Code-based postquantum cryptography : candidates to standardization

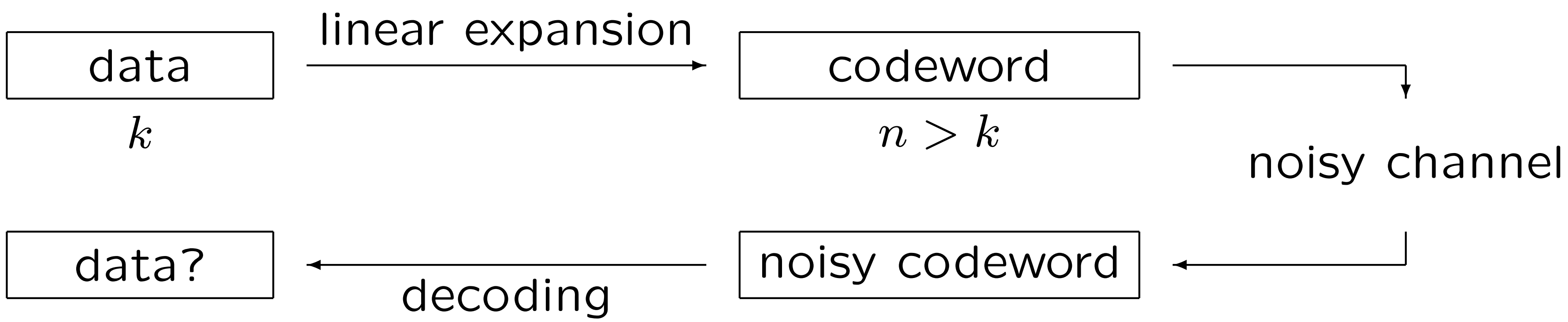
Journées mise en œuvre d'implémentation de cryptographie post-quantique

Rennes, April 23, 2021

Nicolas Sendrier

Prologue

Linear Codes for Telecommunication



[Shannon, 1948] (for a binary symmetric channel of error rate p):

Decoding probability $\longrightarrow 1$ if $\dfrac{k}{n} = R < 1 - h(p)$

($h(p) = -p \log_2 p - (1-p)\log_2(1-p)$ the binary entropy function)

Codes of rate R can correct up to λn errors ($\lambda = h^{-1}(1-R)$)

For instance 11% of errors for $R = 0.5$

Non constructive $\longrightarrow$ no poly-time algorithm for decoding in general

Random Codes Are Hard to Decode

When the linear expansion is random:

- Decoding is NP-complete [Berlekamp, McEliece & van Tilborg, 78]

- Even the tiniest amount of error is (believed to be) hard to remove. Decoding n^{ε} errors is conjectured difficult on average for any $\varepsilon > 0$ [Alekhnovich, 2003].

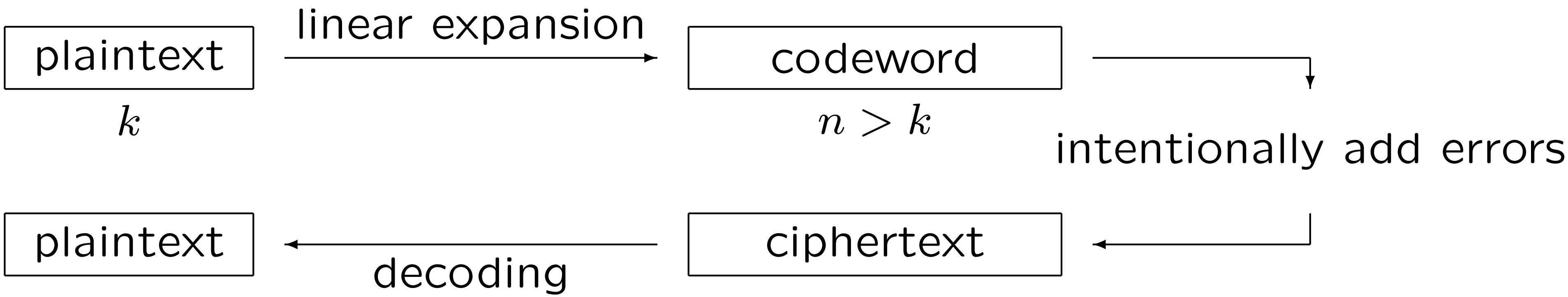
- All known generic decoding algorithm have an exponential complexity *even with access to a quantum computer*

Codes with Good Decoders Exist

Coding theory is about finding "good" codes (i.e. linear expansions)

- alternant codes have a poly-time decoder for $\Theta\left(\frac{n}{\log n}\right)$ errors
- some classes of codes have a poly-time decoder for $\Theta(n)$ errors (algebraic geometry, expander graphs, concatenation, ...)

