The Roadmap from Polynomials to Quantum-safe Cryptosystems

A perspective from discrete mathematics
Part 4/4: Code-based Cryptography and HQC

Chunlei Li (University of Bergen, Norway)

INCP2-2024/10213 on
Mathematical Theory of Data Transmission and Data Encryption

Oct. 6-10, 2025, Tromsø
Fall school on Geometry in Cryptography and Communication

Public-Key Cryptography

Symmetric vs Asymmetric Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Birth of Public-Key Cryptosystems

- 1970: first known (secret) report on public-key cryptography by CESG, UK
- 1976: Diffie and Hellman public introduction to conceptual public-key cryptography
  - Avoid reliance on third-parties for key distribution
  - Allow digital signatures
- 1977: RSA Cryptosystem
- ....

Public and Private Keys

Public Key (PB)

- Public, Available to anyone
- For secrecy: used in encryption
- For authentication: used in decryption

Private Key (PR)

- Secret, known only by owner
- For secrecy: used in decryption
- For authentication: used in encryption

Confidentiality with Public Key Crypto

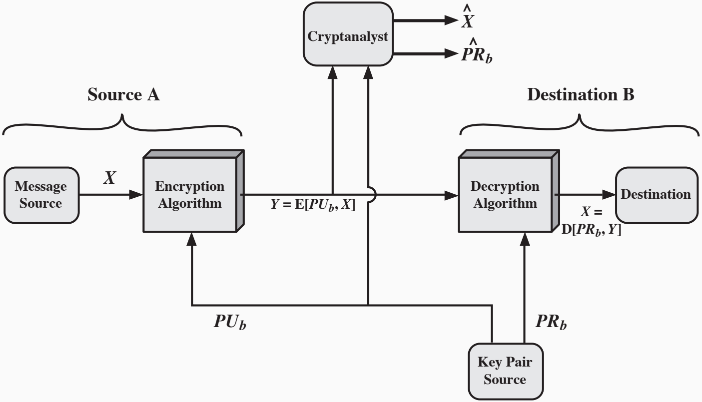


Figure 9.2 Public-Key Cryptosystem: Secrecy

Authentication with Public Key Crypto

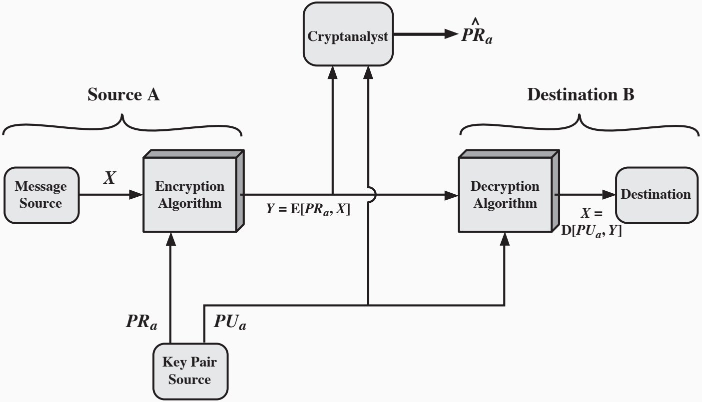


Figure 9.3 Public-Key Cryptosystem: Authentication

Applications of Public Key Cryptosystems

- Secrecy, encryption/decryption of data (messages, keys,..)
- Digital signature, *sign* message with private key
- Key exchange, share secret session keys

Catching up on NIST PQC project

NIST initiated the Post-Quantum Cryptography (PQC) Standardization Process in 2016.

- Selecting quantum-resistant public-key cryptographic algorithms

  Dilithium (**module lattices**)
  Falcon (**NTRU lattices**)
  SPHINCS+ (**hash-based**)

  have been standardized for signatures.
- Kyber (**module lattices**) was the only KEM standardized.
- Need for more diversity of computational hardness assumptions to reduce the risk of a single cryptanalytic breakthrough.

  Even more since `https://eprint.iacr.org/2024/555`.

Catching up on NIST PQC project

NIST initiated the Post-Quantum Cryptography (PQC) Standardization Process in 2016.

- Selecting quantum-resistant public-key cryptographic algorithms

  Dilithium (**module lattices**)
  Falcon (**NTRU lattices**)
  SPHINCS+ (**hash-based**)

  have been standardized for signatures.
- Kyber (**module lattices**) was the only KEM standardized.
- Need for more diversity of computational hardness assumptions to reduce the risk of a single cryptanalytic breakthrough.

  Even more since `https://eprint.iacr.org/2024/555`.

Catching up on NIST PQC project

NIST initiated the Post-Quantum Cryptography (PQC) Standardization Process in 2016.

- Selecting quantum-resistant public-key cryptographic algorithms

  Dilithium (**module lattices**)
  Falcon (**NTRU lattices**)
  SPHINCS+ (**hash-based**)

  have been standardized for signatures.
- Kyber (**module lattices**) was the only KEM standardized.
- Need for more diversity of computational hardness assumptions to reduce the risk of a single cryptanalytic breakthrough.
  Even more since `https://eprint.iacr.org/2024/555`.

HQC

NIST advanced four more algorithms as KEM candidates to the fourth round.

Code-based	Isogeny-based
BIKE	SIKE[1]
HQC	
Classic McEliece	

HQC is an IND-CCA2 KEM selected for standardization.

[1] `https://issikebrokenyet.github.io/`

HQC

NIST advanced four more algorithms as KEM candidates to the fourth round.

