Assignment 4

Final version 1.0 20/12/2023

Politecnico di Torino - Master of Science in Data Science and Engineering

Information Theory for Data Science
Prof. Roberto Garello
roberto.garello@polito.it

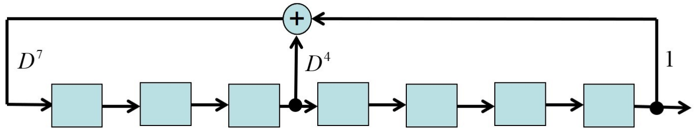
Assignment 4

- Exercise 1 - Stream Ciphers - pt. 15
- Exercise 2a - Mc Eliece cryptosystem - pt. 15
- Alternative: Exercise 2b - Signal/TLS - pt. 15

Exercise 1 - Stream Ciphers - pt. 15

Exercise 1 - LFSR M-sequences

Study the M-sequence generated by this Linear Feedback Shift Register (LFSR):



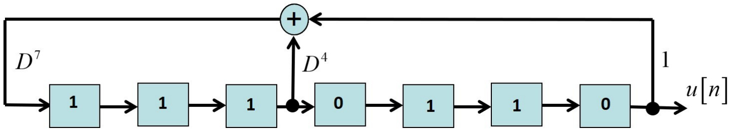
The LFSR is characterized by;

- m=7 cells
- polynomial $D^7 + D^4 + 1$

(The association between the feedback connections and the polynomial coefficients is not unique, the reverse order is used too. The proposed one is the association used by Matlab in its functions.)

Starting seed

Use the starting seed 1110110



The polynomial is primitive then the LFSR generates an m-sequence with

- period $N = 2^m - 1 = 127$ bits
- first bits = 0110111100...

Denote the binary sequence by $v(n)$ and the corresponding bipolar sequence $(0 \to -1,\ 1 \to +1)$ by $v'(n)$, for $0 \leq n \leq N - 1$.
When needed, consider them as the principal periods of periodic sequences.

0/1

1. (pt. 0.5) Generate the 127-bit M-sequence $v(n)$, compute the values of N_1 and N_0 (number of bits equal to 1 or 0 in $v(n)$), check if $N_1 = N_0 + 1$.
2. (pt. 1) Prove that for an M-sequence we always have $N_1 = N_0 + 1$.

Runs

3. (pt. 0.5) Write a table with the values of $NR_0(i)$ and $NR_1(i)$ (number of runs of i consecutive 0 or 1 symbols in $v(n)$).
4. (pt. 0.5) Compare the values of $NR(0)$ and $NR(1)$ against the run properties of an ideal binary random sequence.
5. (pt. 1) Prove that we cannot have a run of m zeros, or a run of $m+1$ ones.

Autocorrelation

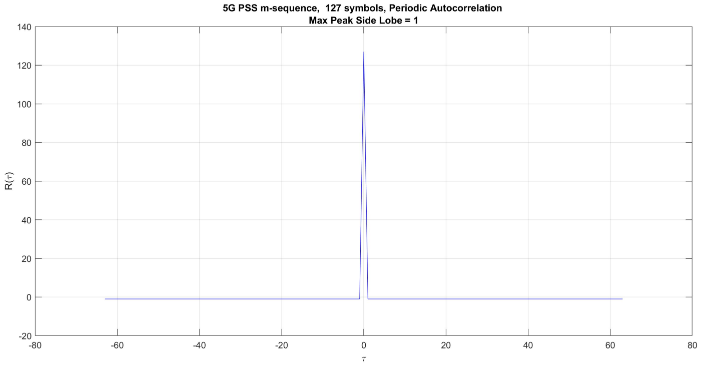
6. (pt. 0.5) Compute and plot the periodic autocorrelation function

$$R(\tau) = \sum_{n=0}^{N-1} v'(n)\, v'(n-\tau) \qquad -\frac{N-1}{2} \leq \tau \leq \frac{N-1}{2}$$

Verify the property:

- $R(\tau) = N$ for $\tau = 0$
- $R(\tau) = -1$ for $\tau \neq 0$

Autocorrelation plot



Matlab

LFSR

```
m=7; % number of cells
Nb=2∧m-1; % period
pnSequence = comm.PNSequence('Polynomial',[7 4 0], ...
'SamplesPerFrame',Nb,'InitialConditions',[1 1 1 0 1 1 0]);
x1 = pnSequence()';
```

Matlab

Autocorrelation

x1b=2*x1-1; % bipolar version 0 $\rightarrow$ -1 1 $\rightarrow$ +1
R=ifft(fft(x1b).*conj(fft(x1b))); % non-normalized periodic autocorr.