Linear Codes for Cryptography



- If a random linear code is used, no one can decode efficiently
- If a "good" code is used, anyone who knows the structure has access to a fast decoder

Assuming that the knowledge of the linear expansion does not reveal the code structure:

- The linear expansion is public and anyone can encrypt
- The decoder is known to the legitimate user who can decrypt
- For anyone else, the code looks random

Postquantum Cryptography

Need for Postquantum Cryptographic Primitives

Most of the public-key cryptography deployed today is vulnerable to quantum computer (Shor, Grover, . . .)

For long term security, new cryptographic solutions are required for public-key encryption, key exchange mechanisms, and digital signatures

Scientific communities, governmental institutions, standardization bodies throughout the world are aware of this

$\rightarrow$ NIST call for postquantum primitives

Postquantum Standardization

NIST call for postquantum primitives started in 2018

- Digital Signature
- Public-Key Encryption/Key Exchange

Three code-based candidates in NIST's 3rd round (all Encryption/Key Exchange):

- one finalist, Classic McEliece
- two alternate candidates, BIKE and HQC

Code-Based Cryptography

McEliece Public-key Encryption Scheme – Overview

Let $\mathcal{F}$ be a family of t-error correcting q-ary linear $[n, k]$ codes
e.g. irreducible binary Goppa codes [McEliece, 1978]

Key generation:

pick $\mathcal{C} \in \mathcal{F} \rightarrow \begin{cases} \textbf{Public Key: } G \in \mathbf{F}_q^{k\times n}, \text{ a generator matrix of } \mathcal{C} \\ \textbf{Secret Key: } \Phi : \mathbf{F}_q^n \rightarrow \mathcal{C}, \text{ a } t\text{-bounded decoder} \end{cases}$

Encryption: $\left[\begin{array}{llll} E_G : & \mathbf{F}_q^k & \rightarrow & \mathbf{F}_q^n \\ & x & \mapsto & xG + e \end{array}\right]$ with e random of weight t

Decryption: $\left[\begin{array}{llll} D_\Phi : & \mathbf{F}_q^n & \rightarrow & \mathbf{F}_q^k \cup \{\perp\} \\ & xG + e & \mapsto & x \end{array}\right]$ derive x from $\Phi(xG + e) = xG$

$G \in \mathbf{F}_q^{k\times n}$ a generator matrix: $\mathcal{C} = \left\{ xG \mid x \in \mathbf{F}_q^k \right\}$

Φ is t-bounded: $\forall (c, e) \in \mathcal{C} \times \mathbf{F}_q^n$, $|e| \leq t \Rightarrow \Phi(c + e) = c$

Niederreiter Public-key Encryption Scheme – Overview

Let $\mathcal{F}$ be a family of t-error correcting q-ary linear $[n,k]$ codes [Niederreiter, 1986]

Key generation: pick $\mathcal{C} \in \mathcal{F}$

$\rightarrow \begin{cases} \textbf{Public Key: } H \in \mathbf{F}_q^{(n-k)\times n}, \text{ a parity check matrix of } \mathcal{C} \\ \textbf{Secret Key: } \Psi : \mathbf{F}_q^r \rightarrow \mathbf{F}_q^n, \text{ a } t\text{-bounded } H\text{-syndrome decoder} \end{cases}$

Encryption: $\left[\begin{array}{rccc} E_H : & \mathcal{S}_n(\mathbf{0},t) & \rightarrow & \mathbf{F}_q^{n-k} \\ & e & \mapsto & eH^T \end{array} \right]$