Code-based	Isogeny-based
BIKE	SIKE[1]
HQC	
Classic McEliece	

HQC is an IND-CCA2 KEM selected for standardization.

[1] `https://issikebrokenyet.github.io/`

HQC is based on the hardness of (variants of) the Searching Syndrome Decoding problem (**SSD**) and the Decisional Syndrome Decoding problem (**DSD**).

Searching Syndrome Decoding problem

Let n, k be positive integers. Given $H, \mathbf{y} \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$, find $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathbf{y} = \mathbf{x}H^\top$ and $wt(\mathbf{x}) = w$.

- This problem has been proven to be NP-complete.

HQC is based on the hardness of (variants of) the Searching Syndrome Decoding problem (**SSD**) and the Decisional Syndrome Decoding problem (**DSD**).

Searching Syndrome Decoding problem

Let n, k be positive integers. Given $H, \mathbf{y} \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$, find $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathbf{y} = \mathbf{x}H^\top$ and $wt(\mathbf{x}) = w$.

- This problem has been proven to be NP-complete.

Difficult problems for code-based cryptography

HQC is based on the hardness of (variants of) the Searching Syndrome Decoding problem (**SSD**) and the Decisional Syndrome Decoding problem (**DSD**).

Searching Syndrome Decoding problem

Let n, k be positive integers. Given $H, \mathbf{y} \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$, find $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathbf{y} = \mathbf{x}H^\top$ and $wt(\mathbf{x}) = w$.

- This problem has been proven to be NP-complete.

Difficult problems for code-based cryptography

Decisional Syndrome Decoding problem

Let n, k be positive integers. Given $(H, \mathbf{y}) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$, decide with non-negligible advantage whether $(H, \mathbf{y})$ came from the **SD** distribution or the uniform distribution.

- The advantage of the attacker is measured as $Adv(\mathtt{A}) = 2 \cdot P(success) - 1$.
- The **DSD** problem helps to achieve IND-CCA2 security.

Hamming Quasi-Cyclic

- Quasi-Cyclic codes have their equivalent problems: the s-**QCSD** and s-**DQCSD** problems.
- Specific to HQC are the 3-**QCSD-PT** problem and its Decisional variant.

For many years the decoder was part of the private key for code-based cryptosystems. It was usually masked into a random public code like for McEliece.

HQC brings significant changes! The decoder is public and the trapdoor becomes the mask put on the plaintext. The decoder works only if able to remove the mask.

Hamming Quasi-Cyclic

- Quasi-Cyclic codes have their equivalent problems: the s-**QCSD** and s-**DQCSD** problems.
- Specific to HQC are the 3-**QCSD-PT** problem and its Decisional variant.

For many years the decoder was part of the private key for code-based cryptosystems. It was usually masked into a random public code like for McEliece.

HQC brings significant changes! The decoder is public and the trapdoor becomes the mask put on the plaintext. The decoder works only if able to remove the mask.

Hamming Quasi-Cyclic

- Quasi-Cyclic codes have their equivalent problems: the s-**QCSD** and s-**DQCSD** problems.
- Specific to HQC are the 3-**QCSD-PT** problem and its Decisional variant.

For many years the decoder was part of the private key for code-based cryptosystems. It was usually masked into a random public code like for McEliece.

HQC brings significant changes! The decoder is public and the trapdoor becomes the mask put on the plaintext. The decoder works only if able to remove the mask.

Hamming Quasi-Cyclic

There are two codes:

1. One public code to create the ciphertext.
   This code doesn't need to remove errors so we can focus on security.
   Chosen Quasi-Cyclic for its efficiency and the compactness of ciphertexts.
2. One public code to remove the errors.
   This code doesn't need to be secure so we can focus on maximum error correction.
   A concatenation of a Reed-Solomon and a (duplicated) Reed-Muller code.

Hamming Quasi-Cyclic

There are two codes:

1. One public code to create the ciphertext.
   This code doesn't need to remove errors so we can focus on security.
   Chosen Quasi-Cyclic for its efficiency and the compactness of ciphertexts.
2. One public code to remove the errors.
   This code doesn't need to be secure so we can focus on maximum error correction.
   A concatenation of a Reed-Solomon and a (duplicated) Reed-Muller code.

Hamming Quasi-Cyclic

Drop the (x) for polynomials so $\mathbf{g}(x)$ becomes $\mathbf{g}$. Public, private and one-time random data.

KeyGen

$\mathbf{G}$ a generator matrix for a public code $\mathcal{C}_{pub}$.

Random $\mathbf{h} \in \mathcal{R}$ where $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$.

Random $\mathbf{x}, \mathbf{y} \in \mathcal{R} \times \mathcal{R}$ such that $wt(\mathbf{x}) = wt(\mathbf{y}) = w$.

The syndrome $\mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y}$

Private key: $(\mathbf{x}, \mathbf{y})$

Public key: $(\mathbf{h}, \mathbf{s})$

Hamming Quasi-Cyclic

Drop the (x) for polynomials so $\mathbf{g}(x)$ becomes $\mathbf{g}$. Public, private and one-time random data.

KeyGen

$\mathbf{G}$ a generator matrix for a public code $\mathcal{C}_{pub}$.

Random $\mathbf{h} \in \mathcal{R}$ where $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$.

Random $\mathbf{x}, \mathbf{y} \in \mathcal{R} \times \mathcal{R}$ such that $wt(\mathbf{x}) = wt(\mathbf{y}) = w$.

