



Code-based Cryptography

Angela Robinson

ICERM Graduate Workshop on LAFFA, August 20, 2025

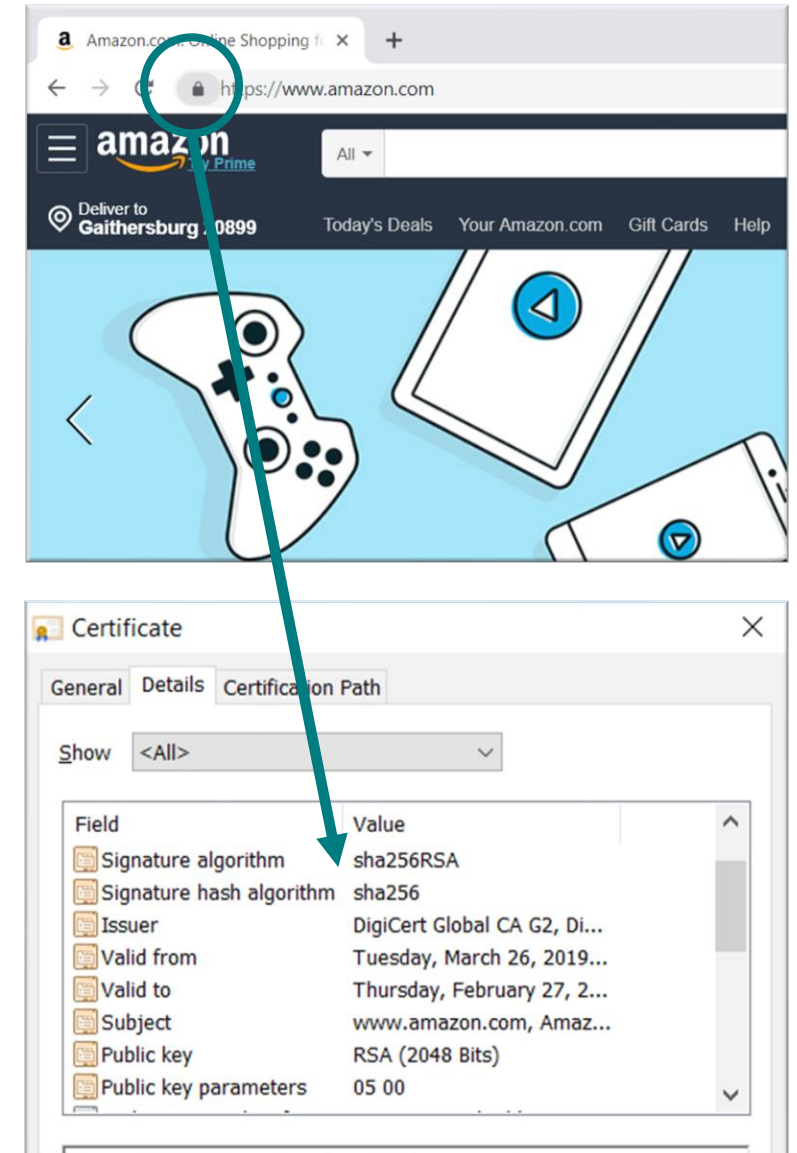


Motivation

Cryptography sightings

Secure websites are protected using cryptography

- Encryption - confidentiality of messages
- Digital signature - authentication
- Certificates - verify identity



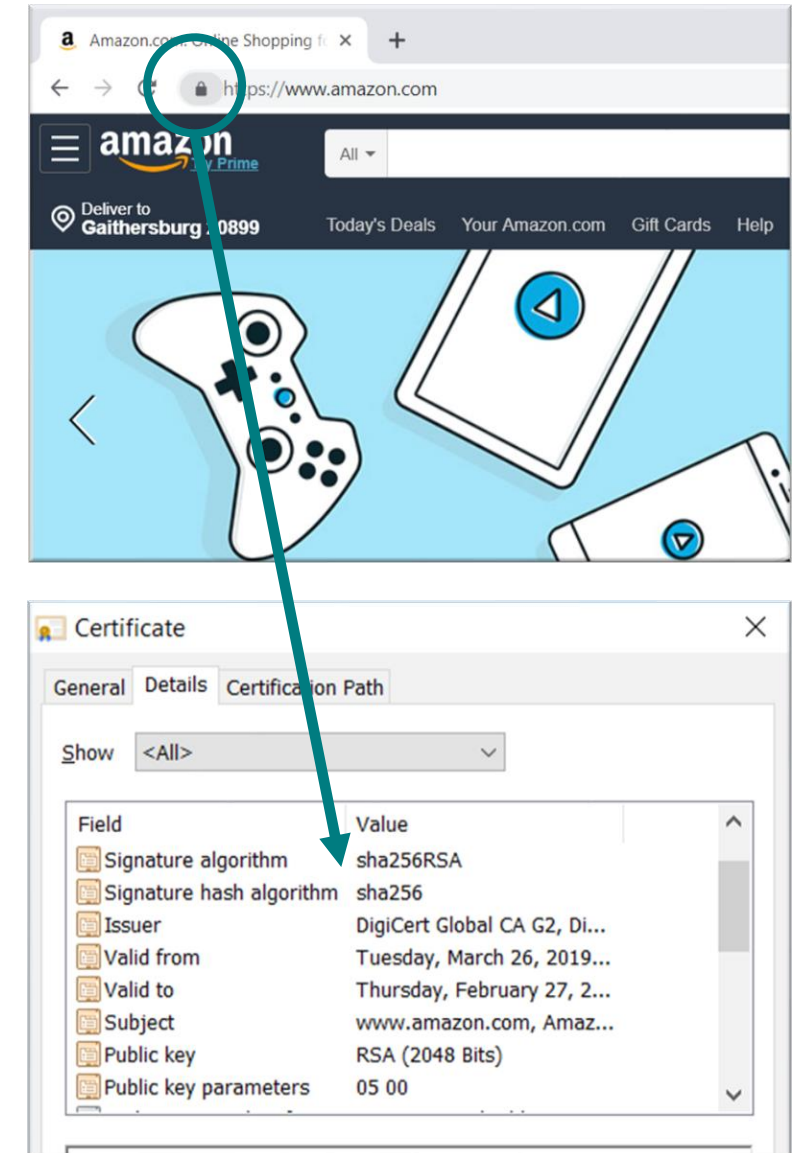
Cryptography sightings

Secure websites are protected using cryptography

- Encryption - confidentiality of messages
- Digital signature - authentication
- Certificates - verify identity

Security is quantified by the resources it takes to break a cryptosystem

- Best known cryptanalysis
- Cost of implementing the cryptanalysis

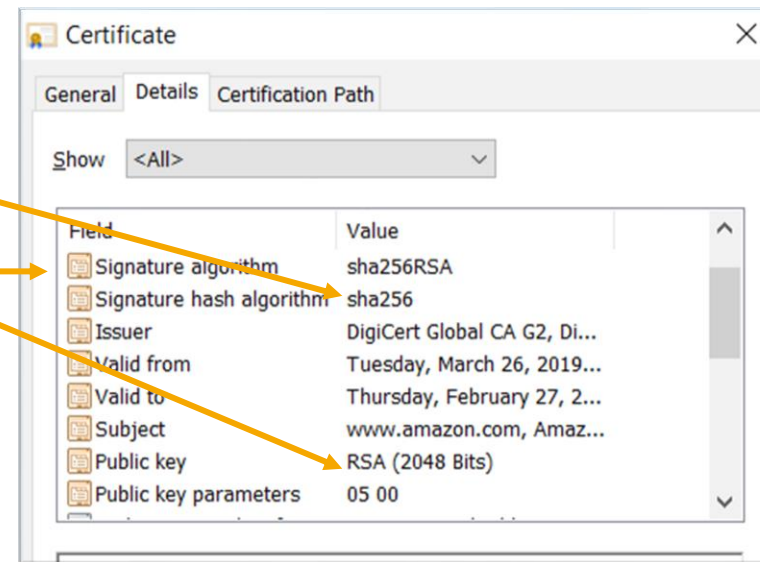


Cryptography at NIST

Cryptographic Standards

- Hash functions
- Encryption schemes
- Digital signatures
- ...

Example



Present threat

Some current NIST standards are vulnerable to quantum threat.