Decryption: $\left[\begin{array}{rccc} D_\Psi : & \mathbf{F}_q^{n-k} & \rightarrow & \mathcal{S}_n(\mathbf{0},t) \cup \{\perp\} \\ & eH^T & \mapsto & e = \Psi(eH^T) \end{array} \right]$

$H \in \mathbf{F}_q^{(n-k)\times n}$ a parity check matrix: $\mathcal{C} = \left\{ c \in \mathbf{F}_q^n \mid cH^T = 0 \right\}$

Ψ is t-bounded: $\forall e \in \mathbf{F}_q^n$, $|e| \leq t \Rightarrow \Psi(eH^T) = e$

Instances of the McEliece/Niederreiter Scheme

Irreducible Binary Goppa Codes

System parameters:

- $m > 0$ an integer $\rightarrow$ extension field $\mathbf{F}_{2^m}$
- $n \leq 2^m$ the code length
- $0 < t < n/m$ the error correcting capability
- $k = n - tm$ the code dimension as a subspace of $\mathbf{F}_2^n$

Goppa code:

- $g(x) \in \mathbf{F}_{2^m}[x]$ monic, irreducible, of degree t
- $L = (\alpha_1, \ldots, \alpha_n)$ distinct elements of $\mathbf{F}_{2^m}$

$$\Gamma(L, g) = \left\{ a \in \mathbf{F}_2^n \mid a\tilde{H}^T = 0 \right\}, \tilde{H} = \begin{pmatrix} \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_n)} \\ \frac{\alpha_1}{g(\alpha_1)} & \cdots & \frac{\alpha_n}{g(\alpha_n)} \\ \vdots & & \vdots \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix}$$

Irreducible Binary Goppa Codes

Key generation:

- build a binary parity check matrix $\widehat{H} \in \mathbf{F}_2^{tm \times n}$ from $\tilde{H}$
  (each $\alpha_j^i / g(\alpha_j) \in \mathbf{F}_{2^m}$ in $\tilde{H}$ becomes a column vector in $\mathbf{F}_2^m$)
- Compute its systematic form $H = (I_{n-k} \mid T) = S\widehat{H}$
- **Private key:** $(g, \alpha_1, \ldots, \alpha_n) \in \mathbf{F}_{2^m}[x] \times \mathbf{F}_{2^m}^n$
- **Public key:** $T \in \mathbf{F}_2^{(n-k) \times k}$

Decoding: in the polynomial ring $\mathbf{F}_{2^m}[x]$

- Compute a syndrome $S(z) = \sum_{i=0}^{2t-1} s_i z^i$ with $s_i = \sum_{j=1}^{n-k} \frac{c_j \alpha_j^i}{g(\alpha_j)^2}$
- Solve the equation $S(z)\sigma(z) = \omega(z) \mod z^{2t}$ with $\begin{cases} \deg \sigma \leq t \\ \deg \omega < t \end{cases}$
- Find the roots of $\sigma(z)$, the error $e = (e_1, \ldots, e_n) \in \mathbf{F}_2^n$ verifies

$$e_j \neq 0 \Leftrightarrow \sigma(\alpha_j^{-1}) = 0$$

Irreducible Binary Goppa Codes

	ciphertext size in bits			
m, n, k, t	McEliece	Niederreiter	key size	security
$10, 1024, 524, 50$	1024	500	32 kB	52
$12, 4096, 3424, 56$	4096	672	288 kB	128
$13, 8192, 6528, 128$	8192	1664	1358 kB	256

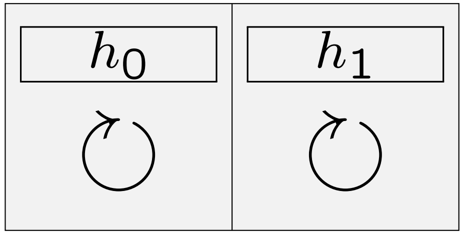
Security assumptions:

- Pseudorandomness of Goppa codes
  (the public key T is computationally indistinguishable from a random uniform binary matrix of same size)
- Hardness of decoding
  (decoding t errors in a random binary linear $[n, k]$ code is intractable)