The syndrome $\mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y}$

Private key: $(\mathbf{x}, \mathbf{y})$

Public key: $(\mathbf{h}, \mathbf{s})$

McEliece

KeyGen

G: generator matrix of a binary Goppa code $[n, k, 2t+1]$.

S: a non-singular $k \times k$ matrix.

P: an $n \times n$ permutation matrix.

Private key: (G, S, P)
Public key: (G', t) such that $G' = SGP$

Encryption

Let a message $\mathbf{m} \in \mathcal{R}$.

Random $\mathbf{e} \in \mathcal{R}$ such that $wt(\mathbf{e}) = w_e$.

Random $(\mathbf{r_1}, \mathbf{r_2}) \in \mathcal{R} \times \mathcal{R}$ such that $wt(\mathbf{r_1}) = wt(\mathbf{r_2}) = w_r$.

$$\mathbf{u} = \mathbf{r_1} + \mathbf{h}\mathbf{r_2}$$

$$\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r_2} + \mathbf{e}$$

The ciphertext is the tuple $(\mathbf{u}, \mathbf{v})$.

u carries information to remove the mask.

v is the actual part containing the plaintext.

Decryption

$\mathbf{m} = \mathrm{Decode}_{\mathbf{G}}(\mathbf{v} - \mathbf{u}\mathbf{y})$

$\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} = \mathbf{m}\mathbf{G} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e}$

$$\begin{aligned}\mathbf{v} - \mathbf{u}\mathbf{y} &= \mathbf{m}\mathbf{G} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e} - \mathbf{y}\mathbf{r}_1 - \mathbf{h}\mathbf{y}\mathbf{r}_2 \\ &= \mathbf{m}\mathbf{G} + \underset{ww_r}{\mathbf{x}\mathbf{r}_2} + \underset{w_e}{\mathbf{e}} - \underset{ww_r}{\mathbf{y}\mathbf{r}_1} \\ &= \mathbf{m}\mathbf{G} + \underset{2ww_r+w_e}{\mathbf{e}'}\end{aligned}$$

Since $wt(\mathbf{e}') \leq 2ww_r + w_e$, we want to choose parameters w, w_r, w_e as large as possible so that $wt(\mathbf{e}') > \lfloor \frac{d-1}{2} \rfloor$ with negligible probability.

Hamming Quasi-Cyclic

Decryption

$\mathbf{m} = \text{Decode}_{\mathbf{G}}(\mathbf{v} - \mathbf{u}\mathbf{y})$

$\mathbf{v} = \mathbf{mG} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} = \mathbf{mG} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e}$

$$\begin{aligned}
\mathbf{v} - \mathbf{u}\mathbf{y} &= \mathbf{mG} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e} - \mathbf{y}\mathbf{r}_1 - \mathbf{h}\mathbf{y}\mathbf{r}_2 \\
&= \mathbf{mG} + \underset{ww_r}{\mathbf{x}\mathbf{r}_2} + \underset{w_e}{\mathbf{e}} - \underset{ww_r}{\mathbf{y}\mathbf{r}_1} \\
&= \mathbf{mG} + \underset{2ww_r + w_e}{\mathbf{e}'}
\end{aligned}$$

Since $wt(\mathbf{e}') \leq 2ww_r + w_e$, we want to choose parameters w, w_r, w_e as large as possible so that $wt(\mathbf{e}') > \lfloor \frac{d-1}{2} \rfloor$ with negligible probability.

Decryption

$\mathbf{m} = \mathrm{Decode}_{\mathbf{G}}(\mathbf{v} - \mathbf{u}\mathbf{y})$

$\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} = \mathbf{m}\mathbf{G} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e}$

$$\begin{aligned}\mathbf{v} - \mathbf{u}\mathbf{y} &= \mathbf{m}\mathbf{G} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e} - \mathbf{y}\mathbf{r}_1 - \mathbf{h}\mathbf{y}\mathbf{r}_2 \\ &= \mathbf{m}\mathbf{G} + \underset{ww_r}{\mathbf{x}\mathbf{r}_2} + \underset{w_e}{\mathbf{e}} - \underset{ww_r}{\mathbf{y}\mathbf{r}_1} \\ &= \mathbf{m}\mathbf{G} + \underset{2ww_r+w_e}{\mathbf{e}'}\end{aligned}$$

Since $wt(\mathbf{e}') \leq 2ww_r + w_e$, we want to choose parameters w, w_r, w_e as large as possible so that $wt(\mathbf{e}') > \lfloor \frac{d-1}{2} \rfloor$ with negligible probability.

Decryption

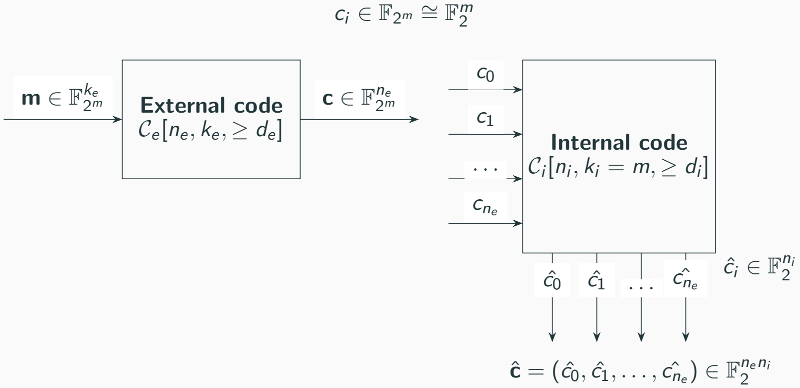
$\mathbf{m} = \text{Decode}_{\mathbf{G}}(\mathbf{v} - \mathbf{u}\mathbf{y})$