Peter Shor (1994): polynomial-time quantum algorithm that breaks

- Integer factorization problem (RSA)
- Discrete logarithm problem (Diffie-Hellman Key Exchange, Elliptic Curve DH, ...)
- **Impact: a full-scale quantum computer can break today's public key crypto**

Options for mitigating the threat

- ~~Stop using public key crypto~~ not practical
- Find quantum-safe public key crypto

NIST PQC Standardization effort

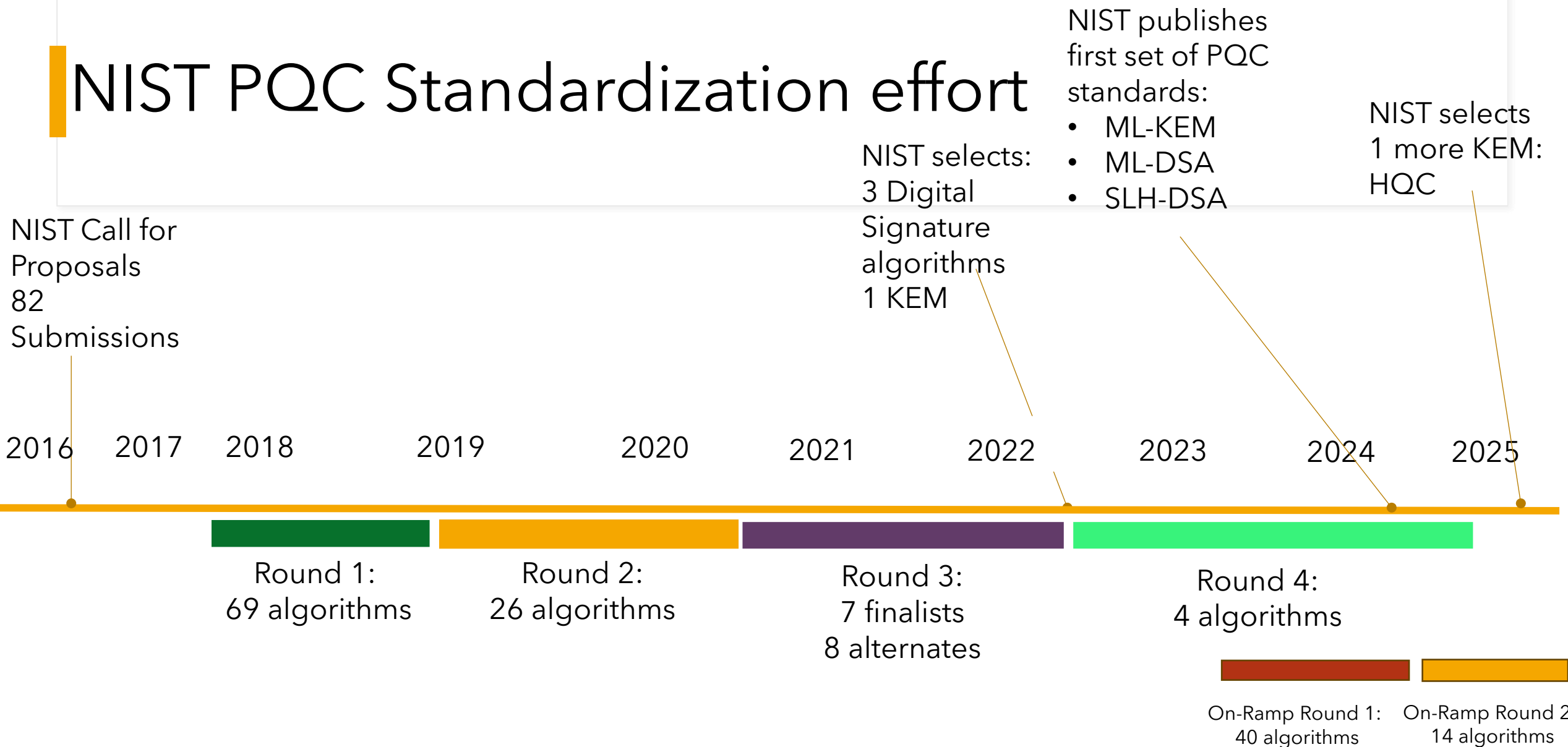
Call for public key cryptographic schemes believed to be quantum-resistant (2016)

- Received 80+ submissions (2017)
- **Code-based algorithms** under consideration during Rounds 2-4
 - Round 2: BIKE, Classic McEliece*, HQC, LEDAcrypt**, NTS-KEM*
 - Round 3: BIKE, Classic McEliece, HQC
 - Round 4: BIKE, Classic McEliece, HQC

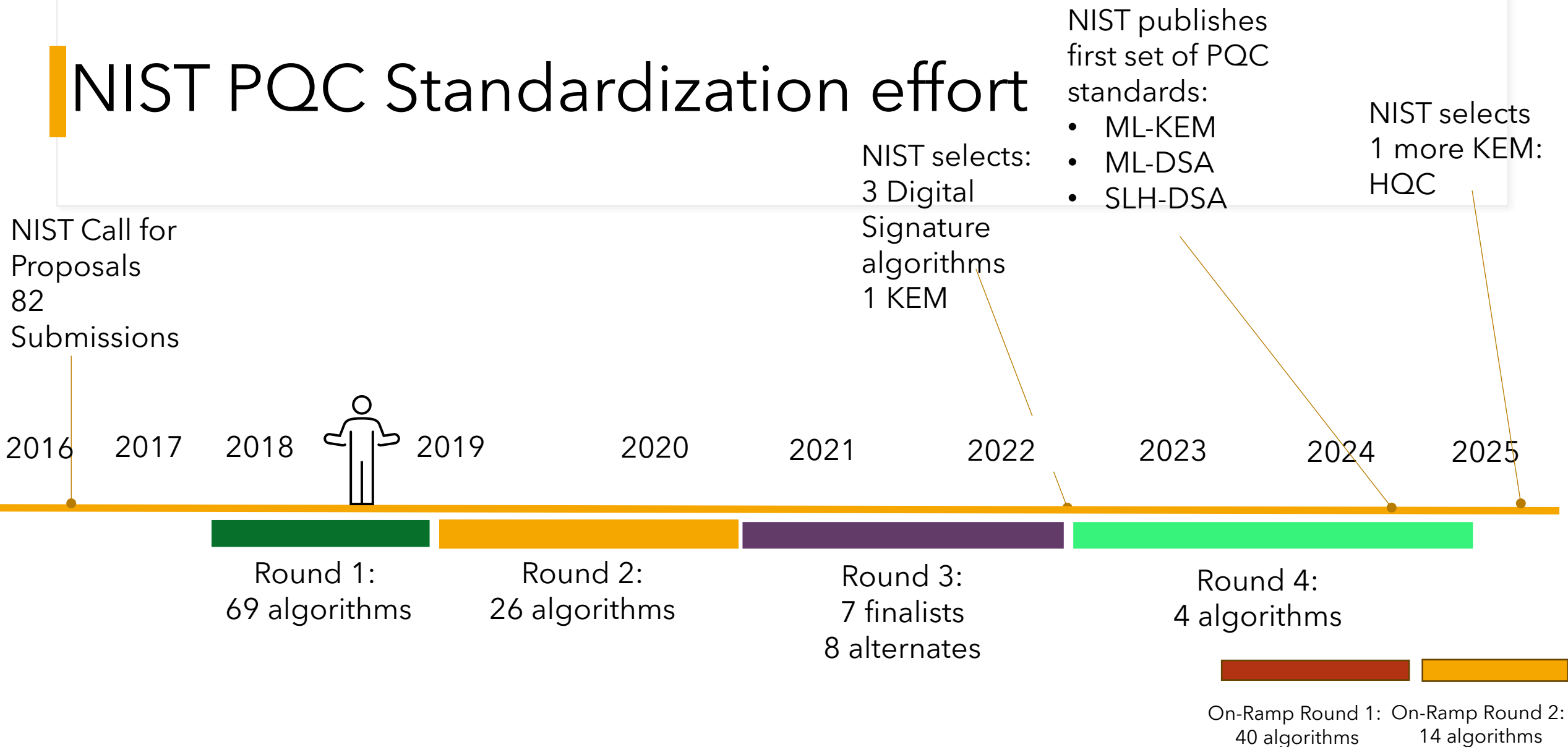
*merged during Round 2

** broken [APRS2020]

NIST PQC Standardization effort



NIST PQC Standardization effort





Background

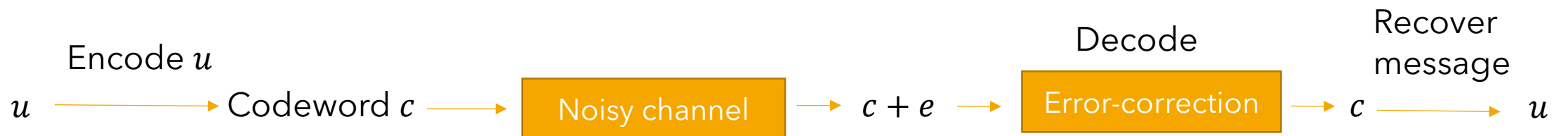
Error-correcting codes

Error-correcting codes

Error-detecting and error-correcting codes are designed to locate and remove noise from messages received over noisy channels



This is accomplished by adding some **extra bits** to the message before transmission that will enable error-detection and error-correction



Definitions

Definition: a **vector space** over a field \mathbb{F} consists of a set V (of vectors) and a set \mathbb{F} (of scalars) along with operations $+$ and \cdot such that

- If $x, y \in V$, then $x + y \in V$
- If $x \in V$ and $\alpha \in \mathbb{F}$, then $\alpha \cdot x \in V$

Definition: Let V be a vector space. A linearly independent spanning set B for V is called a **basis**.