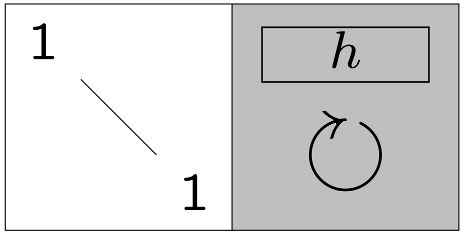
$\rightarrow$ Classic McEliece NIST proposal

QC-MDPC Codes

Quasi-Cyclic Moderate Density Parity Check codes

$H_{\text{secret}} =$ 

$h_0, h_1 \in \mathcal{R} = \mathbf{F}_2[x]/(x^r - 1)$ sparse

$H_{\text{public}} =$ 

$h = h_0^{-1} h_1 \in \mathcal{R}$ dense

binary circulant $r \times r$ matrices are isomorphic to $\mathcal{R} = \mathbf{F}_2[x]/(x^r - 1)$

The sparse parity check matrix H_{secret} allows decoding

The dense parity check matrix H_{public} is indistinguishable from random

QC-MDPC Codes

Quasi-Cyclic Moderate Density Parity Check codes

$$H_{\text{secret}} = \left[\begin{array}{c|c} \boxed{h_0} & \boxed{h_1} \end{array}\right], \quad H_{\text{public}} = \left[\begin{array}{ccc|c} 1 & & & \boxed{h} \\ & \ddots & & \\ & & 1 & \end{array}\right]$$

System parameters:

- r the block size, $n = 2r$ the code length
- w the row weight, $w \approx \sqrt{n}$
- t the error weight, $t \approx \sqrt{n}$

efficient decoding possible as long as $w \cdot t \lessapprox n$

Key generation:

- **Private key:** $(h_0, h_1) \in \mathcal{R}^2$, $|h_0| = |h_1| = w/2$
- **Public key:** $h = h_0^{-1} h_1 \in \mathcal{R}$

QC-MDPC Codes

Bit Flipping Decoding:

Input: $s \in \mathbf{F}_2^r$, $H \in \mathbf{F}_2^{r \times n}$ $\quad \triangleright$ H_j the j-th column of H

$e \leftarrow 0^n$

repeat

$\quad s' \leftarrow s - eH^T$

$\quad T \leftarrow \texttt{threshold}(\mathit{context})$

$\quad$ **for** $j = 1, \ldots, n$ **do**

$\quad\quad$ **if** $\left| s' \cap H_j \right| \geq T$ **then** $\quad \triangleright$ # unsatisfied equations involving j

$\quad\quad\quad e_j \leftarrow e_j + 1$

until $s = eH^T$

return e

QC-MDPC Codes

	size in bits		
r, w, t	block	key	security
$12\,323, 142, 134$	$12\,323$	$12\,323$	128
$24\,659, 206, 199$	$24\,659$	$24\,659$	192
$40\,973, 274, 264$	$40\,973$	$40\,973$	256

Security assumptions:

- Hardness of quasi-cyclic codeword finding
  (the public key h is computationally indistinguishable from a random uniform element of $\mathcal{R}$)
- Hardness of quasi-cyclic decoding
  (decoding t errors in a random binary quasi-cyclic $[n, r]$ code is intractable)

$\rightarrow$ BIKE NIST proposal

The Third Round Code-Based NIST Candidates

The Third Round Code-Based NIST Candidates

- Classic McEliece

  An instance of Niederreiter's scheme using Goppa codes

- BIKE

  An instance of Niederreiter's scheme using QC-MDPC codes

- HQC

  Derives from [Alekhnovich, 2003] rather than [McEliece, 78]

  No trapdoor decoder, the secret is a sparse vector

Classic McEliece KEM

Setup: parameters m, n, t, $k = n - mt$, ℓ, hash function H with output in $\{0,1\}^\ell$

KeyGen Output: sk, pk

$g \xleftarrow{\$}$ monic irreducible polynomials of degree t

$(\alpha_1, \ldots, \alpha_n) \xleftarrow{\$}$ distinct elements of $\mathbf{F}_{2^m}$

$\tilde{H} \leftarrow \left(\alpha_j^i / g(\alpha_j)\right)_{0 \le i < t, 1 \le j \le n}$ $\qquad \triangleright \in \mathbf{F}_{2^m}^{t \times n}$

