</td>
<td>1,591</td>
<td>37</td>
<td>43</td>
<td>2</td>
<td>5</td>
<td>100</td>
<td>10(^{-4})</td>
</tr>
<tr>
<td>Ouroboros-R-II</td>
<td>2,809</td>
<td>53</td>
<td>53</td>
<td>2</td>
<td>5</td>
<td>128</td>
<td>10(^{-8})</td>
</tr>
<tr>
<td>Ouroboros-R-III</td>
<td>3,953</td>
<td>59</td>
<td>67</td>
<td>2</td>
<td>6</td>
<td>192</td>
<td>10(^{-7})</td>
</tr>
<tr>
<td>Ouroboros-R-IV</td>
<td>5,293</td>
<td>67</td>
<td>79</td>
<td>2</td>
<td>7</td>
<td>256</td>
<td>10(^{-5})</td>
</tr>
<tr>
<td>Ouroboros-R-V</td>
<td>5,618</td>
<td>53</td>
<td>53</td>
<td>4</td>
<td>6</td>
<td>256</td>
<td>10(^{-10})</td>
</tr>
</tbody>
</table>

Parameter sets for Ouroboros-R in rank metric.
Rank Metric Interlude (2/2)

- Best Known Attacks have worse complexity in rank metric ($2^\mathcal{O}(n^2)$) than in Hamming metric ($2^\mathcal{O}(n)$)
- Consequence: worse attacks $\Rightarrow$ better parameters

<table>
<thead>
<tr>
<th>Instance</th>
<th>key size (bits)</th>
<th>$n$</th>
<th>$m$</th>
<th>$q$</th>
<th>$w$</th>
<th>security</th>
<th>decoding failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ouroboros-R-I</td>
<td>1,591</td>
<td>37</td>
<td>43</td>
<td>2</td>
<td>5</td>
<td>100</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Ouroboros-R-II</td>
<td>2,809</td>
<td>53</td>
<td>53</td>
<td>2</td>
<td>5</td>
<td>128</td>
<td>$10^{-8}$</td>
</tr>
<tr>
<td>Ouroboros-R-III</td>
<td>3,953</td>
<td>59</td>
<td>67</td>
<td>2</td>
<td>6</td>
<td>192</td>
<td>$10^{-7}$</td>
</tr>
<tr>
<td>Ouroboros-R-IV</td>
<td>5,293</td>
<td>67</td>
<td>79</td>
<td>2</td>
<td>7</td>
<td>256</td>
<td>$10^{-5}$</td>
</tr>
<tr>
<td>Ouroboros-R-V</td>
<td>5,618</td>
<td>53</td>
<td>53</td>
<td>4</td>
<td>6</td>
<td>256</td>
<td>$10^{-10}$</td>
</tr>
</tbody>
</table>

Parameter sets for Ouroboros-R in rank metric.
Rank Metric Interlude (2/2)

- Best Known Attacks have worse complexity in rank metric \(2^{O(n^2)}\) than in Hamming metric \(2^{O(n)}\)
- Consequence: worse attacks ⇒ better parameters

<table>
<thead>
<tr>
<th>Instance</th>
<th>key size (bits)</th>
<th>(n)</th>
<th>(m)</th>
<th>(q)</th>
<th>(w)</th>
<th>security</th>
<th>decoding failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ouroboros-R-I</td>
<td>1,591</td>
<td>37</td>
<td>43</td>
<td>2</td>
<td>5</td>
<td>100</td>
<td>(10^{-4})</td>
</tr>
<tr>
<td>Ouroboros-R-II</td>
<td>2,809</td>
<td>53</td>
<td>53</td>
<td>2</td>
<td>5</td>
<td>128</td>
<td>(10^{-8})</td>
</tr>
<tr>
<td>Ouroboros-R-III</td>
<td>3,953</td>
<td>59</td>
<td>67</td>
<td>2</td>
<td>6</td>
<td>192</td>
<td>(10^{-7})</td>
</tr>
<tr>
<td>Ouroboros-R-IV</td>
<td>5,293</td>
<td>67</td>
<td>79</td>
<td>2</td>
<td>7</td>
<td>256</td>
<td>(10^{-5})</td>
</tr>
<tr>
<td>Ouroboros-R-V</td>
<td>5,618</td>
<td>53</td>
<td>53</td>
<td>4</td>
<td>6</td>
<td>256</td>
<td>(10^{-10})</td>
</tr>
</tbody>
</table>

Parameter sets for Ouroboros-R in rank metric.
Sketch of proof

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$

$\text{Enc}(\epsilon_0)$ \hspace{1cm} $\text{Enc}_{s^*}(\epsilon_0)$ \hspace{1cm} $\text{Enc}_{s^*,r^*}(\epsilon_0)$

