Code-based Public-key Encryption Schemes

Vlad-Florin Dragoi

vlad.dragoi@uav.ro

Universitatea "Aurel Vlaicu" Arad, Romania
LITIS EA 4108 Université de Rouen Normandie, France

spring 2022 WSTC

Insights into the theory of error correcting codes

Error correcting codes

Definition 1

A binary linear code $\mathscr{C}$ *defined over* $\mathbb{F}_2$ *is a* k *dimension sub-vector space of* $\mathbb{F}_2^n$.
$\boldsymbol{G} \in \mathbb{F}_2^{k \times n}$ *a basis, and* $\boldsymbol{H} \in \mathbb{F}_2^{(n-k) \times n}$ *a basis for the dual.*

$$\mathscr{C} =< \boldsymbol{G} >= \{\boldsymbol{c} = \boldsymbol{mG} \mid \boldsymbol{m} \in \mathbb{F}_2^k\} \quad \mathscr{C} =< \boldsymbol{H} >^{\perp}= \{\boldsymbol{Hc} = 0 \mid \boldsymbol{c} \in \mathbb{F}_2^n\}$$

Error correcting codes

Definition 1

A binary linear code $\mathscr{C}$ defined over $\mathbb{F}_2$ is a k dimension sub-vector space of $\mathbb{F}_2^n$.
$\boldsymbol{G} \in \mathbb{F}_2^{k \times n}$ a basis, and $\boldsymbol{H} \in \mathbb{F}_2^{(n-k) \times n}$ a basis for the dual.

$$\mathscr{C} =< \boldsymbol{G} >= \{\boldsymbol{c} = \boldsymbol{mG} \mid \boldsymbol{m} \in \mathbb{F}_2^k\} \quad \mathscr{C} =< \boldsymbol{H} >^{\perp}= \{\boldsymbol{Hc} = 0 \mid \boldsymbol{c} \in \mathbb{F}_2^n\}$$

Remark

For any $\boldsymbol{x} \in \mathbb{F}_2^n$ denote $\mathrm{supp}(\boldsymbol{x}) = \{i \mid x_i \neq 0\}$.
Any $\boldsymbol{x} \in \mathbb{F}_2^n$ with $|\mathrm{supp}(\boldsymbol{x})| = 0 \mod 2$ is self-orthogonal.

$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \sum_{i=1}^{n} x_i \mod 2 = 0.$$

A linear code is a metric space

Definition 2 (Hamming weight and distance)

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ *and* $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{F}_2^n$

$$\|\boldsymbol{x}\| \stackrel{def}{=} |\{i \mid x_i \neq 0\}| \quad \mathrm{d_H}(\boldsymbol{x}, \boldsymbol{y}) \stackrel{def}{=} |\{i \mid x_i \neq y_i\}|$$

A linear code is a metric space

Definition 2 (Hamming weight and distance)

Let $\boldsymbol{x} = (x_1, \dots, x_n)$ *and* $\boldsymbol{y} = (y_1, \dots, y_n) \in \mathbb{F}_2^n$

$$\|\boldsymbol{x}\| \stackrel{def}{=} |\{i \mid x_i \neq 0\}| \quad \mathrm{d_H}(\boldsymbol{x}, \boldsymbol{y}) \stackrel{def}{=} |\{i \mid x_i \neq y_i\}|$$

$$\begin{aligned}
\mathrm{d_{min}}(\mathscr{C}) &= \min_{\substack{(\boldsymbol{c},\boldsymbol{c}^*)\in\mathscr{C}\times\mathscr{C} \\ \boldsymbol{c}\neq\boldsymbol{c}^*}} \mathrm{d_H}(\boldsymbol{c}, \boldsymbol{c}^*) \\
&= \min_{\boldsymbol{c}\in\mathscr{C},\boldsymbol{c}\neq 0} \|\boldsymbol{c}\| \\
&= \min_{\boldsymbol{c}\in\mathscr{C},\boldsymbol{c}\neq 0} |\mathrm{supp}(\boldsymbol{c})|.
\end{aligned}$$

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance
- n, k are easy to determine (basis)

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance
- d depends on the family of codes

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance
- d depends on the family of codes
  - Codes with particular underlying structure could have an easy computable d

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance
- d depends on the family of codes
  - In general computing d given $\boldsymbol{G}$ or $\boldsymbol{H}$ is a difficult problem [1]

1. A. Vardy, "The intractability of computing the minimum distance of a code," in IEEE Transactions on Information Theory, vol. 43, no. 6, pp. 1757-1766, Nov. 1997

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance
- d depends on the family of codes
  - In general computing d given $\boldsymbol{G}$ or $\boldsymbol{H}$ is a difficult problem
  - The Gilbert-Varshamov bound, d_{GV} is the smallest d s.t.

$$\sum_{i=0}^{d-1} \binom{n}{i} \geqslant 2^{(n-k)}$$

Code parameters

- $\mathscr{C}$ is a $[n, k, d]$ code : n-length, k-dimension, d-minimum distance
- d depends on the family of codes
  - In general computing d given $\boldsymbol{G}$ or $\boldsymbol{H}$ is a difficult problem
  - The Gilbert-Varshamov bound, d_{GV} is the smallest d s.t.

$$\sum_{i=0}^{d-1} \binom{n}{i} \geqslant 2^{(n-k)}$$

  - In the asymptotics : The minimum distance of a $[n, k]$ linear code meets the Gilbert-Varshamov bound [1] $d_{GV} = n\delta_{GV}$

$$1 - k/n = H(\delta_{GV})$$

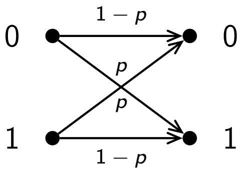
1. A. Barg and G. D. Forney, "Random codes : minimum distances and error exponents," in IEEE Transactions on Information Theory, vol. 48, no. 9, pp. 2568-2573, Sept. 2002

Encoding-Decoding

$\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$ $\quad$ $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ $\quad$ $Alg(\boldsymbol{G}, \boldsymbol{y}) = \boldsymbol{c}$

Encoding-Decoding

$\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$ $\quad$ $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ $\quad$ $Alg(\boldsymbol{G}, \boldsymbol{y}) = \boldsymbol{c}$



Decoding

Definition 1 (Discrete channel)

Let k and m be two strictly positive integers. Then a discrete channel W is defined by

- A finite input alphabet $\mathcal{X} = \{x_1, \ldots, x_k\}$.
- A finite output alphabet $\mathcal{Y} = \{y_1, \ldots, y_m\}$.
- The transition probability matrix $\boldsymbol{P} = (p_{i,j})_{1 \leqslant i \leqslant k \;,\; 1 \leqslant j \leqslant m}$ with $p_{i,j} = W(y_j | x_i)$ is the probability that y_i is received knowing that x_i was sent over the channel.