$\hat{H} \leftarrow \mathsf{expand}(\tilde{H})$ $\qquad \triangleright \in \mathbf{F}_2^{tm \times n}$

$H = ((I_{n-k} \mid T) \leftarrow \mathsf{GaussElim}(\hat{H})$ $\qquad \triangleright$ if fail, restart from top

$s \xleftarrow{\$} \{0,1\}^\ell$

$\mathsf{sk} = ((g, \alpha_1, \ldots, \alpha_n), s)$ $\qquad \triangleright$ we denote $\Gamma = (g, \alpha_1, \ldots, \alpha_n)$

$\mathsf{pk} = T \in \mathbf{F}_2^{(n-k) \times k}$ $\qquad \triangleright$ we denote $H = (I_{n-k} \mid T)$

Encaps Input: pk

Output: $c = (c_0, c_1) \in \mathbf{F}_2^{n-k} \times \{0,1\}^\ell$, $K \in \{0,1\}^\ell$

$e \xleftarrow{\$} \{e \in \mathbf{F}_2^n \mid |e| = t\}$

$c = (c_0, c_1) \leftarrow (eH^T, \mathsf{H}(2, e))$

$K \leftarrow \mathsf{H}(1, e, c)$

Classic McEliece KEM

Decaps Input: sk, $c = (c_0, c_1)$

Output: $K \in \{0,1\}^\ell$

$e \leftarrow \mathsf{GoppaDecode}(c_0, \Gamma)$

if $e = \perp$ **or** $\mathsf{H}(2, e) \neq c_1$ **then** $K \leftarrow \mathsf{H}(0, s, c)$ **else** $K \leftarrow \mathsf{H}(1, e, c)$

GoppaDecode:

- Compute an algebraic syndrome $(c_0, \Gamma) \rightarrow S(z)$
- Solve the key equation $S(z) \rightarrow \sigma(z)$
- Find the roots of $\sigma(z) \rightarrow$ error locations

BIKE

Setup: parameters r, w, t, ℓ, hash functions $\mathbf{K}$, $\mathbf{L}$ with output in $\{0,1\}^\ell$ and $\mathbf{H}$ with output in $\{e = (e_0, e_1) \in \mathcal{R}^2 \mid |e_0| + |e_1| = t\}$

KeyGen Output: sk, pk

$$
\begin{aligned}
&(h_0, h_1) \xleftarrow{\$} \{(h_0, h_1) \in \mathcal{R}^2 \mid |h_0| = |h_1| = w/2\} \\
&h \leftarrow h_1 h_0^{-1} \\
&\sigma \xleftarrow{\$} \{0,1\}^\ell \\
&\mathsf{sk} = ((h_0, h_1), \sigma) \\
&\mathsf{pk} = h
\end{aligned}
$$

Encaps Input: pk

Output: $c = (c_0, c_1) \in \mathcal{R} \times \{0,1\}^\ell$, $K \in \{0,1\}^\ell$

$$
\begin{aligned}
&m \xleftarrow{\$} \{0,1\}^\ell \\
&(e_0, e_1) \leftarrow \mathbf{H}(m) \\
&c \leftarrow (e_0 + e_1 h, m \oplus \mathbf{L}(e_0, e_1)) \\
&K \leftarrow \mathbf{K}(m, c)
\end{aligned}
$$

BIKE

Decaps Input: sk, $c = (c_0, c_1)$

Output: $K \in \{0,1\}^\ell$

$e \leftarrow \text{decoder}(c_0 h_0, h_0, h_1)$

$m \leftarrow c_1 \oplus \mathbf{L}(e)$

if $e = \mathbf{H}(m)$ **then** $K \leftarrow \mathbf{K}(m, c)$ **else** $K \leftarrow \mathbf{K}(\sigma, c)$

decoder() is any variant of bit flipping decoding. It is prone to decoding failure. The decoding failure rate (DFR) is defined as

$$\text{DFR(decoder)} = \Pr[(e_0, e_1) \neq \text{decoder}(e_0 h_0 + e_1 h_1, h_0, h_1)]$$