Breaking a linear LFSR

By a chosen plaintext attack, you manage to intercept this portion of the key: XXXXXXXXXX(40 bits)

I sent the string by email to all the students who delivered Assignment 1. If you did not receive the email, contact me (roberto.garello@polito.it).

7. (pt. 4) Compute the primitive polynomial of the LFSR
8. (pt. 4) Generate the next 20 bits of the key

Matlab

How to solve equations in GF(2)

R = rank(gf(A)); % computes the rank of the binary matrix A
[x,v] = gflineq(A,b,2); % solves the binary linear equation b=Ax
(If $v = 0$ the equation has no solution.)

SNOW

9. (pt. 3) Consider the SNOW stream cipher (choose a version), and prepare no more than 3 slides as if you had to present them to your colleagues: scheme, high-level description, no details, give some info on applications and how non-linearity is achieved, list the references used.

Exercise 2

You can choose between Exercise 2a (Mc Eliece cryptosystem) and Exercise 2b (Signal/TLS). (You cannot deliver both and skip exercise 1.)

Exercise 2a - McEliece cryptosystem - pt. 15

McEliece cryptosystem

The McEliece cryptosystem is a public key algorithm based on error correcting codes.

The security of the algorithm is based on the complexity of decoding a generic linear code.

Currently, the McEliece scheme is one of the four schemes admitted at round 4 of NIST (National Institute of Standards and Technology) Post-Quantum Cryptography Standardization competition.

Summary

Given a linear block code $C(n, k)$ able to correct t errors per block, the user U_1 public key is (G_1, t_1), where G_1 is an obfuscated generator matrix $G_1 = SGP$ (S a non-singular matrix, G a generator matrix of C and P a permutation matrix).

If the user U_2 wants to sends a message to U_1, it represents the message by a vector $\underline{v}$ of k-bits, encodes it by G_1 to obtain $\underline{c} = \underline{v}G_1$ and adds t_1 random errors to obtain the transmitted vector $\underline{y}$.

The user U_1 decodes $\underline{y}$ by applying a low-complexity decoding algorithm designed for C and recover the message vector $\underline{v}$.

Thanks to obfuscation, the true code is unknown to the attacker. The decoding of a generic linear code is known to be NP-HARD then it becomes unfeasible if k and n are large.

Binary Alphabet

We represent the binary alphabet by the symbol $F = \{0, 1\}$ (or GF(2)).
The binary sum has addition table:

$+$	0	1
0	0	1
1	1	0

The binary multiplication sum has multiplication table

$\cdot$	0	1
0	0	0
1	0	1

With these two operations, $(F, +, \cdot)$ is a field.

Vector of bits

We represent the set of all possible k-bit vectors as

$$F^k = \{\underline{v} = (v_1, \cdots, v_i, \cdots, v_k) \;\; v_i \in F = \{0, 1\}\}$$

(ex: $F^2 = \{(00), (01), (10), (11)\}$)
Given two binary vectors we define their sum as the bit-by-bit sum:

$$\underline{v} = (v_1, \cdots, v_i, \cdots, v_k) \in F^k \quad \underline{w} = (w_1, \cdots, w_i, \cdots, w_k) \in F^k$$

$$\underline{a} = \underline{v} + \underline{w} = (v_1 + w_1, \cdots, v_i + w_i, \cdots, v_k + w_k) \in F^k$$

(ex $11 + 10 = 01$)
and the multiplication of a vector by a bit as:

$$b \in F = \{0, 1\} \quad \underline{v} = (v_1, \cdots, v_i, \cdots, v_k) \in F^k$$

$$\underline{c} = b \cdot \underline{v} = (b \cdot v_1, \cdots, b \cdot v_i, \cdots, b \cdot v_k) \in F^k$$

(ex $1 \cdot 11 = 11$ and $0 \cdot 11 = 00$)