Decoding

Definition 2

A decoder for $\mathscr{C}$ with respect to W is a function $\mathcal{D} : \mathcal{Y}^n \to \mathscr{C}$.
The probability that a codeword $\boldsymbol{c}$ is decoded erroneously, given that $\boldsymbol{c}$ was transmitted

$$\mathrm{P}_{\mathrm{err}}(\boldsymbol{c}) \stackrel{\text{def}}{=} \sum_{\substack{\boldsymbol{y} \in \mathcal{Y}^n \\ \mathcal{D}(\boldsymbol{y}) \neq \boldsymbol{c}}} W(\boldsymbol{y} \mid \boldsymbol{c}).$$

The *error probability* of $\mathcal{D}$ is

$$\mathrm{P}_{\mathrm{err}} = \max_{\boldsymbol{c} \in \mathscr{C}} \mathrm{P}_{\mathrm{err}}(\boldsymbol{c}).$$

Decoding

Definition 3 (Maximum-Likelihood Decoder)

Given a $[n, k, d]$ linear code $\mathscr{C}$ over $\mathbb{F}_2$ and a channel $W = (\mathbb{F}_2, \mathcal{Y}, \boldsymbol{P})$ a *maximum-likelihood decoder* (MLD) for $\mathscr{C}$ with respect to W is the function $\mathcal{D}_{\mathrm{MLD}} : \mathcal{Y}^n \to \mathscr{C}$ defined as :

$$\text{for every } \boldsymbol{y} \in \mathcal{Y}^n \text{ , } \mathcal{D}_{\mathrm{MLD}}(\boldsymbol{y}) \stackrel{\text{def}}{=} \mathrm{argmax}_{\boldsymbol{c} \in \mathscr{C}} W(\boldsymbol{y} \mid \boldsymbol{c}).$$

Decoding

Definition 3 (Maximum-Likelihood Decoder)

Given a $[n, k, d]$ linear code $\mathscr{C}$ over $\mathbb{F}_2$ and a channel $W = (\mathbb{F}_2, \mathcal{Y}, \boldsymbol{P})$ a *maximum-likelihood decoder* (MLD) for $\mathscr{C}$ with respect to W is the function $\mathcal{D}_{\mathrm{MLD}} : \mathcal{Y}^n \to \mathscr{C}$ defined as :

$$\text{for every } \boldsymbol{y} \in \mathcal{Y}^n, \mathcal{D}_{\mathrm{MLD}}(\boldsymbol{y}) \stackrel{\text{def}}{=} \mathrm{argmax}_{\boldsymbol{c} \in \mathscr{C}} W(\boldsymbol{y} \mid \boldsymbol{c}).$$

Ex. $\mathrm{BSC}(p)$ with crossover probability $0 < p < 1/2$

$$\begin{aligned} W(\boldsymbol{y} \mid \boldsymbol{c}) &= \prod_{i=1}^{n} W(y_i \mid c_i) \\ &= p^{\mathrm{d_H}(\boldsymbol{y},\boldsymbol{c})}(1-p)^{n-\mathrm{d_H}(\boldsymbol{y},\boldsymbol{c})} \\ &= (1-p)^n \left(\frac{p}{1-p}\right)^{\mathrm{d_H}(\boldsymbol{y},\boldsymbol{c})}. \end{aligned}$$

Decoding

Definition 3 (Maximum-Likelihood Decoder)

Given a $[n, k, d]$ linear code $\mathscr{C}$ over $\mathbb{F}_2$ and a channel $W = (\mathbb{F}_2, \mathcal{Y}, \boldsymbol{P})$ a *maximum-likelihood decoder* (MLD) for $\mathscr{C}$ with respect to W is the function $\mathcal{D}_{\mathrm{MLD}} : \mathcal{Y}^n \to \mathscr{C}$ defined as :

$$\text{for every } \boldsymbol{y} \in \mathcal{Y}^n \ , \mathcal{D}_{\mathrm{MLD}}(\boldsymbol{y}) \stackrel{\text{def}}{=} \mathrm{argmax}_{\boldsymbol{c} \in \mathscr{C}} W(\boldsymbol{y} \mid \boldsymbol{c}).$$

Ex. $\mathrm{BSC}(p)$ with crossover probability $0 < p < 1/2$

$$W(\boldsymbol{y} \mid \boldsymbol{c}) = (1-p)^n \left(\frac{p}{1-p}\right)^{\mathrm{d_H}(\boldsymbol{y}, \boldsymbol{c})}.$$

$\mathcal{D}_{\mathrm{MLD}}(\boldsymbol{y})$ is the codeword $\boldsymbol{c}$ which minimize $\mathrm{d_H}(\boldsymbol{y}, \boldsymbol{c})$

$\boldsymbol{c}$ is the closest codeword of $\mathscr{C}$ to $\boldsymbol{y}$.

Nearest codeword problem

Definition 4 (Nearest Codeword Problem for BSC)

Given : $[n, k, d]$ linear code $\mathscr{C}$ over $\mathbb{F}_2$ and a vector $\boldsymbol{y} \in \mathbb{F}_2^n$.
Find : $\boldsymbol{e} \in \mathbb{F}_2^n$ of minimum Hamming weight such that $\boldsymbol{y} - \boldsymbol{e} \in \mathscr{C}$.

Nearest codeword problem

Definition 4 (Nearest Codeword Problem for BSC)

Given : $[n, k, d]$ linear code $\mathscr{C}$ over $\mathbb{F}_2$ and a vector $\boldsymbol{y} \in \mathbb{F}_2^n$.
Find : $\boldsymbol{e} \in \mathbb{F}_2^n$ of minimum Hamming weight such that $\boldsymbol{y} - \boldsymbol{e} \in \mathscr{C}$.