(probability over all errors (e_0, e_1) and all keys (h_0, h_1))

HQC KEM

Let $\mathcal{R} = \mathbf{F}_2[X]/(X^n - 1)$, let $\mathcal{E}_w = \{z \in \mathcal{R} \mid |z| = w\}$

Setup: parameters $n, w, w_e, w_r, k, \delta$, hash function $\mathbf{K}$ with output in $\{0,1\}^k$ and $\mathbf{H}$ with output in $\mathcal{E}_{w_e} \times \mathcal{E}_{w_r}^2$, G the generator matrix of a δ-error correcting code

KeyGen Output: sk, pk

$h \stackrel{\$}{\leftarrow} \mathcal{R}$
$(x, y) \stackrel{\$}{\leftarrow} \mathcal{E}_w^2$
$s \leftarrow x + hy$
$\mathsf{sk} = (x, y)$
$\mathsf{pk} = (h, s)$

Encaps Input: pk

Output: $(u, v) \in \mathcal{R}^2$, $K \in \{0,1\}^k$
$m \stackrel{\$}{\leftarrow} \{0,1\}^k$
$(e, r_1, r_2) \leftarrow \mathbf{H}(m)$ $\triangleright$ $|e| = w_e, |r_1| = |r_2| = w_r$, sparse
$(u, v) \leftarrow (r_1 + hr_2, mG + sr_2 + e)$
$K \leftarrow \mathbf{K}(m, (u, v))$

HQC KEM

Decaps Input: sk, $(u, v) \in \mathcal{R}^2$

Output: $K \in \{0, 1\}^k$

$m \leftarrow \text{decode}(v - uy)$

$(e, r_1, r_2) \leftarrow \mathbf{H}(m)$

if $(u, v) \neq (r_1 + hr_2, mG + sr_2 + e)$ **then** abort

else $K \leftarrow \mathbf{K}(m, (u, v))$

decode() is a decoder for the code $\mathcal{C}$ spanned by G. This code is part of the system setup, it is public as well as its decoding procedure. It's failure rate however is relevant for the security analysis.

Security

Ephemeral Keys *versus* Static Keys

Alice		**Bob**
samples sk, pk	$\xrightarrow{\mathsf{pk}}$	
$m \leftarrow \mathbf{Dec}_{\mathsf{sk}}(c)$	$\xleftarrow{c = \mathbf{Enc}_{\mathsf{pk}}(m)}$	samples m

Shared key: $K = \mathsf{Hash}(m)$

Ephemeral Keys: the key pair $(\mathsf{sk}, \mathsf{pk})$ is used only once

- allows forward secrecy
- decryption failure doesn't impact security (IND-CPA is enough)
- only synchronous protocols (*e.g.* TLS)

Static Keys: the key pair $(\mathsf{sk}, \mathsf{pk})$ is used multiple times

- reduces communication cost
- decryption failure must be negligible (IND-CCA is required)
- allows asynchronous protocols (*e.g.* email)

Security Models

IND-CPA

Indistinguishability under chosen plaintext attack

Guaranteed by computational assumptions alone

Enough for ephemeral keys

IND-CCA

Indistinguishability under adaptive chosen ciphertext attack

Requires negligible decryption failure

Relevant (only?) for static keys

Security Assumptions

	IND-CPA	IND-CCA
Classic McEliece	• Pseudorandomness of Goppa codes • Hardness of decoding	• Pseudorandomness of Goppa codes • Hardness of decoding
BIKE	• Hardness of QC decoding • Hardness of QC codeword finding	• Hardness of QC decoding • Hardness of QC codeword finding • Negligible decoding failure (for QC-MDPC codes)
HQC	• Hardness of QC decoding	• Hardness of QC decoding • Negligible decoding failure (for any code)

Complexity

Space Complexity (IND-CCA Security)

	pk size	Block size	Sec. level
Classic McEliece	261 KB	128 bytes	1
	525 KB	188 bytes	3
	1.3 MB	226 bytes	5
BIKE	1 541 bytes	1 573 bytes	1
	3 083 bytes	3 115 bytes	3
	5 122 bytes	5 154 bytes	5
HQC	3 125 bytes	6 234 bytes	1
	5 884 bytes	11 752 bytes	3
	8 897 bytes	17 778 bytes	5