Definition: The **dimension** of a vector space is the cardinality of its bases

Example: \mathbb{R}^3 is a vector space, $B = \{1 \ 0 \ 0, \ 0 \ 1 \ 0, \ 0 \ 0 \ 1\}$ is the standard basis for \mathbb{R}^3

$$\dim(\mathbb{R}^3) = 3.$$

Binary linear code

Definition: a **binary linear code** $C(n, k)$ is a k -dimensional subspace of \mathbb{F}_2^n .

The code $C: \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ maps information vectors to codewords



Binary linear code

Definition: a **binary linear code** $C(n, k)$ is a k -dimensional subspace of \mathbb{F}_2^n .

The code $C: \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ maps information vectors to codewords

How do we describe a code?



Binary linear code

Definition: a **binary linear code** $C(n, k)$ is a k -dimensional subspace of \mathbb{F}_2^n .

The code $C: \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ maps information vectors to codewords

How do we describe a code?

1. Select a basis of the k -dim vector space $\{g_0, g_1, \dots, g_{k-1}\}$
2. Basis forms a **generator matrix** $G_{k \times n}$ of the code

$$G = \begin{bmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,n-1} \end{bmatrix}$$



Descriptions of a code $\mathcal{C}(n, k)$

Two equivalent descriptions of $\mathcal{C}(n, k)$

- **Generator matrix**

- Encoding: multiply k -bit information word u by G
- codewords are x such that there's a solution u to $uG = x$

u $\xrightarrow{\text{Encode } u}$ uG is codeword c

Descriptions of a code $\mathcal{C}(n, k)$

Two equivalent descriptions of $\mathcal{C}(n, k)$

- **Generator matrix**
 - Encoding: multiply k -bit information word u by G
 - codewords are x such that there's a solution u to $uG = x$
- **Parity-check matrix** H (dimension $(n - k) \times n$)
 - $GH^T = 0$
 - codewords are x such that $Hx^T = 0$
 - Product of generic n -bit vector with H^T is called a **syndrome**

Encode u
 $u \longrightarrow uG$ is codeword c

Descriptions of a code $\mathcal{C}(n, k)$

Parity-check matrix H (dimension $(n - k) \times n$)

- $GH^T = 0$
- codewords are \mathbf{x} such that $H\mathbf{x}^T = 0$
- Product of generic n -bit vector with H^T is called a **syndrome**

Example: Let H, x_1, x_2 be as follows.

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$x_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1]$$

$$x_2 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]$$

Descriptions of a code $\mathcal{C}(n, k)$

Parity-check matrix H (dimension $(n - k) \times n$)

- $GH^T = 0$
- codewords are x such that $Hx^T = 0$
- Product of generic n -bit vector with H^T is called a **syndrome**

Example: Let H, x_1, x_2 be as follows.

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$x_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1]$$

$$x_2 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]$$

$$Hx_1^T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Descriptions of a code $\mathcal{C}(n, k)$

Parity-check matrix H (dimension $(n - k) \times n$)

- $GH^T = 0$
- codewords are x such that $Hx^T = 0$
- Product of generic n -bit vector with H^T is called a **syndrome**

Example: Let H, x_1, x_2 be as follows.

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$x_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1]$$

$$x_2 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]$$

$$Hx_1^T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

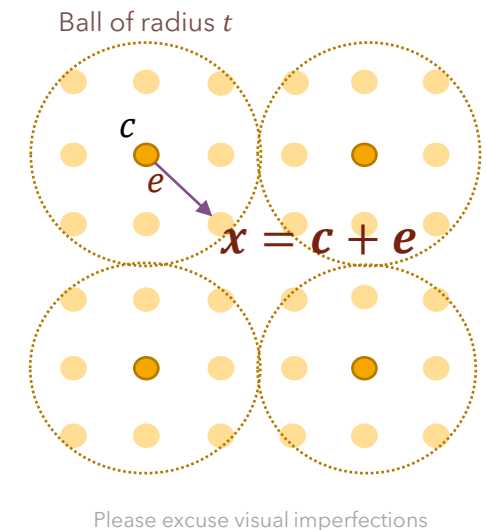
Syndrome is nonzero, so x_1 is not in the code defined by H .

Decoding problems

General Decoding Problem: Given an $[n, k, d]$ linear code C , $t = \left\lfloor \frac{d-1}{2} \right\rfloor$, and a vector $x \in \mathbb{F}^n$, find a codeword $c \in C$ such that $\text{dist}(x, c) \leq t$.

Note: If $x = c + e$, and e is a vector with $|e| \leq t$, then x is uniquely determined.

Shown to be NP-complete for **general linear codes** in 1978 (Berlekamp, McEliece, Tilborg) by reducing the three-dimensional matching problem to these problems.

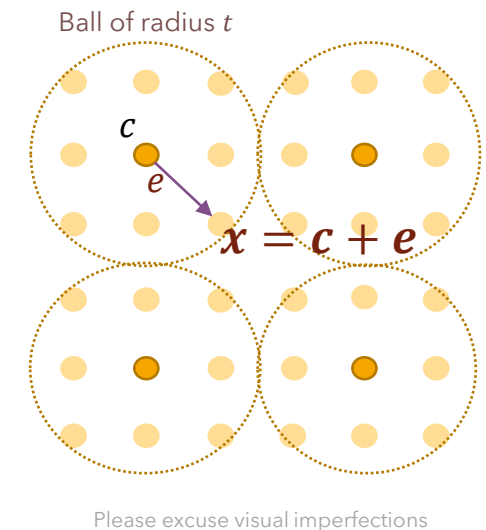


Decoding problems

General Decoding Problem: Given an $[n, k, d]$ linear code C , $t = \left\lfloor \frac{d-1}{2} \right\rfloor$, and a vector $x \in \mathbb{F}^n$, find a codeword $c \in C$ such that $\text{dist}(x, c) \leq t$.

Note: Not all codes have a known minimum distance d . Rewrite problems in terms of linear (n, k) codes.

Shown to be NP-complete for **general linear codes** in 1978 (Berlekamp, McEliece, Tilborg) by reducing the three-dimensional matching problem to these problems.



Decoding problems

Let $\mathcal{C}(n, k)$ be a linear code over finite field \mathbb{F} .

General decoding problem

Given a vector $\mathbf{x} \in \mathbb{F}^n$, a target weight $t > 0$,
find a codeword $\mathbf{c} \in \mathbb{F}^n$ such that $\text{dist}(\mathbf{x}, \mathbf{c}) \leq t$.

Syndrome-decoding problem.

Given a parity check matrix $\mathbf{H} \in \mathbb{F}^{(n-k) \times n}$, a syndrome $\mathbf{s} \in \mathbb{F}^{n-k}$, a target weight $t > 0$,
find a vector $\mathbf{e} \in \mathbb{F}^n$ such that $\text{wt}(\mathbf{e}) = t$ and $\mathbf{H} \cdot \mathbf{e}^T = \mathbf{s}$.


Codeword-finding problem

Given a parity check matrix $\mathbf{H} \in \mathbb{F}^{(n-k) \times n}$ and a target weight $w > 0$
find a vector $\mathbf{e} \in GF_2^n$ such that $\text{wt}(\mathbf{e}) = w$ and $\mathbf{H} \cdot \mathbf{e}^T = \mathbf{0}$.

Relevance

In general, code-based cryptosystems rely upon this property:

- Encryption (some sort of matrix-vector product) is **easy to compute**
- Decryption is **difficult** without the trapdoor (the secret key which enables efficient decoding)



McEliece Cryptosystem

McEliece cryptosystem

First code-based cryptosystem.

Designed by Robert McEliece, presented in 1978.

McEliece cryptosystem

First code-based cryptosystem.

Designed by Robert McEliece, presented in 1978.

Idea: “hide” a message by converting it into a codeword, then add as many errors as the code is capable of correcting

McEliece cryptosystem

First code-based cryptosystem.

Designed by Robert McEliece, presented in 1978.

Idea: “hide” a message by converting it into a codeword, then add as many errors as the code is capable of correcting

Let $\mathcal{C}[n, k, d]$ be a linear code with a fast decoding algorithm that can correct t or fewer errors

- Let G be a generator matrix for \mathcal{C}
- Let S be a $k \times k$ invertible matrix
- Let P be an $n \times n$ permutation matrix

McEliece cryptosystem

Let $\mathcal{C}[n, k, d]$ be a linear code with a fast decoding algorithm that can correct t or fewer errors

- Let G be a generator matrix for \mathcal{C}
- Let S be a $k \times k$ invertible matrix
- Let P be an $n \times n$ permutation matrix

Define public key $G' = SG P$ with private key S, G, P

- Encrypt: $m \rightarrow mG' + e, wt(e) \leq t$
- Decrypt:
 1. Multiply $(mG' + e)P^{-1} = mSG + e'$

$$wt(e) = wt(e')$$

McEliece cryptosystem

Let $\mathcal{C}[n, k, d]$ be a linear code with a fast decoding algorithm that can correct t or fewer errors

- Let G be a generator matrix for \mathcal{C}
- Let S be a $k \times k$ invertible matrix
- Let P be an $n \times n$ permutation matrix

Define public key $G' = SG P$ with private key S, G, P

Encrypt: $m \rightarrow mG' + e, wt(e) \leq t$

Decrypt:

1. Multiply $(mG' + e)P^{-1} = mSG + e'$ $wt(e) = wt(e')$
2. $mSG + e' \longrightarrow$ Fast decoding algorithm $\longrightarrow mSG$
3. .