A possible solution is to use the dual code

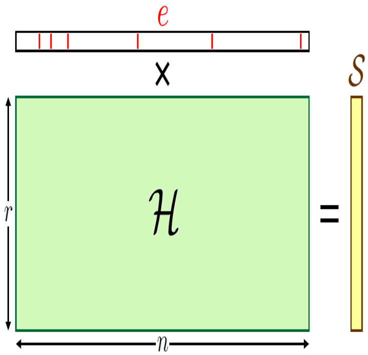
- $\boldsymbol{y} - \boldsymbol{e} \in \mathscr{C} \Leftrightarrow \boldsymbol{H}(\boldsymbol{y} - \boldsymbol{e}) = 0$
- let $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{y}$ be a syndrome (associated to a vector, with respect to a matrix)
- We have

$$\boldsymbol{y} - \boldsymbol{e} \in \mathscr{C} \Leftrightarrow \boldsymbol{H}\boldsymbol{e} = \boldsymbol{s}$$

Syndrome decoding [2]

Given : A parity-check matrix $\boldsymbol{H}$ for a $[n, k, d]$ binary linear code
a syndrome vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \in \mathbb{N}$

Find : $\boldsymbol{e} \in \mathbb{F}_2^n$ of weight at most t such that $\boldsymbol{He} = \boldsymbol{s}$.



2. 1978. Berlekamp E., McEliece R.J., Van Tilborg "On the inherent intractability of certain coding problems."

Bounded decoding [3]

If there is a codeword $\boldsymbol{c}$ s.t. $\mathrm{d_H}(\boldsymbol{c}, \boldsymbol{y}) \leqslant \lfloor \frac{d-1}{2} \rfloor$ we talk about unique solution (bounded decoding).

Given : A parity-check matrix $\boldsymbol{H}$ for a $[n, k]$ binary linear code
a syndrome vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \leqslant \lfloor \frac{d-1}{2} \rfloor$

Promise : any $d - 1$ columns of $\boldsymbol{H}$ are linearly independent

Find : $\boldsymbol{e} \in \mathbb{F}_2^n$ of weight at most t such that $\boldsymbol{He} = \boldsymbol{s}$.

3. A Barg. Complexity issues in coding theory. *Handbook of Coding Theory, Elsevier Science*, 1998.

Bounded decoding [3]

If there is a codeword $\boldsymbol{c}$ s.t. $\mathrm{d_H}(\boldsymbol{c}, \boldsymbol{y}) \leqslant \lfloor \frac{d-1}{2} \rfloor$ we talk about unique solution (bounded decoding).

Given : A parity-check matrix $\boldsymbol{H}$ for a $[n, k]$ binary linear code
a syndrome vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \leqslant \lfloor \frac{d-1}{2} \rfloor$
Promise : any $d - 1$ columns of $\boldsymbol{H}$ are linearly independent
Find : $\boldsymbol{e} \in \mathbb{F}_2^n$ of weight at most t such that $\boldsymbol{He} = \boldsymbol{s}$.

Verifying the promise condition is NP-complete. Bounded Decoding was conjectured NP-hard for random linear codes.

3. A Barg. Complexity issues in coding theory. *Handbook of Coding Theory, Elsevier Science*, 1998.

Decoding

- Random linear codes :
  - maximum likelihood decoding (NP-complete)

Decoding

- Random linear codes :
  - maximum likelihood decoding (NP-complete)
  - bounded decoding (Conjectured NP-hard)

Decoding

- Random linear codes :
  - maximum likelihood decoding (NP-complete)
  - bounded decoding (Conjectured NP-hard)

- Codes with a particular structure :
  - maximum likelihood decoding is NP-complete even for Reed-Solomon, concatenated codes.

Decoding

- Random linear codes :
  - maximum likelihood decoding (NP-complete)
  - bounded decoding (Conjectured NP-hard)

- Codes with a particular structure :
  - maximum likelihood decoding is NP-complete even for Reed-Solomon, concatenated codes.
  - efficient algorithms for bounded decoding exist :
    - ⋆ Patterson/Berlekamp-Massey algorithm - Goppa codes
    - ⋆ Extended Euclidean Algorithm -Alternant codes, Reed-Solomon codes, BCH codes
    - ⋆ Bit flipping algorithm -LDPC/MDPC codes,
    - ⋆ Reed algorithm, Berlekamp-Welsh algorithm - Reed-Muller codes

Some useful problems

- Given a random linear code $\mathscr{C}$ specified by $\boldsymbol{G}$ and an erroneous codeword, retrieve the initial codeword.

$$\psi(\boldsymbol{Gm} + \boldsymbol{e}, \boldsymbol{G}) = \boldsymbol{m}$$

- Given a random linear code $\mathscr{C}$ specified by $\boldsymbol{H}$ and a syndrome vector, retrieve the error vector.

$$\psi(\boldsymbol{He}, \boldsymbol{H}) = \boldsymbol{e}$$

- Given a random linear code $\mathscr{C}$ and a vector, distinguish between random vectors and erroneous codewords.

$$\varphi(\boldsymbol{G}, \boldsymbol{y}) = \begin{cases} 0 & \textit{if } \boldsymbol{y} = \textit{random} \\ 1 & \textit{if } \boldsymbol{y} = \boldsymbol{mG} + \boldsymbol{e} \end{cases}$$

Public-key encryption schemes from codes

Public-key encryption from codes

- Choose a family of codes that admits an efficient decoding algorithm

Public-key encryption from codes

- Choose a family of codes that admits an efficient decoding algorithm
- Intentionally add errors to a codeword – **Encryption**

$$\text{(McEliece) } \boldsymbol{z} = \boldsymbol{mG} + \boldsymbol{e} \qquad \text{or} \qquad \boldsymbol{m} \rightarrow \boldsymbol{e}\,,\ \boldsymbol{z} = \boldsymbol{He}^t \text{ (Niederreiter)}$$

Public-key encryption from codes

- Choose a family of codes that admits an efficient decoding algorithm
- Intentionally add errors to a codeword – **Encryption**

$$\text{(McEliece) } \boldsymbol{z} = \boldsymbol{mG} + \boldsymbol{e} \qquad \text{or} \qquad \boldsymbol{m} \to \boldsymbol{e} \text{ , } \boldsymbol{z} = \boldsymbol{He}^t \text{ (Niederreiter)}$$

- Mask the structure of the underlying code – **Key generation**

$$\boldsymbol{G_{pub}} = \boldsymbol{SGP} \qquad , \qquad \boldsymbol{H_{pub}} = \boldsymbol{SHP}$$

