Code-based Cryptography

a Hands-On Introduction

Daniel Loebenberger
<daniel_loebenberger@genua.de>

Ηράκλειο, September 27, 2018

Post-Quantum Cryptography

Various flavours:

- Lattice-based cryptography
- Hash-based cryptography
- Code-based cryptography
- Further techniques (e.g. multivariate, isogeny-based, ...)

Post-Quantum Cryptography

Various flavours:

- Lattice-based cryptography
- Hash-based cryptography
- Code-based cryptography
- Further techniques (e.g. multivariate, isogeny-based, ...)

Coding theory

- Dating back to Claude Shannon in 1948
- Goal is to protect a message sent over a noisy channel
- Typically, this is achieved by adding redundancy

$$m \xrightarrow{\text{encode}} c \xrightarrow{\text{+error e}} x = c + e \xrightarrow{\text{decode}} c' \xrightarrow{\text{recover}} m'$$

Public key Cryptography

We need to specify three algorithms:

- Key Generation: Create a private and a public key
- Encryption of a message *using the public key*
- Decryption of a ciphertext *using the private key*

Security: It is not possible to decrypt a ciphertext without the private key.

Content

- Linear Codes
- Classic McEliece
- Optimizations
- Conclusion

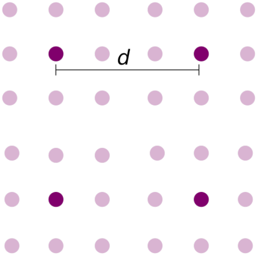
Content

- Linear Codes
- Classic McEliece
- Optimizations
- Conclusion

Basic Definitions

Definition (Linear Code)

A *linear* (n, k, d)*-code* $\mathcal{C}$ over a finite field $\mathbb{F}$ is a k-dimensional subspace of the vector space $\mathbb{F}^n$ with *minimum distance* $d = \min_{x \neq y \in \mathcal{C}} \operatorname{dist}(x, y)$, where dist is the *Hamming-distance*.



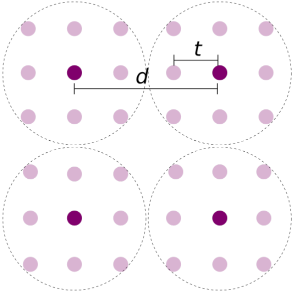
Basic Definitions

Definition (Linear Code)

A *linear* (n, k, d)*-code* $\mathcal{C}$ over a finite field $\mathbb{F}$ is a k-dimensional subspace of the vector space $\mathbb{F}^n$ with *minimum distance* $d = \min_{x \neq y \in \mathcal{C}} \mathrm{dist}(x, y)$, where dist is the *Hamming-distance*.

Theorem

A linear (n, k, d)*-code can correct up to* $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ *errors.*



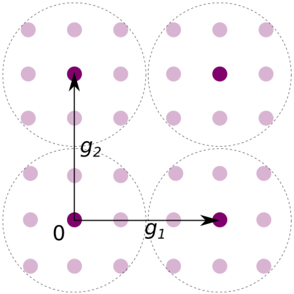
Basic Definitions

Definition (Generator Matrix)

The matrix $G \in \mathbb{F}^{k \times n}$ is a *generator matrix* for the (n, k, d)-code $\mathcal{C}$ if $\mathcal{C} = \langle G \rangle$, i.e. the rows of G span $\mathcal{C}$.

Definition (Encoding)

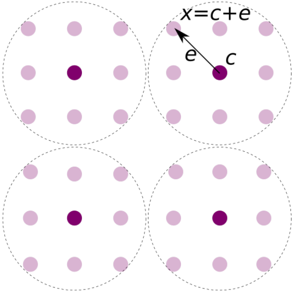
For a message $m \in \mathbb{F}^k$, define its *encoding* as $c = mG \in \mathbb{F}^n$.



Basic Definitions

Problem (Decoding problem)

Given $x \in \mathbb{F}^n$ *find* $c \in \mathcal{C}$, *where* $\mathrm{dist}(x, c)$ *is minimal. If* $x = c + e$ *and* e *is a vector of weight at most* t *then* x *is uniquely determined.*



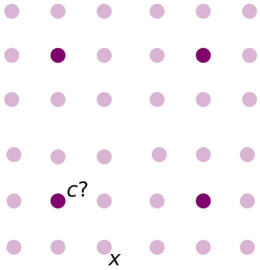
Basic Definitions

Problem (Decoding problem)

Given $x \in \mathbb{F}^n$ *find* $c \in \mathcal{C}$, *where* $\mathrm{dist}(x, c)$ *is minimal. If* $x = c + e$ *and* e *is a vector of weight at most* t *then* x *is uniquely determined.*

