

Code-based encryption schemes

Martin Stanek

Department of Computer Science
Comenius University
`stanek@dcs.fmph.uniba.sk`

Cryptology 1 (2020/21)

Content

Coding Theory

- Linear codes

- Generator matrix, Parity-check matrix

- Hamming code

Cryptosystems

- McEliece

- Niederreiter's variant

Concluding remarks

Coding theory – basics

- ▶ motivation: detect and correct errors in data; compress data
- ▶ important classes of error-correcting codes
 - ▶ linear codes
 - ▶ convolution codes
- ▶ some problems in coding theory are hard
 - ▶ possible use for cryptographic schemes
- ▶ some notation:
 - ▶ \mathbf{F}_q – finite field with q elements ($\text{GF}(q)$)
 - ▶ Hamming weight of a vector $x = (x_1, \dots, x_n) \in \mathbf{F}_q^n$:
 $\text{wt}(x) = |\{i; x_i \neq 0, 1 \leq i \leq n\}|$
 - ▶ Hamming distance of two vectors $x, y \in \mathbf{F}_q^n$:
 $\text{dist}(x, y) = |\{i; x_i \neq y_i, 1 \leq i \leq n\}|$

Linear codes

A q -ary linear $[n, k]$ code C is a k -dimensional subspace of \mathbf{F}_q^n .

- ▶ codewords – set of all elements in C
- ▶ n (length), k (dimension)
- ▶ generator matrix $G \in \mathbf{F}_q^{k \times n}$ of the code C : $C = \{xG; x \in \mathbf{F}_q^k\}$
- ▶ G describes an encoder for C : given $x \in \mathbf{F}_q^k$, codeword is xG
- ▶ systematic (standard) form of generator matrix $G = (\mathbf{I}_k \mid R)$
- ▶ distance of a linear code: $d = \min\{\text{wt}(c); c \in C \setminus \{0\}\}$
equivalently, $d = \min\{\text{dist}(b, c); b, c \in C\}$
- ▶ $[n, k, d]$ code
 - ▶ error $e \in \mathbf{F}_n^k$: $c \mapsto c + e$
 - ▶ can detect any error with $\text{wt}(e) \leq d - 1$
 - ▶ can correct any error with weight up to $\lfloor (d - 1)/2 \rfloor$

Parity-check matrix

- ▶ q -ary linear $[n, k]$ code C
- ▶ testing whether $c \in \mathbf{F}_q^n$ is a codeword of C (what linear relations must hold in the codeword)
- ▶ matrix $H \in \mathbf{F}_q^{(n-k) \times n}$, for any $c \in \mathbf{F}_q^n$: $cH^T = 0 \Leftrightarrow c \in C$
- ▶ H can be constructed easily from G given in a systematic form:

$$G = (\mathbf{I}_k \mid R) \Rightarrow H = (-R^T \mid \mathbf{I}_{n-k})$$

- ▶ we get: $GH^T = -R + R = 0$
- ▶ syndrome for any $x \in \mathbf{F}_q^n$: $s = xH^T$
 - ▶ $s = 0 \Leftrightarrow c \in C$
 - ▶ codeword c with an error e : $s = (c + e)H^T = eH^T$
 - ▶ syndrome decoding by lookup table of syndromes for all (viable) errors

Hamming (7, 4) code

- ▶ codeword length 7: 4 data bits, 3 parity bits
- ▶ linear code with distance 3, i.e. it corrects any single-bit errors
- ▶ generator matrix:

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

- ▶ encoding examples:
 - ▶ $(0, 0, 0, 0) \mapsto (0, 0, 0, 0, 0, 0, 0)$
 - ▶ $(1, 0, 0, 1) \mapsto (1, 0, 0, 1, 1, 0, 0)$
 - ▶ $(0, 0, 1, 1) \mapsto (0, 0, 1, 1, 0, 0, 1)$

Hamming (7, 4) code

- ▶ parity-check matrix (one of many):

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

- ▶ syndromes:

- ▶ $(0, 0, 1, 1, 0, 0, 1)H^T = (0, 0, 0)$
- ▶ $(\textcolor{red}{1}, 0, 1, 1, 0, 0, 1)H^T = (1, 0, 0)$
- ▶ $(0, 0, 1, \textcolor{red}{0}, 0, 0, 1)H^T = (0, 0, 1)$
- ▶ $(0, 0, 1, 1, 0, \textcolor{red}{1}, 1)H^T = (0, 1, 1)$
- ▶ $(0, 0, 1, 1, 0, 0, \textcolor{red}{0})H^T = (1, 1, 1)$

Binary BCH (31, 11) code

- ▶ codeword length 31: 11 bits of data, 20 checksum bits
- ▶ corrects up to 5 errors
- ▶ generator matrix (a cyclic code):

[illegible]

Some complexity problems

- ▶ random binary linear code
 - ▶ defined by a random generator/parity-check matrix (chosen uniformly)
 - ▶ optimal properties
 - ▶ decoding is hard
- ▶ decoding, i.e. for given H and syndrome s compute a minimum weight e such that $eH^T = s$, is NP-hard
- ▶ computing distance of a code is NP-hard
- ▶ worst-case complexity
- ▶ codes used in practice must have an efficient decoding algorithm
 - ▶ Reed-Solomon, Goppa, Reed-Muller, BCH, alternant, LDPC (Gallager), ...