McEliece PKE	Niederreiter PKE
KeyGen$(n, k, t) = (\mathrm{pk}, \mathrm{sk})$	
$\boldsymbol{G}$-generator matrix matrix of $\mathscr{C}$	$\boldsymbol{H}$-parity-check of $\mathscr{C}$
$\backslash\backslash$ $\mathscr{C}$ an $[n, k]$ that corrects t errors	
An $n \times n$ permutation matrix $\boldsymbol{P}$	
A $k \times k$ invertible matrix $\boldsymbol{S}$	An $(n-k) \times (n-k)$ invertible matrix $\boldsymbol{S}$
Compute $\boldsymbol{G_{pub}} = \boldsymbol{SGP}$	Compute $\boldsymbol{H_{pub}} = \boldsymbol{SHP}$
$\mathrm{pk} = (\boldsymbol{G_{pub}}, t)$ $\mathrm{sk} = (\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{P})$	$\mathrm{pk} = (\boldsymbol{H_{pub}}, t)$ $\mathrm{sk} = (\boldsymbol{S}, \boldsymbol{H}, \boldsymbol{P})$

Encrypt$(\boldsymbol{m}, \mathrm{pk}) = \boldsymbol{z}$	
Encode $\boldsymbol{m} \to \boldsymbol{c} = \boldsymbol{mG_{pub}}$ Choose $\boldsymbol{e}$	Encode $\boldsymbol{m} \to \boldsymbol{e}$
$\backslash\backslash$ $\boldsymbol{e}$ a vector of weight t	
$\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e}$	$\boldsymbol{z} = \boldsymbol{H_{pub}} \boldsymbol{e}^t$

Decrypt$(\boldsymbol{z}, \mathrm{sk}) = \boldsymbol{m}$	
Compute $\boldsymbol{z}^* = \boldsymbol{zP}^{-1}$ $\boldsymbol{z}^* = \boldsymbol{mSG} + \boldsymbol{eP}^{-1}$ $\boldsymbol{m}^* = \mathcal{Decode}(\boldsymbol{z}^*, \boldsymbol{G})$ Retrieve $\boldsymbol{m}$ from $\boldsymbol{m}^* \boldsymbol{S}^{-1}$	Compute $\boldsymbol{z}^* = \boldsymbol{S}^{-1}\boldsymbol{z}$ $\boldsymbol{z}^* = \boldsymbol{HPe}$ $\boldsymbol{e}^* = \mathcal{Decode}(\boldsymbol{z}^*, \boldsymbol{H})$ Retrieve $\boldsymbol{m}$ from $\boldsymbol{P}^{-1}\boldsymbol{e}^*$

Semantic Security

One-way function

- Assumptions
  - Indistinguishability : The public code is computationally indistinguishable from a uniformly chosen code of the same size (n,k).
  - Decoding hardness : Decoding a random linear code with parameters n, k, t is hard.

4. B. Biswas, N. Sendrier. McEliece Cryptosystem Implementation : Theory and Practice. PQCrypto. pp. 47-62. 2008.

Semantic Security

One-way function

- Assumptions
  - Indistinguishability : The public code is computationally indistinguishable from a uniformly chosen code of the same size (n,k).
  - Decoding hardness : Decoding a random linear code with parameters n, k, t is hard.
- Given that both the above assumptions hold, the McEliece cryptosystem is one-way secure under passive attacks.[4]

4. B. Biswas, N. Sendrier. McEliece Cryptosystem Implementation : Theory and Practice. PQCrypto. pp. 47-62. 2008.

Decoding hardness in the McEliece scheme [5]

The binary Goppa code is a $[2^m, 2^m - mt, 2t + 1]$

5. Finiasz, Matthieu. "Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique." (2004).

Decoding hardness in the McEliece scheme [5]

The binary Goppa code is a $[2^m, 2^m - mt, 2t+1]$

Given : A parity-check matrix $\boldsymbol{H}$ for a $[n, n-k]$ binary linear code
a syndrome vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \in \mathbb{N}$ $(n = 2^m)$

Find : $\boldsymbol{e} \in \mathbb{F}_2^n$ of weight $t \leqslant (n-k)/\log_2(n)$ such that $\boldsymbol{He} = \boldsymbol{s}$.

5. Finiasz, Matthieu. "Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique." (2004).

Distinguisher assumption for Goppa codes [6]

- Pseudo-randomness assumption
  Input : A generator matrix $\boldsymbol{G}$ for a $[2^m, 2^m - mt]$ binary linear code
  Output : $\boldsymbol{G}$ generates a Goppa code ?

6. Jean-Charles Faugère, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret, Jean-Pierre Tillich. A Distinguisher for High Rate McEliece Cryptosystems. IEEE Transactions on Information Theory 2013.

Semantic Security

Critical Attacks

- McEliece PKE does not satisfy Non-Malleability (linearity)

$$
\begin{aligned}
\text{given a McEliece criptogram} \quad & \boldsymbol{y} = \boldsymbol{m}\boldsymbol{G}_{\boldsymbol{pub}} + \boldsymbol{e} \\
\text{compute a well-choose criptogram} \quad & \boldsymbol{y}^* = \boldsymbol{m}^*\boldsymbol{G}_{\boldsymbol{pub}} \\
\text{as the oracle to decrypt} \quad & \boldsymbol{y} + \boldsymbol{y}^* = (\boldsymbol{m} + \boldsymbol{m}^*)\boldsymbol{G}_{\boldsymbol{pub}} + \boldsymbol{e}
\end{aligned}
$$

Semantic Security

Critical Attacks

- McEliece PKE does not satisfy Non-Malleability (linearity)

$$\begin{aligned} \text{given a McEliece criptogram} \quad & \boldsymbol{y} = \boldsymbol{m}\boldsymbol{G_{pub}} + \boldsymbol{e} \\ \text{compute a well-choose criptogram} \quad & \boldsymbol{y}^* = \boldsymbol{m}^*\boldsymbol{G_{pub}} \\ \text{as the oracle to decrypt} \quad & \boldsymbol{y} + \boldsymbol{y}^* = (\boldsymbol{m} + \boldsymbol{m}^*)\boldsymbol{G_{pub}} + \boldsymbol{e} \end{aligned}$$

- Reaction attacks in the CCA model

$$\begin{aligned} \text{given a McEliece criptogram} \quad & \boldsymbol{y} = \boldsymbol{m}\boldsymbol{G_{pub}} + \boldsymbol{e} \\ \text{flip a bit} \quad & \boldsymbol{y}^{'} = \boldsymbol{y} + (1, 0, \dots, 0) \\ & \boldsymbol{y}^{'} = \boldsymbol{m}\boldsymbol{G_{pub}} + \boldsymbol{e} + (1, 0 \dots, 0) \\ \text{if the decoder reaction is invalid ciphertext} \quad & e_i = 0 \\ \text{if the decoder reaction is valid ciphertext} \quad & e_i = 1 \end{aligned}$$