$\mathbf{v} = \mathbf{mG} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} = \mathbf{mG} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e}$

$$\begin{aligned} \mathbf{v} - \mathbf{u}\mathbf{y} &= \mathbf{mG} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e} - \mathbf{y}\mathbf{r}_1 - \mathbf{h}\mathbf{y}\mathbf{r}_2 \\ &= \mathbf{mG} + \underset{ww_r}{\mathbf{x}\mathbf{r}_2} + \underset{w_e}{\mathbf{e}} - \underset{ww_r}{\mathbf{y}\mathbf{r}_1} \\ &= \mathbf{mG} + \underset{2ww_r + w_e}{\mathbf{e}'} \end{aligned}$$

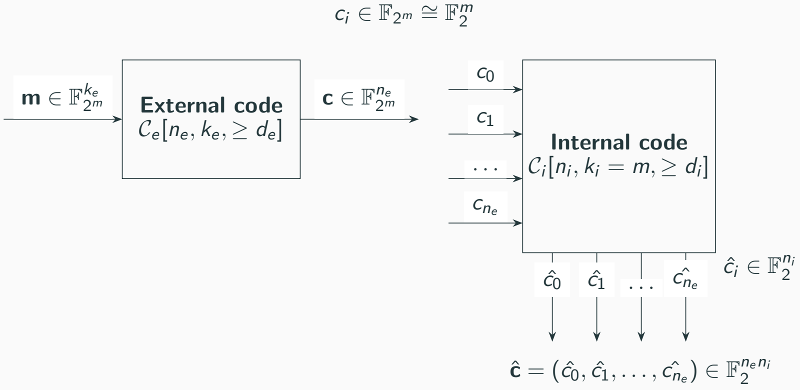
Since $wt(\mathbf{e}') \leq 2ww_r + w_e$, we want to choose parameters w, w_r, w_e as large as possible so that $wt(\mathbf{e}') > \lfloor \frac{d-1}{2} \rfloor$ with negligible probability.

Concatenated codes



The external code is transformed into a binary code of parameters $[n_e n_i, k_e k_i, \geq d_e d_i]$.

Concatenated codes



The external code is transformed into a binary code of parameters $[n_e n_i, k_e k_i, \geq d_e d_i]$.

Reed-Solomon codes

A Reed-Solomon code with elements in $\mathbb{F}_{2^m}$ has the following parameters:

- Length $n = 2^m - 1$
- Minimum distance $d = n - k + 1$ chosen by construction.
- Error correction capacity $t = \lfloor \frac{d-1}{2} \rfloor$

Let α be a primitive element of $\mathbb{F}_{2^m}$, the generator polynomial $g(x)$ of the RS$[n, k, d]$ code is given by

$$g(x) = (x + \alpha)(x + \alpha^2) \dots (x + \alpha^{n-k})$$

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294

Table 1: Reed-Solomon codes and their rates.[1]

Reed-Solomon codes

A Reed-Solomon code with elements in $\mathbb{F}_{2^m}$ has the following parameters:

- Length $n = 2^m - 1$
- Minimum distance $d = n - k + 1$ chosen by construction.
- Error correction capacity $t = \lfloor \frac{d-1}{2} \rfloor$

Let α be a primitive element of $\mathbb{F}_{2^m}$, the generator polynomial $g(x)$ of the RS$[n, k, d]$ code is given by

$$g(x) = (x + \alpha)(x + \alpha^2)\dots(x + \alpha^{n-k})$$

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294

Table 1: Reed-Solomon codes and their rates.[1]

Reed-Solomon codes

A Reed-Solomon code with elements in $\mathbb{F}_{2^m}$ has the following parameters:

- Length $n = 2^m - 1$
- Minimum distance $d = n - k + 1$ chosen by construction.
- Error correction capacity $t = \lfloor \frac{d-1}{2} \rfloor$

Let α be a primitive element of $\mathbb{F}_{2^m}$, the generator polynomial $g(x)$ of the RS$[n, k, d]$ code is given by

$$g(x) = (x + \alpha)(x + \alpha^2) \dots (x + \alpha^{n-k})$$

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294

Table 1: Reed-Solomon codes and their rates.[1]

Shortened Reed-Solomon codes

Reed-Solomon codes can be *shortened* without altering the error correction capacity.

- Shorten by s bits to obtain the $\mathrm{RS}[n-s, k-s, d]$ code.

The encoder takes $k-s$ bits of payload and s padding bits and outputs a codeword holding $n-s$ useful symbols and s bits of padding that are easy to discard with systematic encoding.

We re-insert those s padding bits into the decoder.

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294
RS-S1	46	16	**15**	2.875
RS-S2	56	24	**16**	2.333
RS-S3	90	32	**29**	2.813

Shortened Reed-Solomon codes

Reed-Solomon codes can be *shortened* without altering the error correction capacity.

- Shorten by s bits to obtain the RS$[n-s, k-s, d]$ code.

The encoder takes $k-s$ bits of payload and s padding bits and outputs a codeword holding $n-s$ useful symbols and s bits of padding that are easy to discard with systematic encoding.