Theorem

The (general) decoding problem is $\mathcal{NP}$*-hard.*



Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

$$m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG =$$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

$$m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$cH^t =$$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

$$m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$cH^t = 0$, i.e. $c \in \mathcal{C}$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

$$m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$cH^t = 0, \text{ i.e. } c \in \mathcal{C}$$

One-bit error $e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow x = c + e =$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

$$m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$cH^t = 0, \text{ i.e. } c \in \mathcal{C}$$

One-bit error $e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow x = c + e = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$

Encoding/Decoding: Example

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

$G \in \mathbb{F}_2^{4\times 7}$ generator matrix for code $\mathcal{C}$, $H \in \mathbb{F}_2^{3\times 7}$ *parity check matrix*: $GH^t = 0$

$$m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$cH^t = 0, \text{ i.e. } c \in \mathcal{C}$$

One-bit error $e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow x = c + e = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$

Then: $xH^t = cH^t + eH^t = eH^t = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$

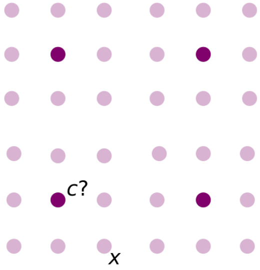
Decoding idea: run over all possible errors e, compute eH^t and compare to xH^t.

Content

- Linear Codes
- Classic McEliece
- Optimizations
- Conclusion

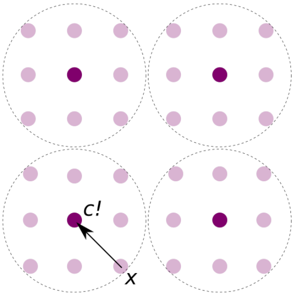
Basic idea

- The general decoding problem is a hard problem



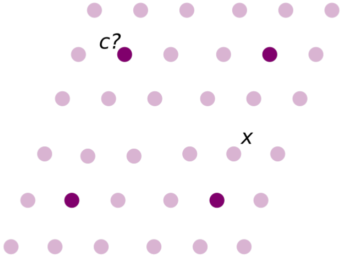
Basic idea

- The general decoding problem is a hard problem
- However, there are certain codes with efficient decoding
- One example are binary Goppa codes



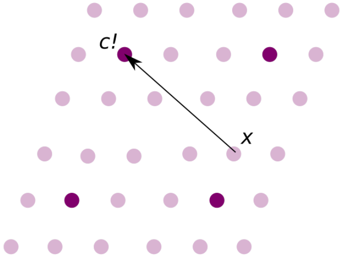
Basic idea

- The general decoding problem is a hard problem
- However, there are certain codes with efficient decoding
- One example are binary Goppa codes
- "Disguise" such a code



Basic idea

- The general decoding problem is a hard problem
- However, there are certain codes with efficient decoding
- One example are binary Goppa codes
- “Disguise” such a code
- Use this information as a trapdoor!



Specification (I)

Key Generation

- Secret generator matrix $G \in \mathbb{F}_2^{k \times n}$ of an easily decodeable (n, k, d)-code $\mathcal{C} = \langle G \rangle$
- Secret random invertible matrix $S \in \mathbb{F}_2^{k \times k}$
- Secret random permutation matrix $P \in \mathbb{F}_2^{n \times n}$

Compute public $G' = SGP$.

Specification (I)

Key Generation

- Secret generator matrix $G \in \mathbb{F}_2^{k \times n}$ of an easily decodeable (n, k, d)-code $\mathcal{C} = \langle G \rangle$
- Secret random invertible matrix $S \in \mathbb{F}_2^{k \times k}$
- Secret random permutation matrix $P \in \mathbb{F}_2^{n \times n}$

Compute public $G' = SGP$.

Heuristically, the matrix G' behaves like a uniformly selected matrix,
i.e. $\mathcal{C}' = \langle G' \rangle$ is hard to decode.

Specification (II)

Encryption of $m \in \mathbb{F}_2^k$

- Choose $e \in \mathbb{F}_2^n$ of weight t
- return $x = mG' + e$

Specification (II)

Encryption of $m \in \mathbb{F}_2^k$

- Choose $e \in \mathbb{F}_2^n$ of weight t
- return $x = mG' + e$

Decryption of $x \in \mathbb{F}_2^n$

- Compute $y = xP^{-1} = (mS)G + eP^{-1}$
- Decode y, obtaining $(mS)G$
- Recover $m' = mS$
- return $m = m'S^{-1}$

Specification (II)

Encryption of $m \in \mathbb{F}_2^k$

- Choose $e \in \mathbb{F}_2^n$ of weight t
- return $x = mG' + e$

Decryption of $x \in \mathbb{F}_2^n$

- Compute $y = xP^{-1} = (mS)G + eP^{-1}$
- Decode y, obtaining $(mS)G$
- Recover $m' = mS$
- return $m = m'S^{-1}$

Both operations are comparatively efficient!