Vector space and group properties

It is easy to show that F^k is a vector space.
In particular, $(F^k, +)$ is a commutative group:
Closure:

$$\forall \underline{v}, \underline{w} \in F^k \;\; \underline{v} + \underline{w} \in F^k$$

Sum identity:

$$\underline{0} = (0, \cdots, 0, \cdots, 0) \in F^k \;\; \forall \underline{v} \in F^k \;\; \underline{v} + \underline{0} = \underline{v}$$

Additive inverse:

$$\forall \underline{v} \in F^k \;\; \underline{v} + \underline{v} = \underline{0}$$

Associative property

$$\forall \underline{v}_1, \underline{v}_2, \underline{v}_3 \in F^k \;\; \underline{v}_1 + (\underline{v}_2 + \underline{v}_3) = (\underline{v}_1 + \underline{v}_2) + \underline{v}_3$$

Commutative property

$$\forall \underline{v}_1, \underline{v}_2 \in F^k \;\; \underline{v} + \underline{w} = \underline{w} + \underline{v}$$

Code $C(n, k)$

A code $C(n, k)$ is characterized by two parameters

- $k =$ information vector length
- $n =$ codeword length

The encoder maps any k-bit information vector $\underline{v} \in F^k$ into an n-bit codeword $\underline{c} \in F^n$:

$$\begin{array}{cccc} e: & F^k & \longrightarrow & F^n \\ & \underline{v} & \longrightarrow & \underline{c} \end{array}$$

The map is linear and one-to-one, then it can be represented by a $k \times n$ generator matrix G, with rank k:

$$\underline{c} = \underline{v} G$$

Example

$C(n = 7, k = 4)$

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

$$\underline{v} = (1000)$$

$$\underline{c} = \underline{v}G = (1000111)$$

Codebook

Given a code $C(n, k)$ the codebook is the set of all possible n-bit codewords $\underline{c} \in F^n$ obtained by encoding all the possible 2^k information vectors $\underline{v} \in F^k$. Since the matrix G has full rank, all the codewords are distint and the codebook cardinality is 2^k, too.

$$C = \left\{ \underline{c} = \underline{v}G \;\; \forall \underline{v} \in F^k \right\}$$

$$|C| = 2^k$$

Example

In the previous example $C(n = 7, k = 4)$

$$\begin{array}{ccc}
0000 & \longrightarrow & 0000000 \\
0001 & \longrightarrow & 0001000 \\
0010 & \longrightarrow & 0010000 \\
0011 & \longrightarrow & 0011000 \\
0100 & \longrightarrow & 0100000 \\
0101 & \longrightarrow & 0101000 \\
0110 & \longrightarrow & 0110000 \\
1000 & \longrightarrow & 1000000 \\
1001 & \longrightarrow & 1001000 \\
1010 & \longrightarrow & 1010000 \\
1011 & \longrightarrow & 1011000 \\
1100 & \longrightarrow & 1100000 \\
1101 & \longrightarrow & 1101000 \\
1110 & \longrightarrow & 1110000 \\
1111 & \longrightarrow & 1111000
\end{array}$$

Hamming weight and Hamming distance

The Hamming weight w_H of a binary vector of F^n is the number of bits equal to one. The Hamming distance d_H between two binary vectors of F^n is the number of bits where they are different. It is easy to show that

$$d_H\left(\underline{c_1}, \underline{c_2}\right) = w_H\left(\underline{c_1} + \underline{c_2}\right)$$

ex $w_H(101) = 3$
ex $d_H(101, 110) = w_H(011) = 2$

Minimum distance of a code

The minimum distance of a code is the minimum Hamming distance between different codewords

$$d_{\min} = \min_{\left[\forall \underline{c}_1, \underline{c}_2 \in C \quad \underline{c}_1 \neq \underline{c}_2\right]} d_H\left(\underline{c}_1, \underline{c}_2\right)$$

It is easy to show that it can be computed as teh minimum Hamming weight of a non-zero codeword

$$d_{\min} = \min_{\left[\forall \underline{c} \in C \quad \underline{c} \neq \underline{0}\right]} w_H\left(\underline{c}\right)$$

Error vector

Suppose to transmit a codeword $\underline{c} \in C$ and to receive a vector $\underline{y}$ where, for some reason (noise, intentionally added errors, ...) some of the bits are received wrong, i.e., inverted.
We can represent the wrong bits by an error vector $\underline{e} = (e_1, \cdots, e_i, \cdots e_n) \in F^n$ where $e_i = 0$ if $y_i = c_i$ and $e_i = 1$ if $y_i \neq c_i$.
Example: $\underline{c} = (1000), \underline{y} = (0101), \underline{e} = (1101)$.