Semantic Security

Critical Attacks

- Resend-message attacks : the same message was encrypted several times

$$\begin{aligned}
\text{intercept} \quad & \boldsymbol{y}_1 = \boldsymbol{m}\boldsymbol{G}_{\boldsymbol{pub}} + \boldsymbol{e}_1 \\
\text{intercept} \quad & \boldsymbol{y}_2 = \boldsymbol{m}\boldsymbol{G}_{\boldsymbol{pub}} + \boldsymbol{e}_2 \\
\text{notice that} \quad & \mathrm{d_H}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \mathrm{d_H}(\boldsymbol{e}_1, \boldsymbol{e}_2) = 2t - 2\delta \\
\text{if the messages were different} \quad & \mathrm{d_H}(\boldsymbol{y}_1, \boldsymbol{y}_2) \sim n/2 \\
\text{select the set} \quad & I = \mathrm{supp}(\boldsymbol{y}_1 - \boldsymbol{y}_2) \\
\text{Gaussian elimination on } I \quad & \boldsymbol{H}_{\boldsymbol{pub}}\boldsymbol{e}_1 = \boldsymbol{s}_1.
\end{aligned}$$

Semantic Security

Conversions

- For a McEliece IND-CPA without random oracles simply randomize the message $\boldsymbol{m}^* = (\boldsymbol{m}|\boldsymbol{r})$ [7]

7. Nojima, R., Imai, H., Kobara, K. et al. Semantic security for the McEliece cryptosystem without random oracles. Des. Codes Cryptogr. 49, 289–305 (2008)
8. K. Kobara and H. Imai. Semantically Secure McEliece Public-Key Cryptosystems Conversions for McEliece PKC, LNCS Springer, 2001

Semantic Security

Conversions

- For a McEliece IND-CPA without random oracles simply randomize the message $\boldsymbol{m}^* = (\boldsymbol{m}|\boldsymbol{r})$ [7]
- For random oracles model - convert the one way trap door function into an IND-CCA2 PKC
  - simple OAEP conversion not working because of reaction attacks
  - Kobara,Imai conversion to obtain an IND-CCA2 [8]

7. Nojima, R., Imai, H., Kobara, K. et al. Semantic security for the McEliece cryptosystem without random oracles. Des. Codes Cryptogr. 49, 289–305 (2008)
8. K. Kobara and H. Imai. Semantically Secure McEliece Public-Key Cryptosystems Conversions for McEliece PKC, LNCS Springer, 2001

McEliece and Niederreiter

MRA, KRA, Distinguisher

	McEliece	Niederreiter
pk	$\boldsymbol{G_{pub}}$	$\boldsymbol{H_{pub}}$
MRA	Generic decoding $Alg(\boldsymbol{m}\boldsymbol{G_{pub}} + \boldsymbol{e}, \boldsymbol{G_{pub}}) = \boldsymbol{m}$ $\lVert\boldsymbol{e}\rVert$ small	Syndrome decoding $Alg(\boldsymbol{H_{pub}}\boldsymbol{e}, \boldsymbol{H_{pub}}) = \boldsymbol{e}$ $\lVert\boldsymbol{e}\rVert$ small
KRA	Code Equivalence Problem $Alg(\boldsymbol{G_{pub}}, \boldsymbol{G}) = \boldsymbol{P}^*$	$Alg(\boldsymbol{H_{pub}}, \boldsymbol{H}) = \boldsymbol{P}^*$
	$\mathscr{C} \overset{P.E.}{\sim} \mathscr{C}_{pub} \Leftrightarrow \mathscr{C}^\perp \overset{P.E.}{\sim} \mathscr{C}^\perp_{pub}$	
Distinguisher	$D(\boldsymbol{G_{pub}}) = \begin{cases} 0 & if\, \delta = \delta_{Goppa} \\ 1 & if\, \delta = \delta_{Random} \end{cases}$	$D(\boldsymbol{H_{pub}}) = \begin{cases} 0 & if\, \delta = \delta_{Reed-Solomon} \\ 1 & if\, \delta = \delta_{Random} \end{cases}$

Distinguish a public code from a random code

Efficient distinguisher for some families of codes

$$\boldsymbol{x} \star \boldsymbol{y} = (x_1 y_1, \ldots, x_n y_n).$$

Efficient distinguisher for some families of codes

$$\boldsymbol{x} \star \boldsymbol{y} = (x_1 y_1, \dots, x_n y_n).$$

Theorem 5 (Cascudo, Cramer, Mirandola, Zemor -2015)

Let $\mathscr{C}_1 = [n, k_1]$ and $\mathscr{C}_2 = [n, k_2]$. Then w.h.p. we have

$$Dim\left(\mathscr{C}_1 \star \mathscr{C}_2\right) = \min\left\{n,\ k_1 k_2 - \binom{Dim\left(\mathscr{C}_1 \cap \mathscr{C}_2\right)}{2}\right\}.$$

Efficient distinguisher for some families of codes

$$\boldsymbol{x} \star \boldsymbol{y} = (x_1 y_1, \ldots, x_n y_n).$$

Theorem 5 (Cascudo, Cramer, Mirandola, Zemor -2015)

Let $\mathscr{C}_1 = [n, k_1]$ and $\mathscr{C}_2 = [n, k_2]$. Then w.h.p. we have

$$Dim\left(\mathscr{C}_1 \star \mathscr{C}_2\right) = \min\left\{n,\ k_1 k_2 - \binom{Dim\left(\mathscr{C}_1 \cap \mathscr{C}_2\right)}{2}\right\}.$$

In particular, for $\mathscr{C} = [n, k]$ random binary code we have

$$Dim\left(\mathscr{C}^2\right) = \min\left\{n,\ \binom{k+1}{2}\right\}. \tag{1}$$

Reed-Solomon codes

Definition 6 (Generalized Reed-Solomon codes)

Let $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_{2^m}^n \times \mathbb{F}_{2^m}^n$ be a pair such that $\forall i, y_i \neq 0$ and $\forall i \neq j, x_i \neq x_j$.

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid f \in \mathbb{F}_q[x]\ ,\ \deg(f) < k\}\ .$$

Reed-Solomon codes

Definition 6 (Generalized Reed-Solomon codes)

Let $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_{2^m}^n \times \mathbb{F}_{2^m}^n$ be a pair such that $\forall i, y_i \neq 0$ and $\forall i \neq j, x_i \neq x_j$.