Encryption: Example

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

Encryption: Example

$$\text{Public } G' = SGP = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Encryption: Example

$$\text{Public } G' = SGP = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\text{Message } m = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow c = mG' = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\text{One-bit error } e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow x = c + e = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Decryption: Example

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

Decryption: Example

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad S^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad P^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Decryption: Example

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad S^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad P^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Receive $x = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$, compute $y = xP^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$

Decryption: Example

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad S^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad P^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Receive $x = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$, compute $y = xP^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$

Use decoding algorithm for G, giving message $m' = mS = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$

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

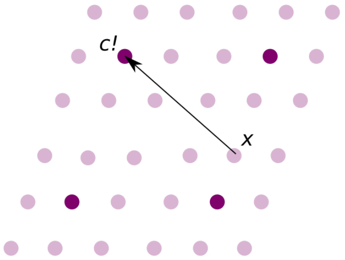
Security (classical)

McEliece Problem

Given a McEliece public key $G' \in \mathbb{F}_2^{k \times n}$ and a ciphertext $x \in \mathbb{F}_2^n$, find (the unique) $m \in \mathbb{F}_2^k$, s.t. $\mathrm{dist}(mG', x) = t$

Fact

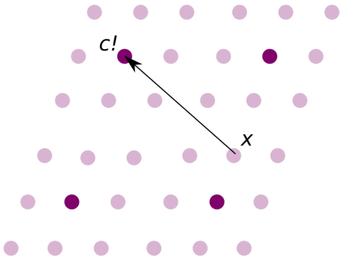
If you can solve the general decoding problem, then you can solve the McEliece problem.



Security (classical)

McEliece Problem

Given a McEliece public key $G' \in \mathbb{F}_2^{k \times n}$ and a ciphertext $x \in \mathbb{F}_2^n$, find (the unique) $m \in \mathbb{F}_2^k$, s.t. $\mathrm{dist}(mG', x) = t$



Fact

If you can solve the general decoding problem, then you can solve the McEliece problem.

However, the converse is not true, since $\mathcal{C}' = \langle G' \rangle$ is *not* a random code, but a disguised binary Goppa code!

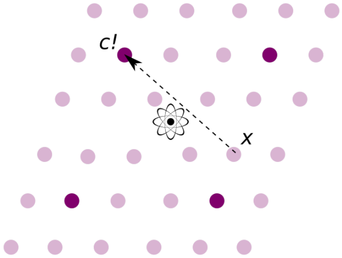
Security (post-quantum)

Grover's algorithm

A quantum-computer can search an unordered set of size l in time $\mathcal{O}(\sqrt{l})$.

Theorem

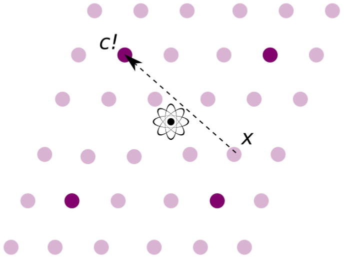
One can apply Grover's algorithm to solve the general decoding problem. This gives (roughly) a quadratic speedup.



Security (post-quantum)

Grover's algorithm

A quantum-computer can search an unordered set of size l in time $\mathcal{O}(\sqrt{l})$.



Theorem

One can apply Grover's algorithm to solve the general decoding problem. This gives (roughly) a quadratic speedup.

Runtime still exponential! Wide believe: This is all of the speedup a quantum-computer provides.

Parameter Sets

From the NIST proposal by Bernstein et al., Nov 2017:

`kem/mceliece6960119`

$k = 5413, n = 6960, t = 119$

`kem/mceliece8192128`

$k = 6528, n = 8192, t = 128$

Parameter Sets

From the NIST proposal by Bernstein et al., Nov 2017:

`kem/mceliece6960119`

$k = 5413, n = 6960, t = 119$

Approximate parameter sizes:
Plaintext: 677B, ciphertext: 870B
Public key: 4.5MB, secret key: 13.75MB

`kem/mceliece8192128`

$k = 6528, n = 8192, t = 128$

Approximate parameter sizes:
Plaintext: 870B, ciphertext: 1024B
Public key: 6.4MB, secret key: 19.45MB

Parameter Sets

From the NIST proposal by Bernstein et al., Nov 2017:

`kem/mceliece6960119`
$k = 5413, n = 6960, t = 119$

Approximate parameter sizes:
Plaintext: 677B, ciphertext: 870B
Public key: 4.5MB, secret key: 13.75MB

`kem/mceliece8192128`
$k = 6528, n = 8192, t = 128$

Approximate parameter sizes:
Plaintext: 870B, ciphertext: 1024B
Public key: 6.4MB, secret key: 19.45MB

Expectedly, both parameter sets fulfill the NIST requirements for an IND-CCA2 KEM, category 5, i.e. a security level of 256 bit.