McEliece cryptosystem

Let $\mathcal{C}[n, k, d]$ be a linear code with a fast decoding algorithm that can correct t or fewer errors

- Let G be a generator matrix for \mathcal{C}
- Let S be a $k \times k$ invertible matrix
- Let P be an $n \times n$ permutation matrix

Define public key $G' = SG P$ with private key S, G, P

Encrypt: $m \rightarrow mG' + e, wt(e) \leq t$

Decrypt:

1. Multiply $(mG' + e)P^{-1} = mSG + e'$ $wt(e) = wt(e')$
2. $mSG + e' \longrightarrow$ Fast decoding algorithm $\longrightarrow mSG$
3. Multiply on the right by G^{-1} , then by S^{-1} to recover m



Example

McEliece using (7,4) Hamming Code

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

<http://www-math.ucdenver.edu/~wcherowi/courses/m5410/ctcmcel.html>

Illustrate McEliece cryptosystem using (7,4) Hamming Code

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Secret scrambler and permutation matrices S, P chosen as

$$S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \text{ and } P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

<http://www-math.ucdenver.edu/~wcherowi/courses/m5410/ctcmcel.html>

Illustrate McEliece cryptosystem using (7,4) Hamming Code

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Secret scrambler and permutation matrices S, P chosen as

$$S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \text{ and } P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\text{Then the public generator matrix } G' = SG P = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

<http://www-math.ucdenver.edu/~wcherowi/courses/m5410/ctcmcel.html>

Encrypt

Suppose Alice wishes to send message $u = 1\ 1\ 0\ 1$ to Bob

<http://www-math.ucdenver.edu/~wcherowi/courses/m5410/ctcmcel.html>

Encrypt

Suppose Alice wishes to send message $u = 1\ 1\ 0\ 1$ to Bob

1. Alice constructs a weight 1 error vector, say $e = 0\ 0\ 0\ 0\ 1\ 0\ 0$
2. Alice computes $uG' + e = 0\ 1\ 1\ 0\ 0\ 1\ 0 + 0\ 0\ 0\ 0\ 1\ 0\ 0$

Encrypt

Suppose Alice wishes to send message $u = 1\ 1\ 0\ 1$ to Bob

1. Alice constructs a weight 1 error vector, say $e = 0\ 0\ 0\ 0\ 1\ 0\ 0$
2. Alice computes $uG' + e = 0\ 1\ 1\ 0\ 0\ 1\ 0 + 0\ 0\ 0\ 0\ 1\ 0\ 0$
 $= 0\ 1\ 1\ 0\ 1\ 1\ 0$

Alice sends ciphertext **0 1 1 0 1 1 0** to Bob

<http://www-math.ucdenver.edu/~wcherowi/courses/m5410/ctcmcel.html>

Decrypt

1. Bob multiplies the ciphertext on the right by P^{-1} : **0 1 1 0 1 1 0**

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Decrypt

1. Bob multiplies the ciphertext on the right by P^{-1} : **0 1 1 0 1 1 0**

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Bob takes the result 1 0 0 0 1 1 1 and uses fast decoding algorithm to remove the single bit of error
3. Bob takes the resulting codeword 1 0 0 0 1 1 0

Decrypt

1. Bob multiplies the ciphertext on the right by P^{-1} : **0 1 1 0 1 1 0**

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Bob takes the result 1 0 0 0 1 1 1 and uses fast decoding algorithm to remove the single bit of error
3. Bob takes the resulting codeword $y = 1\ 0\ 0\ 0\ 1\ 1\ 0$ which is of the form (xSG)

1. Recall that $G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ which contains I_4 in the first four columns!

Decrypt

1. Bob multiplies the ciphertext on the right by P^{-1} : **0 1 1 0 1 1 0**

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Bob takes the result 1 0 0 0 1 1 1 and uses fast decoding algorithm to remove the single bit of error
3. Bob takes the resulting codeword $y = 1\ 0\ 0\ 0\ 1\ 1\ 0$ which is of the form (xSG)

- Recall that $G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ which contains I_4 in the first four columns!

- Bob knows there exists some x' s.t. $x'G = x'$ $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \boxed{1\ 0\ 0\ 0} 1\ 1\ 0$

Decrypt

1. Bob multiplies the ciphertext on the right by P^{-1} : **0 1 1 0 1 1 0**

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Bob takes the result 1 0 0 0 1 1 1 and uses fast decoding algorithm to remove the single bit of error
3. Bob takes the resulting codeword $y = 1\ 0\ 0\ 0\ 1\ 1\ 0$ which is of the form (xSG)

- Recall that $G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ which contains I_4 in the first four columns!

- Bob knows there exists some x' s.t. $x'G = x'$ $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \boxed{1\ 0\ 0\ 0} 1\ 1\ 0$ Here $x' = xS$
- Therefore, Bob knows that $xS = 1\ 0\ 0\ 0$, so multiplying on the right by S^{-1} yields 1 1 0 1

McEliece cryptosystem

Idea: “hide” a message by converting it into a codeword, then adding as many errors as the code is capable of correcting

Underlying code: McEliece used Goppa codes

- Efficient decoding
- Scrambled public key $G' = SGP$ is indistinguishable from random codes
- Public key \approx a few megabits

McEliece cryptosystem

Idea: “hide” a message by converting it into a codeword, then adding as many errors as the code is capable of correcting

Underlying code: McEliece used Goppa codes

- Efficient decoding
- Scrambled public key $G' = SGP$ is indistinguishable from random codes
- Public key \approx a few megabits (2^{19})
 - Typical RSA key sizes are 1,024 or 2,048 or 4,096 bits
 - ECDH key sizes are roughly 256 or 512 bits

Trapdoor

NP-completeness of decoding problem does not indicate cryptographic security for concrete instances

Private key S, G, P turn out to be trapdoors ($G' = SGP$)

Encryption: $mG' + e$ easy to compute

Decryption difficult without S, G, P

Best known algorithm to solve decoding problems: **Information Set Decoding (Prange, 1962)**

Information Set Decoding

Information Set Decoding

Recall the basic McEliece cryptosystem:

Public key: G' scrambled generator matrix of a code C

Ciphertext: $c = mG' + e$

Objective: find a large set of error-free coordinates in a noisy codeword vector s.t. the corresponding columns of G' form an invertible submatrix

Definition: indices of the error-free coordinates are known as the information set I

- G' generator matrix of a code C
- I information set s.t. G'_I invertible
- m information word
- $c = mG' + e$ noisy codeword

Information Set Decoding

Outline of a message recovery attack:

Objective: find a large set of **error-free coordinates** in a noisy codeword vector s.t. the corresponding columns of G' form an invertible submatrix

- indices of the error-free coordinates are known as the **information set I**

- G' generator matrix of a code \mathcal{C}
- I information set s.t. G'_I invertible
- m information word
- $c = mG' + e$ noisy codeword

$$c_I = (mG' + e)_I = (mG')_I + e_I = (mG')_I$$

Recover m by:

$$c_I \cdot G_I'^{-1} = (mG')_I \cdot G_I'^{-1}$$

Example - ISD

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = \mathbf{0\ 1\ 1\ 0\ 1\ 1\ 0}.$$

Example - ISD

Let $G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$. Let $c = \mathbf{0110110}$.

1. Guess information set: first four bits of c are error-free

Example

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = \mathbf{0\ 1\ 1\ 0\ 1\ 1\ 0}.$$

1. Guess information set: first four bits of c are error-free ($I = \{0,1,2,3\}$)
2. Check: is G_I invertible?

Example

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = 0110110.$$

1. Guess information set: first four bits of c are error-free
2. Check: is G_I invertible?



Example

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = \mathbf{0\ 1\ 1\ 0\ 1\ 1\ 0}.$$

1. Guess information set: $I = \{1,2,3,4\}$ are error-free
2. Check: is G_I invertible?

Example

Let $G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$. Let $c = 0\ 1\ 1\ 0\ 1\ 1\ 0$.

1. Guess information set: $I = \{1,2,3,4\}$ are error-free

2. Check: is G_I invertible?



3. Compute guess of m

$$G_I'^{-1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Example

Let $G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$. Let $c = \mathbf{0\ 1\ 1\ 0\ 1\ 1\ 0}$.