$\text{Enc}(\epsilon_1)$ \hspace{1cm} $\text{Enc}_{s^*}(\epsilon_1)$ \hspace{1cm} $\text{Enc}_{s^*,r^*}(\epsilon_1)$

$\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda) \leq 2 \cdot \left( \text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$

back to security
Sketch of proof

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$

\[
\begin{align*}
\text{Enc}(\epsilon_0) & \quad \rightarrow \quad \text{Enc}_{s^*}(\epsilon_0) & \quad \rightarrow \quad \text{Enc}_{s^*,r^*}(\epsilon_0) \\
\text{Enc}(\epsilon_1) & \quad \rightarrow \quad \text{Enc}_{s^*}(\epsilon_1) & \quad \rightarrow \quad \text{Enc}_{s^*,r^*}(\epsilon_1)
\end{align*}
\]

\[
\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda) \leq 2 \cdot \left( \text{Adv}^{2-\text{DQCSD}}(\lambda) + \text{Adv}^{3-\text{DQCSD}}(\lambda) \right)
\]

back to security
Sketch of proof

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$

$\text{Enc}(\epsilon_0) \xrightarrow{} \text{Enc}_{s^*}(\epsilon_0) \xrightarrow{} \text{Enc}_{s^*,r^*}(\epsilon_0)$

$\text{Enc}(\epsilon_1) \xleftarrow{} \text{Enc}_{s^*}(\epsilon_1) \xleftarrow{} \text{Enc}_{s^*,r^*}(\epsilon_1)$

$$\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda) \leq 2 \cdot \left( \text{Adv}^{2-\text{DQCSD}}(\lambda) + \text{Adv}^{3-\text{DQCSD}}(\lambda) \right)$$
Sketch of proof

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$

$$
\text{Enc}(\epsilon_0) \xrightarrow{} \text{Enc}_{s^*}(\epsilon_0) \xrightarrow{} \text{Enc}_{s^*,r^*}(\epsilon_0) \\
\text{Enc}(\epsilon_1) \xleftarrow{} \text{Enc}_{s^*}(\epsilon_1) \xleftarrow{} \text{Enc}_{s^*,r^*}(\epsilon_1)
$$

$$
\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda) \leq 2 \cdot \left( \text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)
$$

back to security
Sketch of proof

Sequence of games from \( \text{Enc}(\epsilon_0) \) to \( \text{Enc}(\epsilon_1) \)

\[
\begin{align*}
\text{Enc}(\epsilon_0) & \quad \rightarrow \quad \text{Enc}_{s^*}(\epsilon_0) & \quad \rightarrow \quad \text{Enc}_{s^*,r^*}(\epsilon_0) \\
\text{Enc}(\epsilon_1) & \quad \leftarrow \quad \text{Enc}_{s^*}(\epsilon_1) & \quad \leftarrow \quad \text{Enc}_{s^*,r^*}(\epsilon_1)
\end{align*}
\]

\[
\text{Adv}_{\text{ind}}^{\text{E},A}(\lambda) \leq 2 \cdot \left( \text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)
\]

back to security
Sketch of proof

Sequence of games from \(\text{Enc}(\epsilon_0)\) to \(\text{Enc}(\epsilon_1)\)

\[
\text{Enc}(\epsilon_0) \rightarrow \text{Enc}_{s^*}(\epsilon_0) \rightarrow \text{Enc}_{s^*,r^*}(\epsilon_0) \\
\text{Enc}(\epsilon_1) \leftarrow \text{Enc}_{s^*}(\epsilon_1) \leftarrow \text{Enc}_{s^*,r^*}(\epsilon_1)
\]

\[
\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda)\right)
\]

back to security
Sketch of proof

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$

$$
\text{Adv}^{\text{ind}}_{\mathcal{E}, \mathcal{A}}(\lambda) \leq 2 \cdot \left( \text{Adv}^{2-\text{DQCSD}}(\lambda) + \text{Adv}^{3-\text{DQCSD}}(\lambda) \right)
$$

-back to security
Sketch of proof

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$

$\text{Enc}(\epsilon_0) \xrightarrow{\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda)} \text{Enc}_{s^*}(\epsilon_0) \xleftarrow{\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda)} \text{Enc}(\epsilon_1) \xleftarrow{\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda)} \text{Enc}_{s^*}(\epsilon_1) \xrightarrow{\text{Adv}^{\text{ind}}_{\epsilon,A}(\lambda)} \text{Enc}(\epsilon_1) $