McEliece cryptosystem

- ▶ Robert McEliece, 1978
- ▶ originally proposed with irreducible binary Goppa codes
- ▶ other codes can be used (be *very* careful – lots of broken proposals)
- ▶ initialization:
 1. select random binary linear $[n, k]$ code C that corrects up to t errors;
let G be a generator matrix for C
(C must have an efficient decoder $\mathcal{D} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^k$)
 2. select random $n \times n$ permutation matrix P
 3. select random $k \times k$ non-singular binary matrix S
 4. compute $G' = SGP$

private key: (G, S, P, \mathcal{D})
public key: (G', t)

McEliece cryptosystem – encryption and decryption

- ▶ encryption of plaintext $m \in \mathbf{F}_2^k$:
 1. choose random $e \in \mathbf{F}_2^n$ such that $\text{wt}(e) = t$
 2. ciphertext: $c = mG' + e$
- ▶ decryption of ciphertext c : $m = \mathcal{D}(cP^{-1})S^{-1}$
- ▶ correctness:
 - ▶ $cP^{-1} = (mSGP + e)P^{-1} = mSG + eP^{-1}$
 - ▶ $\text{wt}(eP^{-1}) = t$ (P is a permutation matrix)
 - ▶ $(mS)G$ is a codeword, and \mathcal{D} can correct up to t errors, therefore $\mathcal{D}(cP^{-1}) = mS$
 - ▶ finally, $(mS)S^{-1} = m$

Niederreiter's variant

- ▶ Harald Niederreiter, 1986
- ▶ variant of McEliece cryptosystem
 - ▶ equivalent security
 - ▶ faster decryption
 - ▶ smaller public key
- ▶ syndrome decoder computes e for given syndrome eH^T ($\text{wt}(e) \leq t$)
- ▶ initialization:
 1. select random binary linear $[n, k]$ code C that corrects up to t errors;
let H be a parity-check matrix for C
(C has an efficient syndrome decoder $\mathcal{D} : \mathbf{F}_2^n \rightarrow \mathbf{F}_2^n$)
 2. select random $n \times n$ permutation matrix P
 3. select random $(n - k) \times (n - k)$ non-singular binary matrix S
 4. compute $H' = SHP$

private key: (H, S, P, \mathcal{D})
public key: (H', t)

Niederreiter's variant – encryption and decryption

- ▶ plaintexts: $\{e \in \mathbf{F}_2^n ; \text{wt}(e) = t\}$
- ▶ encryption of plaintext $e \in \mathbf{F}_2^k$: $c = H'e^T$
- ▶ decryption of ciphertext c : $e = \mathcal{D}((S^{-1}c)^T) \cdot (P^T)^{-1}$
- ▶ correctness:
 - ▶ $(S^{-1}c)^T = (S^{-1}H'e^T)^T = (H(Pe^T))^T = (eP^T)H^T$
 - ▶ $\text{wt}(eP^T) = t$ (P is a permutation matrix)
 - ▶ \mathcal{D} computes eP^T , and e can be recovered: $eP^T \cdot (P^T)^{-1}$
- ▶ symmetric key transfer:
 - ▶ generate random e with $\text{wt}(e) = t$
 - ▶ symmetric key for encryption/authentication computed by hashing e

McEliece/Niederreiter – remarks

- ▶ very fast encryption (vector-matrix multiplication)
- ▶ fast decryption possible (e.g. binary.cr.yp.to/mcbits.html)
- ▶ two types of attacks:
 - ▶ generic attacks, e.g. information-set decoding
 - ▶ structural attacks (specific structure of the code)
- ▶ the main problem of these systems: key size
 - ▶ codes with shorter representation, e.g. Quasi-cyclic Moderate-Density Parity-Check (QC-MDPC) code

PQC Competition

- ▶ round 3 (2020): 3 code-based proposals (Encryption/KEM category)
 - ▶ Classic McEliece (finalist!) – binary Goppa codes, Niederreiter variant
 - ▶ merger of Classic McEliece and NTS-KEM

- ▶ Parameters for some of the proposed Classic McEliece instances:

security	n	m	$k = n - mt$	t	
128	3488	12	2720	64	<i>mciece348864</i>
192	4608	13	3360	96	<i>mciece460896</i>
256	6688	13	5024	128	<i>mciece6688128</i>

- ▶ Sizes of parameters for some of the proposed Classic McEliece instances (bytes):

security	public key	private key	ciphertext	
128	261 120	6 452	128	<i>mciece348864</i>
192	524 160	13 568	188	<i>mciece460896</i>
256	1 044 992	13 892	240	<i>mciece6688128</i>

Classic McEliece – remarks

- ▶ NIST Status Report on the 2nd Round:

Classic McEliece has a somewhat unusual performance profile—it has a very large public key but the smallest ciphertexts of all competing KEMs.

Goppa code McEliece has been a well-known construction for over 40 years with only incremental improvements on attacks.

Key encapsulation in Classic McEliece

- ▶ ...and OW-CPA \mapsto IND-CCA2 transformation
- ▶ H is a hash function
- ▶ Encapsulation and session key:
 - ▶ e is random with $\text{wt}(e) = t$
 - ▶ ciphertext $C = (C_0, C_1)$,
where C_0 is the public-key encryption of e , and $C_1 = H(2, e)$
 - ▶ session key $K = H(1, e, C)$
- ▶ Decapsulation for (C_0, C_1) :
 - ▶ set $b = 1$
 - ▶ decrypt C_0 to get e (if error: set $b = 0$ and $e = s$ for some s)
 - ▶ verify that $H(2, e) = C_1$ (if not: set $b = 0$ and $e = s$)
 - ▶ compute session key $K = H(b, e, C)$