We re-insert those s padding bits into the decoder.

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294
RS-S1	46	16	**15**	2.875
RS-S2	56	24	**16**	2.333
RS-S3	90	32	**29**	2.813

Shortened Reed-Solomon codes

Reed-Solomon codes can be *shortened* without altering the error correction capacity.

- Shorten by s bits to obtain the RS$[n-s, k-s, d]$ code.

The encoder takes $k-s$ bits of payload and s padding bits and outputs a codeword holding $n-s$ useful symbols and s bits of padding that are easy to discard with systematic encoding.

We re-insert those s padding bits into the decoder.

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294
RS-S1	46	16	**15**	2.875
RS-S2	56	24	**16**	2.333
RS-S3	90	32	**29**	2.813

Shortened Reed-Solomon codes

Reed-Solomon codes can be *shortened* without altering the error correction capacity.

- Shorten by s bits to obtain the RS$[n-s, k-s, d]$ code.

The encoder takes $k-s$ bits of payload and s padding bits and outputs a codeword holding $n-s$ useful symbols and s bits of padding that are easy to discard with systematic encoding.

We re-insert those s padding bits into the decoder.

Code	n	k	t	R
RS-1	255	225	**15**	1.133
RS-2	255	223	**16**	1.143
RS-3	255	197	**29**	1.294
RS-S1	46	16	**15**	2.875
RS-S2	56	24	**16**	2.333
RS-S3	90	32	**29**	2.813

Decoding Reed-Solomon codes (Recap)

1 - Calculate the syndromes.

We receive $r(x) = c(x) + e(x)$ and assume $wt(e) \leq t$.

- $e = (0, \ldots, \underset{i_1}{1}, \ldots, \underset{i_2}{1}, \ldots, \underset{i_t}{1}, \ldots, 0)$
- $e(x) = x^{i_1} + x^{i_2} + \ldots + x^{i_t}$

Let $\mathbb{F}_{2^m} = \langle \alpha \rangle$, all α^i are roots of $g(x)$ so of $c(x)$.

The syndrome $s_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$

For example:

- $s_1 = e(\alpha) = \alpha^{i_1} + \alpha^{i_2} + \alpha^{i_3} + \ldots + \alpha^{i_t}$
- $s_2 = e(\alpha^2) = \alpha^{2i_1} + \alpha^{2i_2} + \alpha^{2i_3} + \ldots + \alpha^{2i_t}$
- $s_3 = e(\alpha^3) = \alpha^{3i_1} + \alpha^{3i_2} + \alpha^{3i_3} + \ldots + \alpha^{3i_t}$
- $\ldots$

Decoding Reed-Solomon codes (Recap)

1 - Calculate the syndromes.

We receive $r(x) = c(x) + e(x)$ and assume $wt(e) \leq t$.

- $e = (0, \ldots, \underset{i_1}{1}, \ldots, \underset{i_2}{1}, \ldots, \underset{i_t}{1}, \ldots, 0)$
- $e(x) = x^{i_1} + x^{i_2} + \ldots + x^{i_t}$

Let $\mathbb{F}_{2^m} = \langle \alpha \rangle$, all α^i are roots of $g(x)$ so of $c(x)$.

The syndrome $s_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$

For example:

- $s_1 = e(\alpha) = \alpha^{i_1} + \alpha^{i_2} + \alpha^{i_3} + \ldots + \alpha^{i_t}$
- $s_2 = e(\alpha^2) = \alpha^{2i_1} + \alpha^{2i_2} + \alpha^{2i_3} + \ldots + \alpha^{2i_t}$
- $s_3 = e(\alpha^3) = \alpha^{3i_1} + \alpha^{3i_2} + \alpha^{3i_3} + \ldots + \alpha^{3i_t}$
- $\ldots$

Decoding Reed-Solomon codes (Recap)

1 - Calculate the syndromes.

We receive $r(x) = c(x) + e(x)$ and assume $wt(e) \leq t$.

- $e = (0, \ldots, \underset{i_1}{1}, \ldots, \underset{i_2}{1}, \ldots, \underset{i_t}{1}, \ldots, 0)$
- $e(x) = x^{i_1} + x^{i_2} + \ldots + x^{i_t}$

Let $\mathbb{F}_{2^m} = \langle \alpha \rangle$, all α^i are roots of $g(x)$ so of $c(x)$.

The syndrome $s_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$

For example:

- $s_1 = e(\alpha) = \alpha^{i_1} + \alpha^{i_2} + \alpha^{i_3} + \ldots + \alpha^{i_t}$
- $s_2 = e(\alpha^2) = \alpha^{2i_1} + \alpha^{2i_2} + \alpha^{2i_3} + \ldots + \alpha^{2i_t}$
- $s_3 = e(\alpha^3) = \alpha^{3i_1} + \alpha^{3i_2} + \alpha^{3i_3} + \ldots + \alpha^{3i_t}$
- $\ldots$

Decoding Reed-Solomon codes (Recap)

1 - Calculate the syndromes.

We receive $r(x) = c(x) + e(x)$ and assume $wt(e) \leq t$.

- $e = (0, \ldots, \underset{i_1}{1}, \ldots, \underset{i_2}{1}, \ldots, \underset{i_t}{1}, \ldots, 0)$
- $e(x) = x^{i_1} + x^{i_2} + \ldots + x^{i_t}$

Let $\mathbb{F}_{2^m} = \langle \alpha \rangle$, all α^i are roots of $g(x)$ so of $c(x)$.