1. Guess information set: $I = \{1,2,3,4\}$ are error-free

2. Check: is G_I invertible?



3. Compute guess of m

$$G_I'^{-1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$c_I \cdot G_I'^{-1} = [1\ 1\ 0\ 1] \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Example

"Suppose Alice wishes to send message $u = 1\ 1\ 0\ 1$ to Bob"

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = \mathbf{0\ 1\ 1\ 0\ 1\ 1\ 0}.$$

1. Guess information set: $I = \{1,2,3,4\}$ are error-free

2. Check: is G_I invertible?



$$G_I'^{-1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

3. Compute guess of m

$$c_I \cdot G_I'^{-1} = [1\ 1\ 0\ 1] \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = [1\ 0\ 1\ 0]$$



Example


Let $G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$. Let $c = \mathbf{0\ 1\ 1\ 0\ 1\ 1\ 0}$.

1. Guess information set: first two and last two bits of c are error-free ($I = \{0,1,5,6\}$)
2. Check: is G_I invertible?

Example

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = 0110110.$$

1. Guess information set: ($I = \{1,2,6,7\}$)

2. Check: is G_I invertible? 


$$G_I'^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Example

"Suppose Alice wishes to send message $u = 1\ 1\ 0\ 1$ to Bob"

$$\text{Let } G' = SGP = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \text{ Let } c = 0\ 1\ 1\ 0\ 1\ 1\ 0.$$

1. Guess information set: $(I = \{1,2,6,7\})$

2. Check: is G_I invertible? 

$$G_I'^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$c_I \cdot G_I'^{-1} = [1\ 1\ 1\ 0] \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [1\ 1\ 0\ 1]$$

How to reduce McEliece key sizes?

McEliece's cryptosystem is secure, but not very practical for applications.

Use another code:

- Generalized Reed-Solomon codes
- Subcodes of GRS
- Reed-Muller Codes
- Algebraic geometry codes
- Low density parity check code – a linear code for which there exists a sparse parity check matrix (Hamming weight of rows in $O(1)$)
- Moderate density parity check code – a linear code for which there exists a moderately sparse parity check matrix (Hamming weight of rows in $O(\sqrt{n})$)

How to reduce McEliece key sizes?

McEliece's cryptosystem is secure, but not very practical for applications.

Use another code:

- Generalized Reed-Solomon codes → Proposed in 1986, Broken in 1992
- Subcodes of GRS → Proposed in 2005, Broken in 2010
- Reed-Muller Codes → Proposed in 1994, Broken in 2007 and 2013
- Algebraic geometry codes → Proposed in 1996, Broken in 2008, 2014
- Low density parity check code – a linear code for which there exists a sparse parity check matrix (Hamming weight of rows in $O(1)$)
- Moderate density parity check code – a linear code for which there exists a moderately sparse parity check matrix (Hamming weight of rows in $O(\sqrt{n})$)

Quasi-cyclic codes

Quasi-cyclic structure

- A $v \times v$ circulant matrix has the form:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & \cdots & a_{v-1} \\ a_{v-1} & a_0 & a_1 & a_2 & \cdots & a_{v-2} \\ a_{v-2} & a_{v-1} & a_0 & a_1 & \cdots & a_{v-3} \\ a_{v-3} & a_{v-2} & a_{v-1} & a_0 & \cdots & a_{v-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & a_4 & \cdots & a_0 \end{bmatrix}$$

- A Quasi-cyclic (QC) matrix is a matrix of circulant blocks
- Blocks of size $p \times p$ for p prime and 2 is primitive mod p to ensure no further decomposition possible
- $w(A)$ the number of nonzero coefficients of a polynomial

Ring of $v \times v$ circulant matrices over \mathbb{F}_2 isomorphic to $\mathbb{F}_2[x]/\langle x^v + 1 \rangle$.

$$a_0 + a_1x + a_2x^2 \bmod x^3 + 1$$

$$\begin{array}{c} \updownarrow \\ \begin{bmatrix} a_0 & a_1 & a_2 \\ a_2 & a_0 & a_1 \\ a_1 & a_2 & a_0 \end{bmatrix} \end{array}$$

BIKE at a high level

- Based on binary, linear codes with
 - Quasi-cyclic structure – private key composed of two circulant blocks H_0, H_1
 - Moderately-dense parity check matrices

For context

NIST Security Category 1, taken from algorithm specifications

BIKE - AVX512

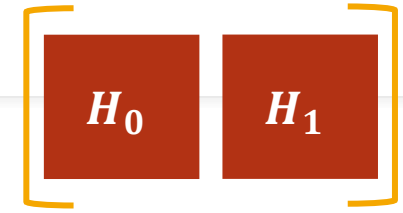
HQC AVX2 optimized

Classic McEliece AVX

ML-KEM performance included for comparison

	Public key size (bytes)	Private key (bytes)	Ciphertext size (bytes)	KeyGen (kilocycles)	Encaps (kilocycles)	Decaps (kilocycles)
BIKE	1,540	2,801	1,572	589	97	1,135
HQC	2,249	56	4,497	87	204	362
mceliece348864f	261,120	6,492	96	35,978	38	128
Kyber-512	800	32	768	123	155	289

BIKE at a high level



- Based on binary, linear codes with
 - Quasi-cyclic structure - private key composed of two circulant blocks H_0, H_1
 - Moderately-dense parity check matrices
- Let r denote circulant block length. Let t denote max. error weight
 - Secret key $H \in \mathbb{F}_2^{r \times 2r}$ is of the form $H = [H_0 | H_1]$
 - Public key $H' = H_0^{-1}(H)$
 - Message encoded as error-vector $e \in \mathbb{F}_2^{2r}$ of weight t
 - Ciphertext is syndrome $s = He^T \in \mathbb{F}_2^r$. Decrypt using syndrome decoder

BIKE At a high level

Parameters

- r : block length
- w : row weight of secret key
- t : maximum error weight
- λ : security parameter

Design principles

- r prime
- $x^r - 1$ has only two irreducible factors modulo 2
- $w \in \mathcal{O}(\sqrt{n})$
- $w = 2d, d$ odd
- $\lambda \approx t - \frac{1}{2} \log_2 r \approx w - \frac{1}{2} \log_2 r$

Syndrome decoding

- Ciphertext is syndrome $s = He^T \in \mathbb{F}_2^r$. Decrypt using syndrome decoder

Parity checks

For parity check matrix $H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ and error-free bit-string of 7 bits $b = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$, these three equations are satisfied

$$\begin{aligned} b_0 + b_1 + b_3 + b_4 &= 0 \\ b_0 + b_2 + b_3 + b_5 &= 0 \\ b_1 + b_2 + b_3 + b_6 &= 0 \end{aligned}$$

Parity checks

For parity check matrix $H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ and error-free bit-string of 7 bits $b = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$, these three equations are satisfied

$$\begin{aligned} b_0 + b_1 + b_3 + b_4 &= 0 \\ b_0 + b_2 + b_3 + b_5 &= 0 \\ b_1 + b_2 + b_3 + b_6 &= 0 \end{aligned}$$

If one error occurred, at least one equation will not hold.

Parity checks

For parity check matrix $H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ and error-free bit-string of 7 bits $b = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$, these three equations are satisfied

$$\begin{aligned} b_0 + b_1 + b_3 + b_4 &= 0 \\ b_0 + b_2 + b_3 + b_5 &= 0 \\ b_1 + b_2 + b_3 + b_6 &= 0 \end{aligned}$$

If one error occurred, at least one equation will not hold.

Parity checks

For parity check matrix $H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ and $b = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$, let syndrome $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$b_0 + b_1 + \quad \quad \quad b_3 + b_4 \quad \quad \quad = 1$$

$$b_0 + \quad \quad \quad b_2 + b_3 + \quad \quad \quad b_5 \quad \quad \quad = 0$$

$$\quad \quad \quad b_1 + b_2 + b_3 + \quad \quad \quad \quad \quad \quad b_6 = 1$$

Which bit was corrupted?

Parity checks

For parity check matrix $H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ and $b =$

$(b_0, b_1, b_2, b_3, b_4, b_5, b_6)$, let syndrome $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$\begin{array}{rcl} b_0 + b_1 + & b_3 + b_4 & = 1 \\ b_0 + & b_2 + b_3 + & b_5 = 0 \\ & b_1 + b_2 + b_3 + & b_6 = 1 \end{array}$$

Which bit was corrupted?

Bit-flipping decoder

Algorithm 6.2: Parallel bit-flipping.

function parallel_bitflip[⊥](**H**, **s**):

input : A sparse parity check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$,
a syndrome $\mathbf{s} = \mathbf{H}\mathbf{e}^\top \in \mathbb{F}_2^{n-k}$.

output: An error pattern $\mathbf{e}' \in \mathbb{F}_2^n$ such that $\mathbf{H}\mathbf{e}'^\top = \mathbf{s}$.