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid f \in \mathbb{F}_q[x] \;,\; \deg(f) < k\} \,.$$

$$\boldsymbol{G}_{\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \\ x_1^2 & x_2^2 & \ldots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{k-1} & x_2^{k-1} & \ldots & x_n^{k-1} \end{pmatrix} \begin{pmatrix} y_1 & & & \\ & y_2 & & 0 \\ 0 & & \ddots & \\ & & & y_n \end{pmatrix} .$$

Reed-Solomon codes

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathbf{GRS}_{n-k}(\boldsymbol{x}, \boldsymbol{z}), \quad \boldsymbol{H}_{\mathbf{GRS}_{n-1}(\boldsymbol{x},\boldsymbol{y})}\boldsymbol{z}^T = 0$$

Reed-Solomon codes

$$\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})^\perp = \mathbf{GRS}_{n-k}(\boldsymbol{x},\boldsymbol{z}), \quad \boldsymbol{H}_{\mathbf{GRS}_{n-1}(\boldsymbol{x},\boldsymbol{y})}\boldsymbol{z}^T = 0$$

$$\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})^2 = \mathbf{GRS}_{2k-1}(\boldsymbol{x},\boldsymbol{y}^2)$$

Reed-Solomon codes

$$\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})^{\perp} = \mathbf{GRS}_{n-k}(\boldsymbol{x},\boldsymbol{z}), \quad \boldsymbol{H}_{\mathbf{GRS}_{n-1}(\boldsymbol{x},\boldsymbol{y})}\boldsymbol{z}^T = 0$$

$$\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})^2 = \mathbf{GRS}_{2k-1}(\boldsymbol{x},\boldsymbol{y}^2)$$

$$3 \leqslant k \leqslant \frac{n+1}{2}, \quad Dim(\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})^2) = 2k-1 < \binom{k+1}{2}$$

Reed-Muller codes

$$\mathscr{R}(r,m) \stackrel{\text{def}}{=} \left\{ \left(g(v_1,\ldots,v_m)\right)_{(v_1,\ldots,v_m)\in\mathbb{F}_2^m} \;\middle|\; g \in \mathbb{F}_2[x_1,\ldots,x_m], \deg g \leqslant r \right\}.$$

Reed-Muller codes

$$\mathscr{R}(r,m) \stackrel{\text{def}}{=} \left\{ (g(v_1,\dots,v_m))_{(v_1,\dots,v_m)\in\mathbb{F}_2^m} \;\middle|\; g \in \mathbb{F}_2[x_1,\dots,x_m], \deg g \leqslant r \right\}.$$

$$Dim(\mathscr{R}(r,m)) = \sum_{i=0}^{r} \binom{m}{i}$$

Reed-Muller

$$\mathscr{R}(r, m)^{\perp} = \mathscr{R}(m - r - 1, m)$$

Reed-Muller

$$\mathscr{R}(r, m)^{\perp} = \mathscr{R}(m - r - 1, m)$$

$$\mathscr{R}(r, m)^{2} = \mathscr{R}(2r, m)$$

Reed-Muller

$$\mathscr{R}(r,m)^{\perp} = \mathscr{R}(m-r-1,m)$$

$$\mathscr{R}(r,m)^{2} = \mathscr{R}(2r,m)$$

$$Dim(\mathscr{R}(r,m)^2) = \sum_{i=0}^{2r}\binom{m}{i} < \binom{\sum_{i=0}^{r}\binom{m}{i}+1}{2}.$$

Alternant and Goppa codes

$$\mathbf{Alt}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_2^n.$$

9. `https://arxiv.org/pdf/2111.13038.pdf`

Alternant and Goppa codes

$$\mathbf{Alt}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} \cap \mathbb{F}_2^n.$$

$$\Gamma(\boldsymbol{x}, g) \stackrel{\text{def}}{=} \mathbf{Alt}_t(\boldsymbol{x}, \boldsymbol{y}), \text{ where } y_i = \frac{1}{g(x_i)}, g \in \mathbb{F}_{2^m}[x], \deg g = t$$

9. https://arxiv.org/pdf/2111.13038.pdf

Alternant and Goppa codes

$$\mathbf{Alt}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} \cap \mathbb{F}_2^n.$$

$$\Gamma(\boldsymbol{x}, g) \stackrel{\text{def}}{=} \mathbf{Alt}_t(\boldsymbol{x}, \boldsymbol{y}), \text{ where } y_i = \frac{1}{g(x_i)}, g \in \mathbb{F}_{2^m}[x], \deg g = t$$

$$\mathbf{Alt}_r(\boldsymbol{x}, \boldsymbol{y})^2 = ???$$ [9]

9. `https://arxiv.org/pdf/2111.13038.pdf`

Binary Goppa codes

Wanted for crypto resilience

Definition 7 (Binary Goppa codes)

Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{2^m}^n$ with $x_i \neq x_j$,
$g \in \mathbb{F}_{2^m}[x]$ with $\deg(g) = t$ s.t. $\forall\ 1 \leqslant i \leqslant n, g(x_i) \neq 0$.
$\forall \boldsymbol{c} \in \mathbb{F}_2^n$ define the rational function $s_{\boldsymbol{c}}(x) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \frac{c_i}{x - x_i}$.
The binary Goppa code is

$$\Gamma(\boldsymbol{x}, g) \stackrel{\text{def}}{=} \left\{ \boldsymbol{c} \in \mathbb{F}_2^n \mid s_{\boldsymbol{c}}(x) \equiv 0 \mod g(x) \right\}.$$

Patterson Algorithm

- If $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ then

$$s_{\boldsymbol{y}}(x) = \sum_{i=0}^{n} \frac{c_i + e_i}{x - x_i} \equiv s_{\boldsymbol{e}}(x) \mod g(x)$$

Patterson Algorithm

- If $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ then

$$s_{\boldsymbol{y}}(x) = \sum_{i=0}^{n} \frac{c_i + e_i}{x - x_i} \equiv s_{\boldsymbol{e}}(x) \mod g(x)$$

- This implies

$$s_{\boldsymbol{y}}(x) \equiv \sum_{i \in \mathrm{supp}(\boldsymbol{e})} \frac{1}{x - x_i} \mod g(x)$$

- $\sigma(x)$ is called the error-locator polynomial : $\sigma(x) = \prod_{i \in \mathrm{supp}(\boldsymbol{e})} (x - x_i)$.