The syndrome $s_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$

For example:

- $s_1 = e(\alpha) = \alpha^{i_1} + \alpha^{i_2} + \alpha^{i_3} + \ldots + \alpha^{i_t}$
- $s_2 = e(\alpha^2) = \alpha^{2i_1} + \alpha^{2i_2} + \alpha^{2i_3} + \ldots + \alpha^{2i_t}$
- $s_3 = e(\alpha^3) = \alpha^{3i_1} + \alpha^{3i_2} + \alpha^{3i_3} + \ldots + \alpha^{3i_t}$
- $\ldots$

2 - Error Locator Polynomial

Let $z_j = \alpha^{i_j}$, define the polynomial

$$\begin{aligned}\sigma(x) &= (1 + z_1 x) \cdots (1 + z_t x) \\ &= 1 + \sigma_1 x + \sigma_2 x^2 + \ldots + \sigma_t x^t\end{aligned}$$

The σ_i are the *error coefficients*. Finding them allows us to find the roots α^{-i_j} of $\sigma(x)$ to locate the errors.

3 - Error coefficients.

Let us fix $t = 3$ for the following.

We have a linear relation:

$$s_{i+4} + \sigma_1 s_{i+3} + \sigma_2 s_{i+2} + \sigma_3 s_{i+1} = 0$$

for $i = 0, 1, 2$, i.e.,

$$\begin{bmatrix} s_3 & s_2 & s_1 \\ s_4 & s_3 & s_2 \\ s_5 & s_4 & s_3 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix} = \begin{bmatrix} s_4 \\ s_5 \\ s_6 \end{bmatrix}$$

From the relation

$$s_{i+4} + \sigma_1 s_{i+3} + \sigma_2 s_{i+2} + \sigma_3 s_{i+1} = 0, \quad i = 0, 1, \ldots, t-1$$

we can obtain σ_i's for $\sigma(x)$.

Solving $\sigma(x) = 0$ gives the error locations $i_1, i_2, \ldots, i_t$.

Reed-Muller codes take advantage of Lagrange interpolation to decode.

The message is a polynomial in m variables over $\mathbb{F}_q$ and of algebraic degree at most r. This defines a $RM(r, m)$ code.

For $r = 2, m = 3$, let a message

$$f(x_1, x_2, x_3) = \underline{f_0} + \underline{f_1 x + f_2 x_2 + f_3 x_3} + \underline{f_4 x_1 x_2 + f_5 x_1 x_3 + f_6 x_2 x_3}$$

As such, Reed-Muller codes have dimension $k = \sum_{i=0}^{r} \binom{m}{i}$.

Reed-Muller codes take advantage of Lagrange interpolation to decode.

The message is a polynomial in m variables over $\mathbb{F}_q$ and of algebraic degree at most r. This defines a $RM(r, m)$ code.

For $r = 2, m = 3$, let a message

$$f(x_1, x_2, x_3) = \underline{f_0} + \underline{f_1 x + f_2 x_2 + f_3 x_3} + \underline{f_4 x_1 x_2 + f_5 x_1 x_3 + f_6 x_2 x_3}$$

As such, Reed-Muller codes have dimension $k = \sum_{i=0}^{r} \binom{m}{i}$.

Reed-Muller codes

Reed-Muller codes take advantage of Lagrange interpolation to decode.

The message is a polynomial in m variables over $\mathbb{F}_q$ and of algebraic degree at most r. This defines a $RM(r, m)$ code.

For $r = 2, m = 3$, let a message

$$f(x_1, x_2, x_3) = \underline{f_0} + \underline{f_1 x + f_2 x_2 + f_3 x_3} + \underline{f_4 x_1 x_2 + f_5 x_1 x_3 + f_6 x_2 x_3}$$

As such, Reed-Muller codes have dimension $k = \sum_{i=0}^{r} \binom{m}{i}$.

Reed-Muller codes

Reed-Muller codes take advantage of Lagrange interpolation to decode.

The message is a polynomial in m variables over $\mathbb{F}_q$ and of algebraic degree at most r. This defines a $RM(r, m)$ code.

For $r = 2, m = 3$, let a message

$$f(x_1, x_2, x_3) = \underline{f_0} + \underline{f_1 x + f_2 x_2 + f_3 x_3} + \underline{f_4 x_1 x_2 + f_5 x_1 x_3 + f_6 x_2 x_3}$$

As such, Reed-Muller codes have dimension $k = \sum_{i=0}^{r} \binom{m}{i}$.

Encoding Reed-Muller codes

The codeword of a message consists on all possible evaluation points. Over $\mathbb{F}_2$, that makes the length of the code $n = 2^m$.

$$
\begin{aligned}
f(0,0,0) &= c_0 \\
f(0,0,1) &= c_1 \\
f(0,1,0) &= c_2 \\
&\dots \\
f(1,1,1) &= c_{n-1}
\end{aligned}
$$

$c = (c_0, \dots, c_{n-1})$ is the codeword.

A μ-duplicated Reed-Muller code is simply repeating μ times the codeword symbols.

HQC instance	RM code	Multiplicity μ	Duplicated RM code
hqc-128	$[128, 8, 64]$	3	$[384, 8, 192]$
hqc-192	$[128, 8, 64]$	5	$[640, 8, 320]$
hqc-256	$[128, 8, 64]$	5	$[640, 8, 320]$

Table 3: Duplicated Reed-Muller codes.[1]

Let $\mu = 3$, a duplicated codeword $\mathbf{c}'$ from $\mathbf{c}$ is
$\mathbf{c}' = (c_0, c_0, c_0, c_1, c_1, c_1, \ldots, c_{n-1}, c_{n-1}, c_{n-1})$.