$\mathbf{e}' \leftarrow \mathbf{0}$;

$\mathbf{s}' \leftarrow \mathbf{s}$;

$T \leftarrow \text{threshold}(\text{context})$;

while $\mathbf{s}' \neq \mathbf{0}$ **do**

for $j \in \{0, \dots, n-1\}$ **do**

if $|\mathbf{h}_j \star \mathbf{s}'| \geq T$ **then**

$\mathbf{e}'_j \leftarrow 1 - \mathbf{e}'_j$;

$\mathbf{s}' \leftarrow \mathbf{s} - \mathbf{H}\mathbf{e}'^\top$;

return \mathbf{e}' ;

Bit-flip decoding

Algorithm 6.2

Algorithm 6.2 takes as input a parity check matrix H and a syndrome s
Outputs an error pattern e' such that $He' = s$

Let $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ secret $e = [0 \ 0 \ 0 \ 1 \ 0 \ 0]$.

Bit-flip decoding

Algorithm 6.2

Algorithm 6.2 takes as input a parity check matrix H and a syndrome s
Outputs an error pattern e' such that $He' = s$

Let $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ secret $e = [0 \ 0 \ 0 \ 1 \ 0 \ 0]$.

NOTE

- H is QC-MDPC
- $n = 6$
- Row weight $w = 3 \approx \sqrt{6}$
- Using parameter relationship $\frac{w}{2}t \approx r$, we expect to be able to decode up to $t \approx \frac{2r}{w} = 2$ errors
- I'm using $t = 1$ for the sake of needing fewer iterations of the decoder

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round1:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round₁:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$|h_0 \star s'| = \left| \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right| = 1$$

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round1:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$|h_0 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |0| \end{vmatrix} = 1$$

$$|h_1 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |0| \end{vmatrix} = 0$$

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round1:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$|h_0 \star s'| = \begin{bmatrix} |1| \\ |0| \\ |0| \end{bmatrix} = 1$$

$$|h_1 \star s'| = \begin{bmatrix} |0| \\ |0| \\ |0| \end{bmatrix} = 0$$

$$|h_2 \star s'| = \begin{bmatrix} |0| \\ |0| \\ |1| \end{bmatrix} = 1$$

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round1:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$|h_0 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |0| \end{vmatrix} = 1$$

$$|h_1 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |0| \end{vmatrix} = 0$$

$$|h_2 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |1| \end{vmatrix} = 1$$

$$|h_3 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |1| \end{vmatrix} = 2$$

$$|h_4 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |0| \end{vmatrix} = 1$$

$$|h_5 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |1| \end{vmatrix} = 1$$

Bit-flip decoding

Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round1:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

$$|h_0 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |0| \end{vmatrix} = 1$$

$$|h_1 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |0| \end{vmatrix} = 0$$

$$|h_2 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |1| \end{vmatrix} = 1$$

$$|h_3 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |1| \end{vmatrix} = 2 \geq T$$

$$|h_4 \star s'| = \begin{vmatrix} |1| \\ |0| \\ |0| \end{vmatrix} = 1$$

$$|h_5 \star s'| = \begin{vmatrix} |0| \\ |0| \\ |1| \end{vmatrix} = 1$$

Since $|h_3 \star s'| \geq T$, we must flip bit e'_3

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round₁:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

Since $|h_3 \star s'| \geq T$, we must flip bit e'_3

Update $e' = [0 \ 0 \ 0 \ 1 \ 0 \ 0]$

Compute $s - He'^T = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round₁:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

Since $|h_3 \star s'| \geq T$, we must flip bit e'_3

Update $e' = [0 \ 0 \ 0 \ 1 \ 0 \ 0]$

Compute $s - He'^T =$

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round₁:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

Since $|h_3 \star s'| \geq T$, we must flip bit e'_3

Update $e' = [0 \ 0 \ 0 \ 1 \ 0 \ 0]$

Compute $s - He'^T = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Bit-flip decoding Algorithm 6.2

Input: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $s = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Output: e' such that $He'^T = s$

Round1:

Initialize $e' = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$, $T = 2$, $s' = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.

Since $|h_3 * s'| \geq T$, we must flip bit e'_4

Update $e' = [0 \ 0 \ 0 \ 1 \ 0 \ 0]$

$$\begin{aligned} \text{Update } s' &= s - He'^T = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

The while loop
terminates once
 $s' = \mathbf{0}$ outputs e'

Security of BIKE

IND-CCA security claim

- Decoding failure rates (DFR) must be sufficiently low to achieve IND-CCA security and to prevent GJS key recovery attack

Analyzing BIKE DFR

- Direct computation for $\lambda \geq 128$ infeasible by design
- Compute average DFR directly for smaller λ then use extrapolation techniques to approximate larger block sizes.

BIKE: Bit flipping key encapsulation - <https://bikesuite.org>

Guo, Johansson, and Stankovski. A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors (2016)

Jean-Pierre Tillich. The decoding failure probability of MDPC codes (2018)

Nir Drucker, Shay Gueron, and Dusan Kostic. On constant-time QC-MDPC decoders with negligible failure rate (2020)

Nir Drucker, Shay Gueron, and Dusan Kostic. QC-MDPC decoders with several shades of gray (2020)

Nicolas Sendrier and Valentin Vasseur. On the decoding failure rate of QC-MDPC bit-flipping decoders (2019)

DFR of Round 2-4 Code-based KEMs

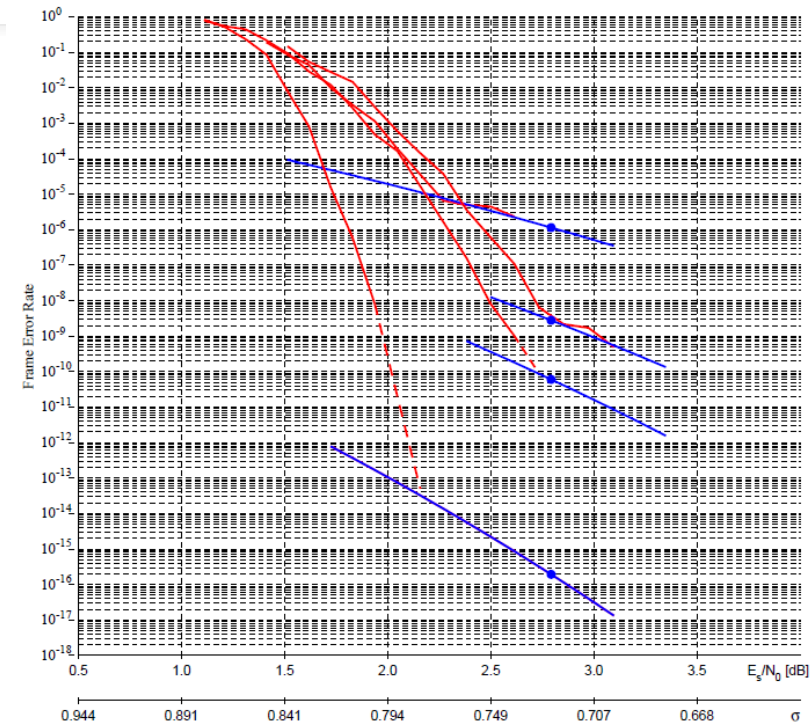
The probability of decoding failure (DFR) for any code-based cryptosystem which allows key-reuse decoder **must satisfy**

Code-based cryptosystem	DFR	DFR justification
Classic McEliece	0	Goppa codes do not have decoding failures
HQC	$< \frac{1}{2^\lambda}$	Proof
BIKE	$< \frac{1}{2^\lambda}$	Experiments, extrapolation, analysis, no closed-form model of the decoder

DFR Curve

The graphs of DFRs on a log scale for low-moderate density parity check codes with iterative decoders display phenomenon

- Initial, rapid decrease of decoding failures: *waterfall region*
- Eventual plateau, more linear decrease: *error floor region*



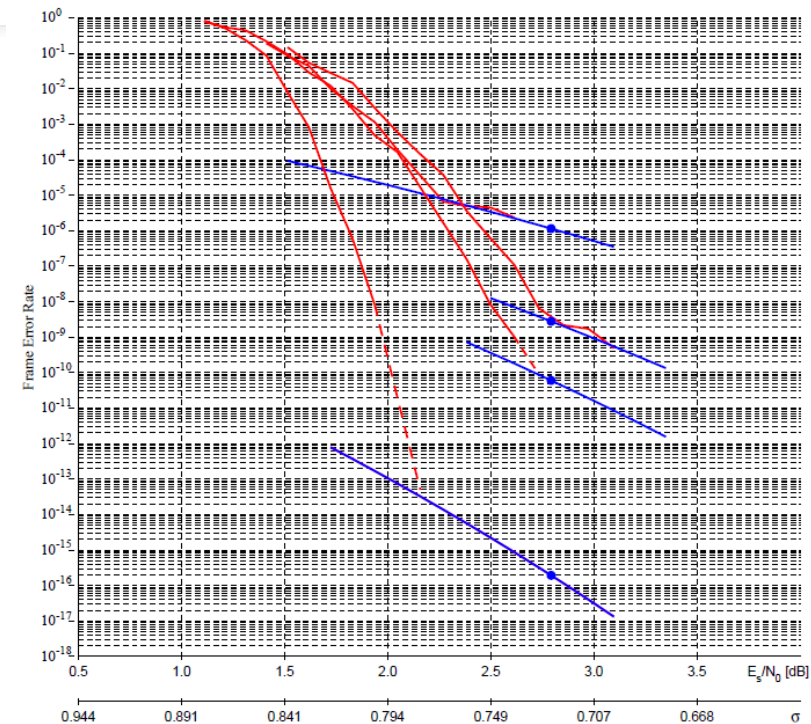
Bane Vasić, Shashi Kiran Chilappagari, and Dung Viet Nguyen. Failures and Error Floors of Iterative Decoders (2014)
Tom Richardson. Error floors of LDPC codes (2003)