Patterson Algorithm

$$
\begin{aligned}
\sigma(x)' &= \sum_{i \in \mathrm{supp}(\boldsymbol{e})}^{n} \prod_{j \in \mathrm{supp}(\boldsymbol{e}), j \neq i} (x - x_j) \\
&= \sum_{i \in \mathrm{supp}(\boldsymbol{e})}^{n} \frac{1}{x - x_i} \prod_{i \in \mathrm{supp}(\boldsymbol{e})} (x - x_i) \\
&= \sigma(x) \sum_{i \in \mathrm{supp}(\boldsymbol{e})}^{n} \frac{1}{x - x_i}
\end{aligned}
$$

$$
\sigma'(x) \equiv \sigma(x) s_{\boldsymbol{y}}(x) \mod g(x).
$$

Patterson Algorithm

- Let $\sigma(x) = a(x)^2 + xb(x)^2$ $(\deg(a) \leqslant (t-1)/2, \deg(b) \leqslant t/2)$.
- This implies $\sigma(x)' = b(x)^2$ (over $\mathbb{F}_2$), which makes

$$b^2 = \sigma' = \sigma s_{\mathbf{y}} = (a^2 + xb^2)s_{\mathbf{y}} \mod g$$

- Since $s_{\mathbf{y}}, g$ coprime, we have

$$a^2 = b^2\sqrt{x + s_{\mathbf{y}}^{-1}} \mod g.$$

- Find $a(x), b(x)$ using Extended Euclidean Algorithm and compute $\sigma(x)$.

Patterson Algorithm

Input : The syndrome polynomial $s_{\mathbf{s}}(x)$ and the Goppa code $g(x)$.
Output : The error vector $\boldsymbol{e}$

1. $s_{\mathbf{s}}(x)^{-1} \leftarrow \mathrm{EEA}(g(x), s_{\mathbf{s}}(x))$
2. $\tau(x) \quad \leftarrow \sqrt{x + s_{\mathbf{s}(x)}{}^{-1}}$
3. $a(x), b(x) \leftarrow \mathrm{EEA}(g(x), \tau(x))$ s.t. $b(x)\tau(x) \equiv a(x) \mod g(x)$
4. $\sigma(x) \quad \leftarrow a^2(x) + xb^2(x)$
5. $\boldsymbol{e} \quad \leftarrow (\sigma(x_1), \ldots, \sigma(x_n)) \oplus (1, \ldots, 1)$;

McEliece and Niederreiter Summary
Perspectives

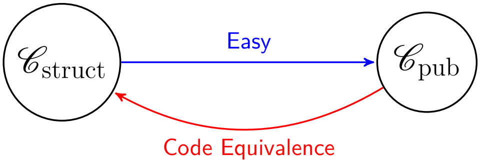
Summary

$\mathscr{C}_{\text{struct}}$

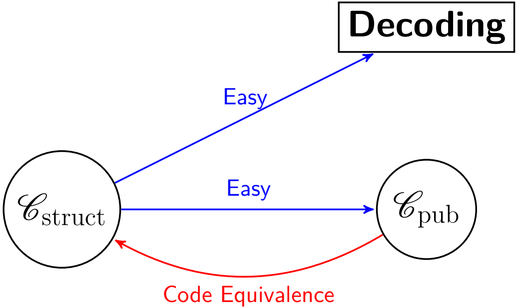
Summary

$\mathscr{C}_{\mathrm{struct}}$ $\mathscr{C}_{\mathrm{pub}}$

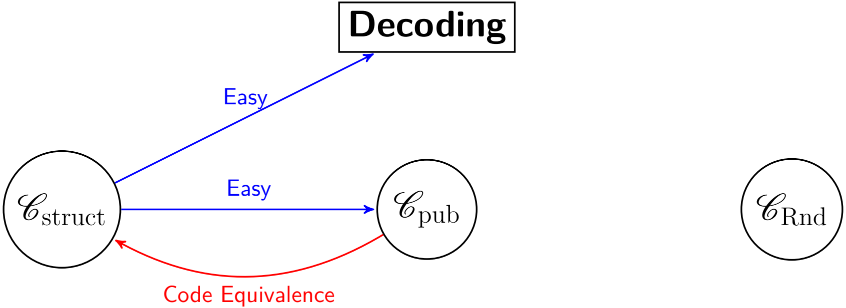
Summary


$\mathscr{C}_{\text{struct}}$
Easy
$\mathscr{C}_{\text{pub}}$
Code Equivalence

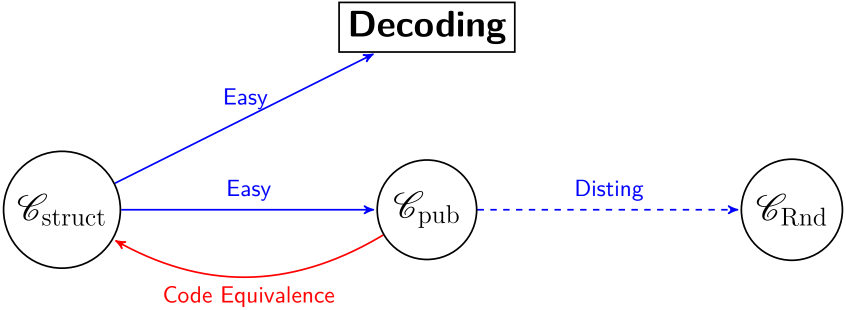
Summary


Decoding
Easy
Easy
$\mathscr{C}_{\mathrm{struct}}$
$\mathscr{C}_{\mathrm{pub}}$
Code Equivalence

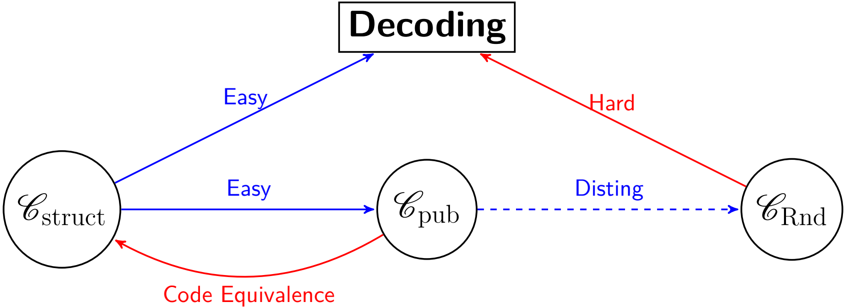
Summary


Decoding
Easy
Easy
$\mathscr{C}_{\text{struct}}$
$\mathscr{C}_{\text{pub}}$
$\mathscr{C}_{\text{Rnd}}$
Code Equivalence