Time Complexity

Software:

- BIKE and HQC are comparable, with an advantage to BIKE (ranges from a few 100k to a few mega cycles)
- Classic McEliece:
  - key generation is ridiculously slow in software (several 100 mega cycles)
  - encaps/decaps are very fast (50k to a few 100k cycles)

Fair comparison is difficult, but third party implementation are appearing and things might clarify in the coming years

Secure Implementation

Secure Implementations

All remaining code-based NIST candidates feature constant-time implementation by design:

- specifications allow constant-time implementation
- constant-time optimized software implementation are available (for some parameter sets)

KeyGen

Output: sk, pk

$g \xleftarrow{\$}$ monic irreducible polynomials of degree t

$(\alpha_1, \ldots, \alpha_n) \xleftarrow{\$}$ distinct elements of $\mathbf{F}_{2^m}$

$\tilde{H} \leftarrow \left(\alpha_j^i / g(\alpha_j)\right)_{0 \le i < t, 1 \le j \le n}$ $\quad \triangleright \in \mathbf{F}_{2^m}^{t \times n}$

$\hat{H} \leftarrow \mathsf{expand}(\tilde{H})$ $\quad \triangleright \in \mathbf{F}_2^{tm \times n}$

$H = ((I_{n-k} \mid T) \leftarrow \mathsf{GaussElim}(\hat{H})$ $\quad \triangleright$ if fail, restart from top

$s \xleftarrow{\$} \{0, 1\}^\ell$

$\mathsf{sk} = ((g, \alpha_1, \ldots, \alpha_n), s)$

$\mathsf{pk} = T \in \mathbf{F}_2^{(n-k) \times k}$

Key operations:

- Arithmetic in the extension field $\mathbf{F}_{2^m}$
- Gaussian elimination over a binary matrix is the bottleneck
  > 3 failures on average $\rightarrow$ "Semi-systematic" form could avoid that, implies an evolution of the specification

Classic McEliece – Encaps

Encaps

Input: pk

Output: $c = (c_0, c_1) \in \mathbf{F}_2^{n-k} \times \{0,1\}^\ell$, $K \in \{0,1\}^\ell$

$e \stackrel{\$}{\leftarrow} \{e \in \mathbf{F}_2^n \mid |e| = t\}$

$c = (c_0, c_1) \leftarrow (eH^T, \mathsf{H}(2, e))$

$K \leftarrow \mathsf{H}(1, e, c)$

Key operations:

- Binary linear algebra

Decaps

Input: sk, $c = (c_0, c_1)$
Output: $K \in \{0,1\}^\ell$
$e \leftarrow \mathsf{GoppaDecode}(c_0, \Gamma)$
if $e = \perp$ **or** $\mathsf{H}(2, e) \neq c_1$ **then** $K \leftarrow \mathsf{H}(0, s, c)$ **else** $K \leftarrow \mathsf{H}(1, e, c)$

GoppaDecode:

1. Compute an algebraic syndrome $(c_0, \Gamma) \to S(z)$
2. Solve the key equation $S(z) \to \sigma(z)$
3. Find the roots of $\sigma(z) \to$ error locations

Key operations:

- Syndrome computation and root finding use an ad-hoc FFT
- Key equation is solved by the Berlekamp-Massey algorithm
- Permutation is implemented through a Beneš network

BIKE – KeyGen

KeyGen

Output: sk, pk

$(h_0, h_1) \stackrel{\$}{\leftarrow} \{(h_0, h_1) \in \mathcal{R}^2 \mid |h_0| = |h_1| = w/2\}$

$h \leftarrow h_1 h_0^{-1}$

$\sigma \stackrel{\$}{\leftarrow} \{0, 1\}^\ell$

$\mathsf{sk} = ((h_0, h_1), \sigma)$

$\mathsf{pk} = h$

Key operations:

- Arithmetic in $\mathcal{R} = \mathbf{F}_2[x]/(x^r - 1)$
  bottleneck is the inversion
- Sampling constant weight words