DFR Curve

The graphs of DFRs on a log scale for low-moderate density parity check codes with iterative decoders display phenomenon

- Initial, rapid decrease of decoding failures: *waterfall region*
- Eventual plateau, more linear decrease: *error floor region*

To accurately predict BIKE DFR for larger code length (signal-to-noise ratio), one must consider the error floor behavior

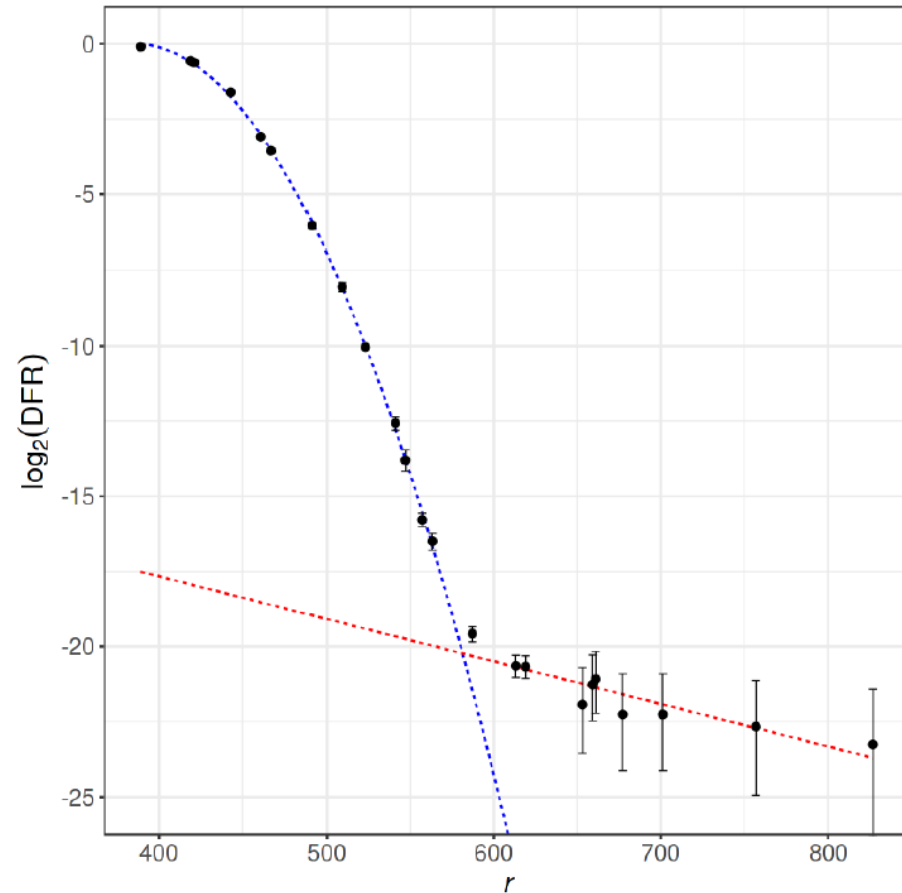


Bane Vasić, Shashi Kiran Chilappagari, and Dung Viet Nguyen. Failures and Error Floors of Iterative Decoders (2014)
Tom Richardson. Error floors of LDPC codes (2003)

Scaling down BIKE

To better understand the error floor behavior of BIKE DFR curves, we experimentally consider BIKE at the 20-bit security level

1. Use BIKE design parameters to generate parameter sets for $\lambda = 20$
2. Use Boston University Shared Computing Cluster to run highly parallelizable experiments
3. Examine factors that increase decoding failures, affecting the error floor



BIKE DFR Curve for $\lambda = 20$

Initial Results

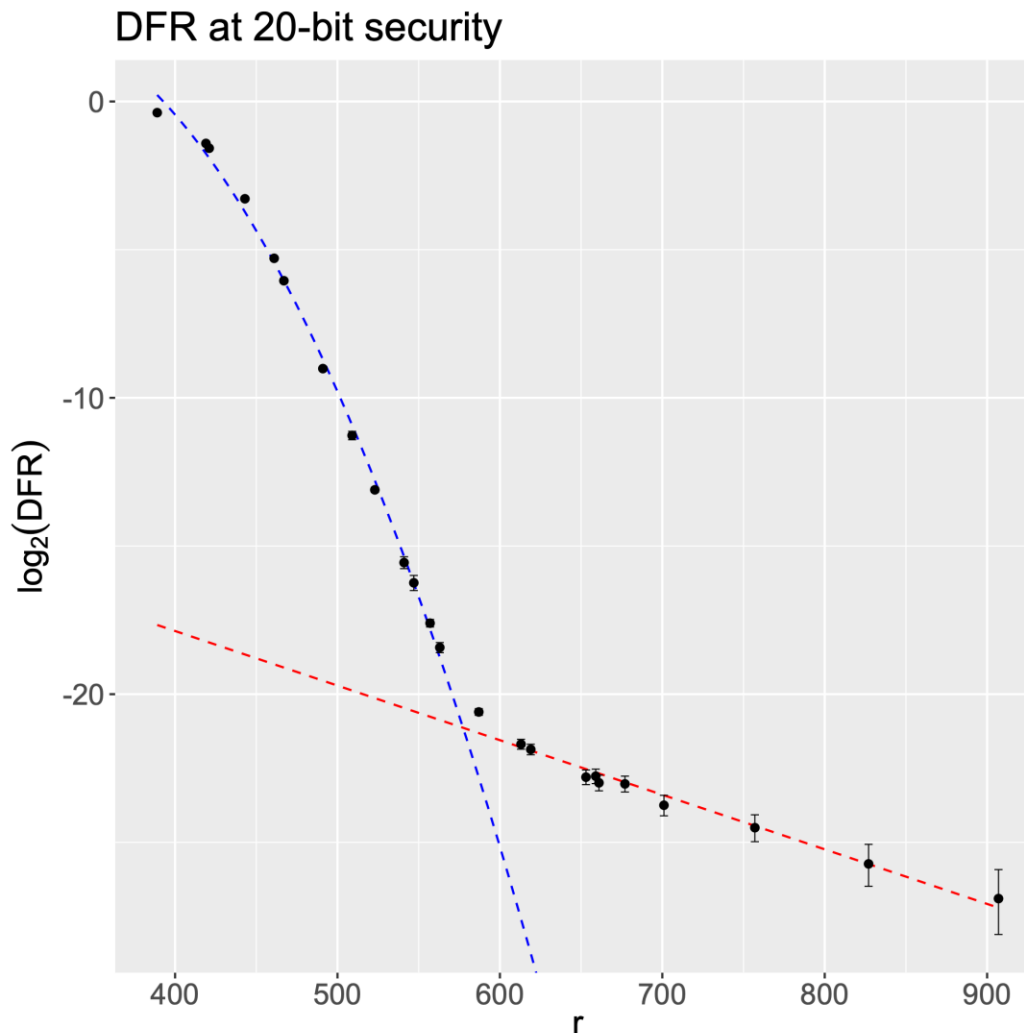
Error floor region begins $r \approx 587$

95% confidence intervals shown

Number of trials vary from 10^4 to 10^{10}

- The set of weak keys is more dense for smaller r making non-weak key sampling more time-consuming
- Finding decoding failures takes longer for larger r

Raw data available at <https://github.com/HastD/BIKE-error-floor>



BIKE DFR Curve for $\lambda = 20$

Error floor studies in LDPC codes

Definition: Let $H_{n-k \times n}$ be a parity-check matrix describing a code $\mathcal{C}(n, k)$. A (\mathbf{u}, \mathbf{v}) *near-codeword* is an error vector \mathbf{e} of weight \mathbf{u} whose syndrome $\mathbf{s} = H\mathbf{e}^T$ has weight \mathbf{v} .