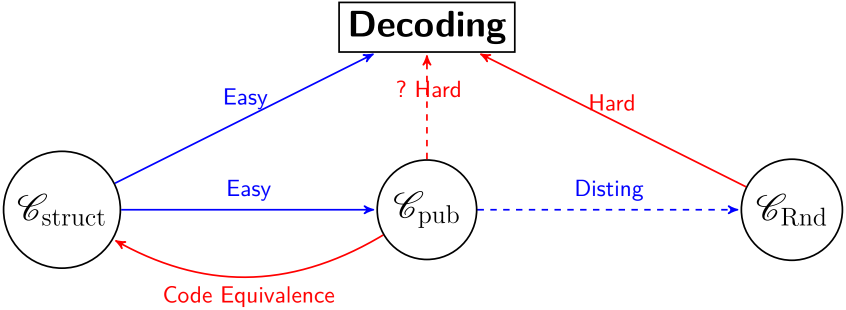
Summary


Decoding
Easy
Easy
$\mathscr{C}_{\text{struct}}$
$\mathscr{C}_{\text{pub}}$
Disting
$\mathscr{C}_{\text{Rnd}}$
Code Equivalence

Summary


Decoding
Easy
Hard
$\mathscr{C}_{\text{struct}}$
Easy
$\mathscr{C}_{\text{pub}}$
Disting
$\mathscr{C}_{\text{Rnd}}$
Code Equivalence

Summary


Decoding
Easy
? Hard
Hard
Easy
Disting
$\mathscr{C}_{\mathrm{struct}}$
$\mathscr{C}_{\mathrm{pub}}$
$\mathscr{C}_{\mathrm{Rnd}}$
Code Equivalence

Other constructions

Alekhnovich's Cryptosystems

- Underlying problem : distinguish a random vector from an erroneous codeword of a random code $\mathscr{C}$.

Other constructions

Alekhnovich's Cryptosystems

- Underlying problem : distinguish a random vector from an erroneous codeword of a random code $\mathscr{C}$.
- The public key is a random code while the private key is an error vector.

Other constructions

Alekhnovich's Cryptosystems

- Underlying problem : distinguish a random vector from an erroneous codeword of a random code $\mathscr{C}$.
- The public key is a random code while the private key is an error vector.
- Decryption is probabilistic

Alekhnovich's Cryptosystems

- *Key Generation*

Alekhnovich's Cryptosystems

- *Key Generation*

  1. Chose a random matrix $\boldsymbol{A} \in \mathcal{M}_{k,n}(\mathbb{F}_2)$

Alekhnovich's Cryptosystems

- *Key Generation*

  1. Chose a random matrix $\boldsymbol{A} \in \mathcal{M}_{k,n}(\mathbb{F}_2)$
  2. Choose $\boldsymbol{e} \in \mathbb{F}_2^n$ at random of weight t

Alekhnovich's Cryptosystems

- *Key Generation*

  1. Chose a random matrix $\boldsymbol{A} \in \mathcal{M}_{k,n}(\mathbb{F}_2)$
  2. Choose $\boldsymbol{e} \in \mathbb{F}_2^n$ at random of weight t
  3. Choose $\boldsymbol{x} \in \mathbb{F}_2^k$ at random

Alekhnovich's Cryptosystems

- *Key Generation*

  1. Chose a random matrix $\boldsymbol{A} \in \mathcal{M}_{k,n}(\mathbb{F}_2)$
  2. Choose $\boldsymbol{e} \in \mathbb{F}_2^n$ at random of weight t
  3. Choose $\boldsymbol{x} \in \mathbb{F}_2^k$ at random
  4. Compute $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A} + \boldsymbol{e}$ and $\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{y} \end{pmatrix}$

Alekhnovich's Cryptosystems

- *Key Generation*

  1. Chose a random matrix $\boldsymbol{A} \in \mathcal{M}_{k,n}(\mathbb{F}_2)$
  2. Choose $\boldsymbol{e} \in \mathbb{F}_2^n$ at random of weight t
  3. Choose $\boldsymbol{x} \in \mathbb{F}_2^k$ at random
  4. Compute $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A} + \boldsymbol{e}$ and $\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{y} \end{pmatrix}$
  5. Choose $\boldsymbol{G}$ a generator matrix for $\mathscr{C} = \ker(\boldsymbol{H})$.

Alekhnovich's Cryptosystems

- *Key Generation*

1. Chose a random matrix $\boldsymbol{A} \in \mathcal{M}_{k,n}(\mathbb{F}_2)$
2. Choose $\boldsymbol{e} \in \mathbb{F}_2^n$ at random of weight t
3. Choose $\boldsymbol{x} \in \mathbb{F}_2^k$ at random
4. Compute $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A} + \boldsymbol{e}$ and $\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{y} \end{pmatrix}$
5. Choose $\boldsymbol{G}$ a generator matrix for $\mathscr{C} = \ker(\boldsymbol{H})$.
6. The private key $\mathsf{sk} = (\boldsymbol{e})$ and the public key $\mathsf{pk} = (\boldsymbol{G}, t)$

Alekhnovich's cryptosystem

Encryption

Let $\boldsymbol{m} \in \mathbb{F}_2$,

1. If $\boldsymbol{m} = 0$ then
   - choose $\boldsymbol{a} \in \mathbb{F}_2^{n-k}$
   - choose $\boldsymbol{e}^{'} \in \mathbb{F}_2^n$ of weight t
   - send $\boldsymbol{c} = \boldsymbol{aG} + \boldsymbol{e}^{'}$
2. If $\boldsymbol{m} = 1$ then send a random vector $\boldsymbol{c} \in \mathbb{F}_2^n$

Alekhnovich's cryptosystem

Encryption

Let $\boldsymbol{m} \in \mathbb{F}_2$,

1. If $\boldsymbol{m} = 0$ then
   - choose $\boldsymbol{a} \in \mathbb{F}_2^{n-k}$
   - choose $\boldsymbol{e}' \in \mathbb{F}_2^n$ of weight t
   - send $\boldsymbol{c} = \boldsymbol{aG} + \boldsymbol{e}'$
2. If $\boldsymbol{m} = 1$ then send a random vector $\boldsymbol{c} \in \mathbb{F}_2^n$

Decryption

1. Compute $\boldsymbol{b} = \langle \boldsymbol{e}, \boldsymbol{c} \rangle$
2. If $\boldsymbol{m} = 0$ then $\boldsymbol{b} = 0$ w.h.p. $\quad \boldsymbol{b} = \langle \boldsymbol{e}, \boldsymbol{aG} \rangle + \langle \boldsymbol{e}, \boldsymbol{e}' \rangle = \langle \boldsymbol{e}, \boldsymbol{e}' \rangle$
3. If $\boldsymbol{m} = 1$ then $\boldsymbol{b} = 1$ w.p. $1/2$

Questions