Minimum distance decoding rule

Given $\underline{y}$ we can choose the codeword that minimizes the error probability by the minimum distance decoding rule:

$$\hat{\underline{c}} = arg \min_{\underline{c} \in C} d_H \left(\underline{y}, \underline{c} \right)$$

Unfortunately, the complexity of this decoding rule is proportional to the code cardinality 2^k and becomes rapidly unfeasible for non-trivial k.

Error correction capability

The minimum distance of the code determines the number of errors that a code is certainly able to correct by a minimum distance rule:

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

Parity check matrix

Given a code $C(n,k)$ we define $r = n - k$.
Given an n-bit vector $\underline{y}$ we want to establish a test to determine if it belongs to the code or not. Let us introduce an $n \times r$ binary matrix H and consider this linear function:

$$\begin{array}{cccc} f: & F^n & \longrightarrow & F^r \\ & \underline{y} & \longrightarrow & \underline{s} = \underline{y}H \end{array}$$

We call the r-bit vector $\underline{s} \in F^r$ the syndrome of $\underline{y}$.
We want to build a matrix H such that $\underline{s} = \underline{0}$ if and only if $\underline{y} \in C$.
It is easy to show that this matrix must have these two properties:

- rank(H)=r
- $GH = 0$

Syndrome decoding

The syndrome decoding is a simplified decoding algorithm that we can apply when we know the parity check matrix H of the code.
First we note that, given $\underline{y} = \underline{c} + \underline{e} \Longrightarrow \underline{s} = \underline{y}H = \underline{e}H$, i.e., the syndrome only depends on the error vector.
Now, suppose we want to correct all the error vectors up to weight t. For each of them we precompute the syndrome and we store it into a LookUpTable (LUT) mapping each syndrome into the corresponding error vector.
At the received side, given $\underline{y}$ we compute the syndrome $\underline{s} = \underline{y}H$.
We check if $\underline{s}$ is listed in the LUT: in this case we get the corresponding error vector $\underline{e}$ and we build the received vector as

$$\underline{\hat{c}} = \underline{y} + \underline{e}$$

McEliece cryptosystem

The user U_1 selects a $k \times n$ generator matrix G.
It obfuscates it by computing

$$G_1 = SGP$$

where

- S is a non-singular (full rank) $k \times k$ binary matrix
- P is an $n \times n$ permutation matrix

Note that if k and n are big enough, it is computationally hard to identify the true generator matrix G.
The public key is (G_1, t_1) where $t_1 \leq t$ is the number of errors that U_1 wants to correct.

Encryption

The user U_2 partition its message into k-bit vectors $\underline{v}$, then for each vector $\underline{v}$:

- Computes $\underline{c} = \underline{v}G_1$
- Generates a random error vector $\underline{e}$ with t_1 errors: $w_H(\underline{e}) = t_1$.
- Computes $\underline{y} = \underline{c} + \underline{e}$: this is the ciphertext sent to U_1.

Decryption

The user U_1 receives $\underline{y}$ then

- Generates $\underline{y}_1 = \underline{y}P^{-1}$.
- Decodes $\underline{y}_1$ to obtain $\underline{c}_1$
- From $\underline{c}_1 = G\underline{v}_1$ recovers $\underline{v}_1$
- Generates $\underline{\hat{v}} = \underline{v}_1 S^{-1}$.

Finally we have $\underline{\hat{v}} = \underline{v}$ and the original message is retrieved.