McKay, Postol (2003): near codewords with small \mathbf{u}, \mathbf{v} and low-weight codewords cause high error floor for certain LDPC codes

Marco Baldi. QC-LDPC Code-Based Cryptography (2014)

David J.C. MacKay and Michael S. Postol. Weaknesses of Margulis and Ramanujan-Margulis Low-Density Parity-Check Codes (2003)

Tom Richardson. Error floors of LDPC codes (2003)

Gerd Richter. Finding small stopping sets in the Tanner graphs of LDPC codes (2006)

Error Floor Prediction with Markov Models for QC-MDPC Codes

Joint work with:

Sarah Arpin, Jun Bo Lau, Antoine Mesnard, Ray Perlner, Jean-Pierre Tillich, Valentin Vasseur

Overall DFR – Testing the models

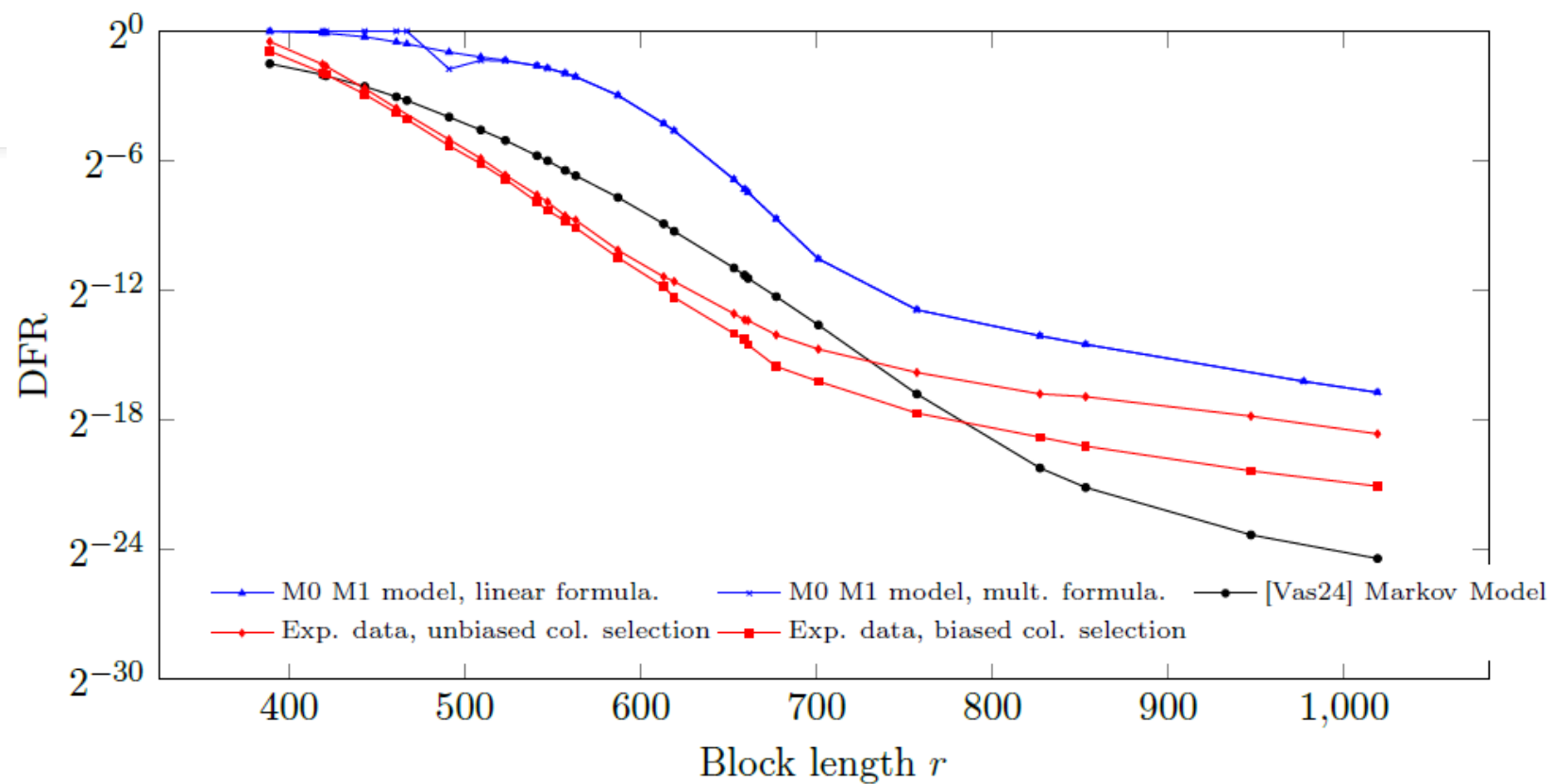


FIGURE 7. Plot of experimental DFR data, our linear and multiplicative Markov model DFRs, and the DFR of the Markov model in described in [SV19b].

Final Model

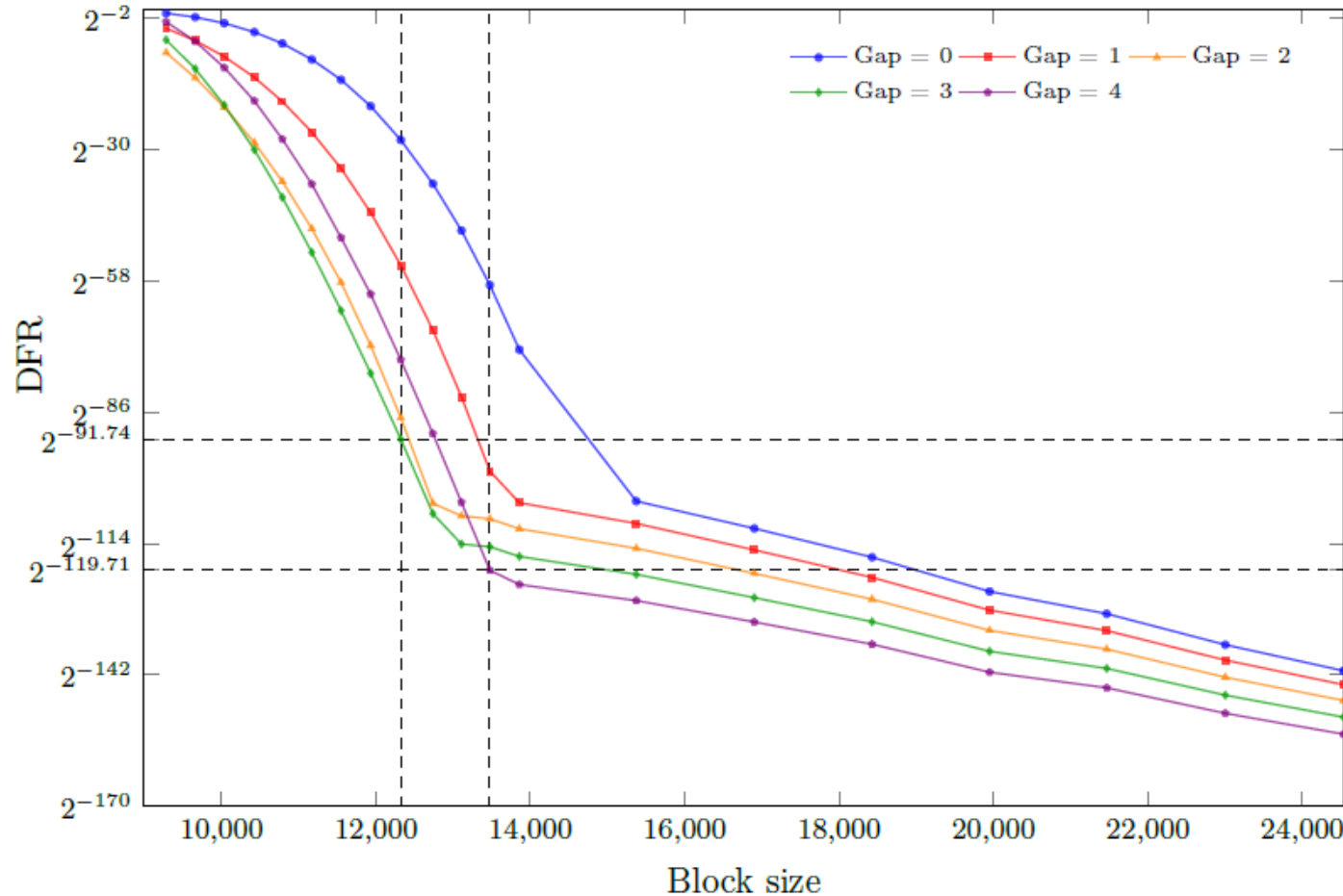


FIGURE 14. DFR vs. block size ($d = 71$, $t = 134$).

$$T = \min \left(d, \max \left(A_{r,t}(s) + \delta, \frac{d+1}{2} \right) \right)$$

We use the model to estimate the DFR of BIKE under different threshold T rules:

- Under BIKE level 1 parameters we predict DFR 2^{-91}
- Predicts DFR $2^{-119.71}$ for block size r-13477
- Using SBS decoder
- Expect parallel decoders to have far fewer failures
- With slight modification to the Threshold function for BIKE and block size, achieve DFR $2^{-129.5}$

<https://eprint.iacr.org/2025/153>

Thank you!