Duplicated Reed-Muller codes

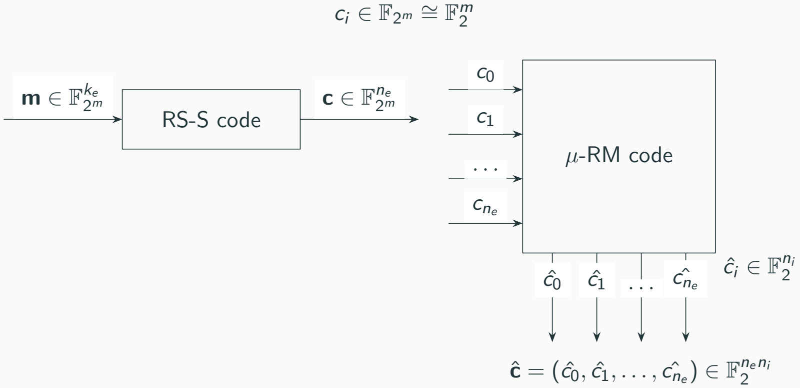
A μ-duplicated Reed-Muller code is simply repeating μ times the codeword symbols.

HQC instance	RM code	Multiplicity μ	Duplicated RM code
hqc-128	$[128, 8, 64]$	3	$[384, 8, 192]$
hqc-192	$[128, 8, 64]$	5	$[640, 8, 320]$
hqc-256	$[128, 8, 64]$	5	$[640, 8, 320]$

Table 3: Duplicated Reed-Muller codes.[1]

Let $\mu = 3$, a duplicated codeword $\mathbf{c}'$ from $\mathbf{c}$ is
$\mathbf{c}' = (c_0, c_0, c_0, c_1, c_1, c_1, \ldots, c_{n-1}, c_{n-1}, c_{n-1})$.

Decoding in HQC - overview



Decrypting (decoding) failure rate

Decryption

$\mathbf{m} = \text{Decode}(\mathbf{v} - \mathbf{u}\mathbf{y})$

$\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} = \mathbf{m}\mathbf{G} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e}$

$$\begin{aligned} \mathbf{v} - \mathbf{u}\mathbf{y} &= \mathbf{m}\mathbf{G} + \underset{ww_r}{\mathbf{x}\mathbf{r}_2} + \underset{w_e}{\mathbf{e}} - \underset{ww_r}{\mathbf{y}\mathbf{r}_1} \\ &= \mathbf{m}\mathbf{G} + \underset{2ww_r + w_e}{\mathbf{e}'} \end{aligned}$$

- $v = mG + sr_2 + e$ is a noisy codeword but $s = x + hy$ is **not** a low weight polynomial. Its noise is way above the decoding radius.
- if $wt(\mathbf{e}')$ is outside the decoding radius, we face a decoding failure.

Decrypting (decoding) failure rate

Decryption

$\mathbf{m} = \text{Decode}(\mathbf{v} - \mathbf{u}\mathbf{y})$

$\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} = \mathbf{m}\mathbf{G} + \mathbf{x}\mathbf{r}_2 + \mathbf{h}\mathbf{y}\mathbf{r}_2 + \mathbf{e}$

$$\begin{aligned} \mathbf{v} - \mathbf{u}\mathbf{y} &= \mathbf{m}\mathbf{G} + \underset{ww_r}{\mathbf{x}\mathbf{r}_2} + \underset{w_e}{\mathbf{e}} - \underset{ww_r}{\mathbf{y}\mathbf{r}_1} \\ &= \mathbf{m}\mathbf{G} + \underset{2ww_r+w_e}{\mathbf{e}'} \end{aligned}$$

- $v = mG + sr_2 + e$ is a noisy codeword but $s = x + hy$ is **not** a low weight polynomial. Its noise is way above the decoding radius.
- if $wt(\mathbf{e}')$ is outside the decoding radius, we face a decoding failure.

Fast Decoding of the 1-st order Reed-Muller Codes

The 1-st order RM codes can be efficiently decoded using a fast Hadamard transform. This can be efficiently done in 3 steps:

1. Build the 2^m-order Hadamard matrix.
2. Apply Binary Phase Shift Keying on the received word r.
3. Compute its Walsh coefficients.

The Hadamard matrix of order n is defined as

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \text{ with } H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Actually this recursion helps achieve *fast* transform and drop the complexity from $O(2^m \times 2^m)$ to $O(m2^m)$.

Fast Decoding of the 1-st order Reed-Muller Codes

The 1-st order RM codes can be efficiently decoded using a fast Hadamard transform. This can be efficiently done in 3 steps:

1. Build the 2^m-order Hadamard matrix.
2. Apply Binary Phase Shift Keying on the received word r.
3. Compute its Walsh coefficients.

The Hadamard matrix of order n is defined as

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \text{ with } H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Actually this recursion helps achieve *fast* transform and drop the complexity from $O(2^m \times 2^m)$ to $O(m2^m)$.

Fast Decoding of the 1-st order Reed-Muller Codes

The 1-st order RM codes can be efficiently decoded using a fast Hadamard transform. This can be efficiently done in 3 steps:

1. Build the 2^m-order Hadamard matrix.
2. Apply Binary Phase Shift Keying on the received word r.
3. Compute its Walsh coefficients.