Exercise 4.2a - Mc Eliece cryptosystem

You will receive by email:

- the generator matrix G
- the parity check matrix H
- the non-singular matrix S
- the permutation matrix P
- the number of errors $t_1 = 1$
- a sequence of ciphertexts corresponding to a message

Exercise 4.2: ASCII

The message is the name of a city. Each capital letter is encoded into an 8-bit vector by using ASCII Windows-1252 encoding according to this table:

letter	ASCII code	binary	letter	ASCII code	binary
A	65	01000001	N	78	01001110
B	66	01000010	O	79	01001111
C	67	01000011	P	80	01010000
D	68	01000100	Q	81	01010001
E	69	01000101	R	82	01010010
F	70	01000110	S	83	01010011
G	71	01000111	T	84	01010100
H	72	01001000	U	85	01010101
I	73	01001001	V	86	01010110
J	74	01001010	W	87	01010111
K	75	01001011	X	88	01011000
L	76	01001100	Y	89	01011001
M	77	01001101	Z	90	01011010

Be careful to the order of the binary vector in your program. Is 'A' encoded into 01000001 or into 10000010? Note that the first (or last) bit of these capital letter is always zero.

Exercise 4.2a: Suggestion

We strongly suggest to implement the entire McEliece scheme (encryption + decryption) and test the toy example presented on the handwritten notes.

Exercise 4.2a: LUT

Since (in this simplified example) we have $t_1 = 1$ you must consider $n + 1$ error vectors $\underline{e}$:

- the all-zero error vector
- the n error vectors of weight 1

For each error vector compute the corresponding syndrome $\underline{s} = \underline{e}H$ and store all the syndromes/error vectors pairs.

Exercise 4.2a: McEliece decryption

- Divide your string into n-bit vectors $\underline{y}$
- Compute $\underline{y}_1 = \underline{y}P^{-1}$
- Decode $\underline{y}_1$ to obtain $\underline{v}_1$ by applying syndrome decoding

Exercise 4.2a: Syndrome decoding

- Compute the syndrome $\underline{s} = \underline{y}_1 H$
- Obtain the corresponding error vector $\underline{e}$ from the LUT
- Obtain the received codeword $\underline{c}_1 = \underline{y}_1 + \underline{e}$
- Obtain the corresponding information vector $\underline{v}_1$.

Exercise 4.2a: Final result

- Obtain $\underline{\hat{v}} = \underline{v}_1 S^{-1}$
- Collect together all information vectors $\underline{\hat{v}}$
- Apply the ASCII decoding to obtain the name of the city

Exercise 4.2a. Program

- (pt. 13) Write a program to implement the deciphering of the McEliece scheme to recover the transmitted message.

Important

In the presentation, write:

- All the received inputs (string, G, H, ...)
- All the matrices you computed to solve the problem (S^{-1}, P^{-1})
- The LUT
- The name of the city

Exercise 4.2a: Question

3. (pt. 2) Which values of k and n are considered for practical applications of the McEliece scheme? Add the **references** where you found the results.

Matlab

ASCII

```
lett=binaryVectorToDecimal(flip(vett));
L(i)=char(lett);
```

Inverse of a binary matrix

```
inv(gf(S))
```

Recover the information vector from the codeword

```
gflineq(G',c',2)
```

Exercise 4.2b

Important: this is an alternative to Exercise 4.2a. You cannot solve both of them.

Exercise 4.2b: Signal/TLS

1. (pt. 7.5) The Signal Protocol (WhatsApp encryption): 1. what is it, 2. how the keys are generated and exchanged, 3. which algorithms are used for encryption (no detailed description of the algorithm, but write on which technique they are based on (e.g., prime factorization or discrete log or elliptic curves or ...), 4. which modes are used, 5. how group chats are managed, 6. role of symmetric and asymmetric encryption.
2. (pt.7.5) The TLS Protocol 1.3 (https encryption): 1. what is it, 2. how the keys are generated and exchanged, 3. which algorithms are used for encryption (no detailed description of the algorithm, but write on which technique they are based on (e.g., prime factorization or discrete log or elliptic curves or ...), 4. which modes are used, 5. role of symmetric and asymmetric encryption.

Exercise 4.2b

Important:

- Max 5 pages for each question.
- Properly cite all the used references

Important
Delivery by

- January, 21-th, 11.59 PM: $+$ 2 points
- January, 28-th, 11.59 PM: $+1$ point
- February, 4-th, 11.59 PM: 0 points
- Later: not accepted