Encaps

Input: pk
Output: $c = (c_0, c_1) \in \mathcal{R} \times \{0,1\}^\ell$, $K \in \{0,1\}^\ell$
$m \xleftarrow{\$} \{0,1\}^\ell$
$(e_0, e_1) \leftarrow \mathbf{H}(m)$
$c \leftarrow (e_0 + e_1 h, m \oplus \mathbf{L}(e_0, e_1))$
$K \leftarrow \mathbf{K}(m, c)$

Key operations:

- Arithmetic in $\mathcal{R} = \mathbf{F}_2[x]/(x^r - 1)$
- sampling constant weight words (hash function $\mathbf{H}$)

BIKE – Decaps

Decaps

Input: sk, $c = (c_0, c_1)$
Output: $K \in \{0,1\}^\ell$
$e \leftarrow \text{decoder}(c_0 h_0, h_0, h_1)$
$m \leftarrow c_1 \oplus \mathbf{L}(e)$
if $e = \mathbf{H}(m)$ **then** $K \leftarrow \mathbf{K}(m, c)$ **else** $K \leftarrow \mathbf{K}(\sigma, c)$

Key operations:

- Arithmetic in $\mathcal{R} = \mathbf{F}_2[x]/(x^r - 1)$
- Sampling constant weight words (hash function $\mathbf{H}$)
- Bit flipping decoding

BIKE – Bit Flipping

Bit Flipping Decoding

Input: $s \in \mathbf{F}_2^r$, $H \in \mathbf{F}_2^{r \times n}$

1: $e \leftarrow 0^n$
2: **repeat** a fixed number of times
3: $\quad s' \leftarrow s - eH^T$
4: $\quad T \leftarrow \texttt{threshold}(\mathit{context})$
5: $\quad$ **for** $j = 1, \ldots, n$ **do**
6: $\qquad$ **if** $\left| s' \cap H_j \right| \geq T$ **then**
7: $\qquad\quad e_j \leftarrow e_j + 1$
8: **until**
9: **return** e

The actual algorithm is different but key operation are the same:

- Syndrome update, instruction 3:
- Counters computation, instruction 6:
  in practice all counters $\left| s' \cap H_j \right|$ are computed at once

KeyGen

Output: sk, pk

$h \stackrel{\$}{\leftarrow} \mathcal{R}$

$(x, y) \stackrel{\$}{\leftarrow} \mathcal{E}_w^2$

$s \leftarrow x + hy$

$\mathsf{sk} = (x, y)$

$\mathsf{pk} = (h, s)$

Key operations:

- Arithmetic in $\mathcal{R} = \mathbf{F}_2[x]/(x^n - 1)$
- Sampling constant weight words

Encaps

Input: pk

Output: $(u, v) \in \mathcal{R}^2$, $K \in \{0,1\}^k$

$m \xleftarrow{\$} \{0,1\}^k$

$(e, r_1, r_2) \leftarrow \mathbf{H}(m)$

$(u, v) \leftarrow (r_1 + hr_2, mG + sr_2 + e)$

$K \leftarrow \mathbf{K}(m, (u, v))$

Key operations:

- Arithmetic in $\mathcal{R} = \mathbf{F}_2[x]/(x^n - 1)$
- (Linear algebra over $\mathbf{F}_2$)
- Sampling constant weight words

HQC KEM – Decaps

Decaps Input: sk, $(u, v) \in \mathcal{R}^2$

Output: $K \in \{0, 1\}^k$

$m \leftarrow \text{decode}(v - uy)$

$(e, r_1, r_2) \leftarrow \mathbf{H}(m)$

if $(u, v) \neq (r_1 + hr_2, mG + sr_2 + e)$ **then** abort

else $K \leftarrow \mathbf{K}(m, (u, v))$

Key operations:

- Arithmetic in $\mathcal{R} = \mathbf{F}_2[x]/(x^n - 1)$
- (Linear algebra over $\mathbf{F}_2$)
- Sampling constant weight words
- decoding in the code $\mathcal{C}$ spanned by G

Conclusion

Code-based NIST candidates enjoy some nice features

- Specifications are simple
- Implementation are efficient
- Classic McEliece is well suited to static key
- BIKE and HQC are well suited to ephemeral key

Thank you for your attention