The Hadamard matrix of order n is defined as

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \text{ with } H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Actually this recursion helps achieve *fast* transform and drop the complexity from $O(2^m \times 2^m)$ to $O(m2^m)$.

Decrypting (decoding) failure rate

Since Reed-Muller codes follow a maximum-likelihood strategy for decoding, there is no exact decoding probability formula.

There is an upper bound for the DFR of the concatenated code given by:

$$\mathrm{DFR}_{\mathcal{C}} = \sum_{k=t_e}^{n_e} \binom{n_e}{k} p_{\mathrm{RM}}^k (1 - p_{\mathrm{RM}})^{n_e - k}$$

where p_{RM} is the lower bound on the probability decoding of Reed-Muller codes.

Decrypting (decoding) failure rate

Since Reed-Muller codes follow a maximum-likelihood strategy for decoding, there is no exact decoding probability formula.

There is an upper bound for the DFR of the concatenated code given by:

$$\mathrm{DFR}_{\mathcal{C}} = \sum_{k=t_e}^{n_e} \binom{n_e}{k} p_{\mathrm{RM}}^{k} (1 - p_{\mathrm{RM}})^{n_e - k}$$

where p_{RM} is the lower bound on the probability decoding of Reed-Muller codes.

DFR and security

Choosing w, w_r, w_e for negligible failure probability.

Instance	n_e	n_i	n	w	$w_r = w_e$	security	DFR
hqc-128	46	384	17,669	66	75	128	$< 2^{-128}$
hqc-192	56	640	35,851	100	114	192	$< 2^{-192}$
hqc-256	90	640	57,637	131	149	256	$< 2^{-256}$

Table 4: Security parameters for HQC.[1]

DFR and security

Structural attacks. A generic attack is the DOOM attack[6] that gains $O(\sqrt{n})$ because of cyclicity ($O(n)$ for MDPC).

Some attacks[3, 4, 6] are efficient when $x^n - 1$ has many low degree factors but become inefficient when $x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \ldots + x + 1)$ which is the case when n is a primitive prime.

This is why $n = n_e n_i + l$ is used in HQC. The last l bits are truncated, breaking the quasi-cyclicity and weakening the attacker.

DFR and security

Structural attacks. A generic attack is the DOOM attack[6] that gains $O(\sqrt{n})$ because of cyclicity ($O(n)$ for MDPC).

Some attacks[3, 4, 6] are efficient when $x^n - 1$ has many low degree factors but become inefficient when $x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \ldots + x + 1)$ which is the case when n is a primitive prime.

This is why $n = n_e n_i + l$ is used in HQC. The last l bits are truncated, breaking the quasi-cyclicity and weakening the attacker.

DFR and security

Security of code-based hard problems. The best attack remains Prange's ISD [5] of exponential order.

It has been more than 60 years and only improvements of the exponent constant have been made.

Performance of primitives

Instance	KeyGen	Encapsulation	Decapsulation
hqc-128	105	197	360
hqc-192	244	460	746
hqc-256	447	844	1,410

Table 5: HQC performance (x86_64 kilocycles)[2]

Instance	KeyGen	Encapsulation	Decapsulation
mceliece6960119	602,164	167	252
mceliece8192128	686,110	203	269

Table 6: Classic McEliece performance (x86_64 kilocycles)[2]

Key size

Instance	Public key	Private key	Ciphertext
hqc-128	2,249	56	4,497
hqc-192	4,522	64	9,042
hqc-256	7,245	72	14,485

Table 7: HQC key size (bytes)[2]

Instance	Public key	Private key	Ciphertext
mceliece6960119	1,047,319	13,948	194
mceliece8192128	1,357,824	14,120	208

Table 8: Classic McEliece key size (bytes)[2]

Conclusion

- BIKE also uses Quasi-Cyclic codes but in the same way as McElice. While the DFR analysis of HQC depends on very well studied codes which makes it trustworthy, BIKE depends on an extrapolation method obtained through a much more complicated analysis.

  I absolutely believe in the security of McEliece. [...] This has resisted security for many years, [...] I don't think this is the reason why NIST decided not to standardize it. The size is definitely a consideration but I believe that NIST primary goal was to have a general purpose KEM.
  — Edoardo Persichetti, HQC co-inventor.

C. Aguilar-Melchor, J.-C. Deneuville, A. Dion, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J. Bos, P. Gaborit, J. Lacan, E. Persichetti, J.-M. Robert, P. Véron, and G. Zémor.
Hamming quasi-cyclic (hqc).
02 2025.

G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, and N. Waller.
Status report on the fourth round of the nist post-quantum cryptography standardization process, March 2025.

- Q. Guo, T. Johansson, and C. Löndahl.
A new algorithm for solving Ring-LPN with a reducible polynomial, 2014.

- C. Löndahl, T. Johansson, M. Shooshtari, M. Ahmadian Attari, and M. Aref.
Squaring attacks on mceliece public-key cryptosystems using quasi-cyclic codes of even dimension.
Designs, Codes and Cryptography, 80, 06 2015.

- E. Prange.
The use of information sets in decoding cyclic codes.
IRE Transactions on Information Theory, 8(5):5–9, 1962.

N. Sendrier.
Decoding one out of many.
Cryptology ePrint Archive, Paper 2011/367, 2011.