McEliece: Conclusion

The security of the McEliece cryptosystem is convincing.
It comes, however, at the cost of large key-sizes.

Content

- Linear Codes
- Classic McEliece
- Optimizations
- Conclusion

Reducing the Key-Sizes

Opimization possibilities not affecting security:

- If $k > n-k$, rewrite using the parity check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$

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

Reducing the Key-Sizes

Opimization possibilities not affecting security:

- If $k > n - k$, rewrite using the parity check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$

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

- Store the permutation P in tuple representation!

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow P = (7, 2, 6, 3, 1, 5, 4)$$

Reducing the Key-Sizes: Effect

`kem/mceliece6960119`

$k = 5413, n = 6960, t = 119$

Public key: 4.5MB $\Rightarrow$ 1.3MB
Secret key: 13.75MB $\Rightarrow$ 4.8MB

`kem/mceliece8192128`

$k = 6528, n = 8192, t = 128$

Public key: 6.4MB $\Rightarrow$ 1.6MB
Secret key: 19.45MB $\Rightarrow$ 6.7MB

Reducing the Key-Sizes: Effect

`kem/mceliece6960119`

$k = 5413, n = 6960, t = 119$

Public key: 4.5MB $\Rightarrow$ 1.3MB
Secret key: 13.75MB $\Rightarrow$ 4.8MB

`kem/mceliece8192128`

$k = 6528, n = 8192, t = 128$

Public key: 6.4MB $\Rightarrow$ 1.6MB
Secret key: 19.45MB $\Rightarrow$ 6.7MB

Even more reductions possible, c.f. NIST submission on classic McEliece!

The Code-Based NIST Submissions

- BIG QUAKE
- BIKE
- Classic McEliece
- DAGS
- Edon-K (withdrawn)
- HQC
- LAKE
- LEDAkem
- LEDApkc
- Lepton
- LOCKER
- McNie
- NTS-KEM
- Ouroboros-R
- pqsigRM
- QC-MDPC KEM
- RaCoSS
- Ramstake;
- RankSign (withdrawn)
- RLCE-KEM
- RQC

The Code-Based NIST Submissions

- QC-MDPC codes
- Binary Goppa codes
- Quasi-Cyclic codes
- BCH codes
- Rank Metric codes
- Rank Quasi-Cyclic codes
- Random Linear codes
- QC-LDPC codes
- Quasi-Dyadic Gen. Srivastava codes
- LRPC codes
- Ideal-LRPC codes
- Punctured Reed-Muller codes
- Quasi-cyclic Goppa codes

Security Considerations

- Several good proposals
- Most aim on the goal of having much smaller key-sizes
- Security based on the problem of decoding special codes
- Further cryptanalysis necessary!

Security Considerations

- Several good proposals
- Most aim on the goal of having much smaller key-sizes
- Security based on the problem of decoding special codes
- Further cryptanalysis necessary!

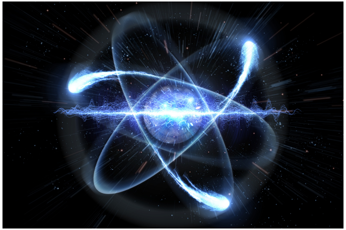
We live in exciting times :-)

Content

- Linear Codes
- Classic McEliece
- Optimizations
- Conclusion

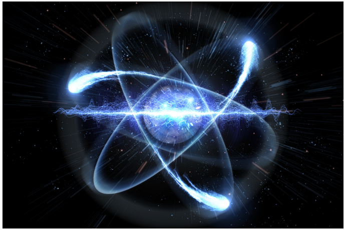
Summary

- McEliece is well studied and appears to be secure...
- ...even in a post-quantum setting
- This comes at cost of large key-sizes



Summary

- McEliece is well studied and appears to be secure...
- ...even in a post-quantum setting
- This comes at cost of large key-sizes
- Most NIST submissions try to address this issue by using special classes of codes
- Their decoding problem is much less analyzed



The study of this tradeoff will probably continue over the next years.

Further questions?