Jornada de Teoria de Nombres

Universitat de Lleida

Alternant codes and the McEliece cryptosystem

S. Xambó & N. Sayols

UPC

7/10/2017

2003: S. Xambó, *Block-error correcting codes: a computdational primer*. Universitext, Springer. Computational system: WIRIS/CC.

2015—: **PyECC****. "Python package enabling the construction, coding and decoding of error-correcting codes, to be freely available for teachers and researchers". The plan is to go beyond CC and be able to work as easily with convolutional codes and other close classes as easily as with algebraic block codes.

2017: Farré-Sayols-Xambó, *On PGZ decoding of alternant codes*. Improved version of the classical PGZ decoding algorithm. **arXiv**.

2017: Molina-Sayols-Xambó, *A bootstrap for the number of $\mathbb{F}_{q^m}$-rational points on a curve over $\mathbb{F}_q$*. **arXive**.

** https://mat-web.upc.edu/people/sebastia.xambo/PyECC.html

2017: *Error-correcting codes: mathematics and computations* (SGA UB-UAB-UPC, 5.5.2017).

All algorithms implemented in PyECC, including the best known decoding algorithms for alternant codes.

Immediately after we began looking at the McEliece cryptosystem based on (classical) Goppa codes (McEliece, 1978):

A public-key cryptosystem based on algebraic coding theory.

Context of that paper:

- *The algebraic decoding of Goppa Codes* (Patterson, 1974);
- *New directions in cryptography* (Diffie-Hellman, 1976);
- *RSA*, 1978;
- *On the inherent intractability of certain coding problems* (Berlekamp-McEliece-van Tilborg, 1978).

Using the fact that a fast decoding algorithm exists for a general [classical] Goppa code, while no such algorithm exists for a general linear code, we construct a public-key cryptosystem which appears quite secure while at the same time allowing extremely rapid data rates. This kind of cryptosystem is ideal for use in multi-user communication networks, such as those envisioned by NASA for the distribution of space-acquired data.

More recent references: Proceeings PQCrypto 2008, and in particular *Attacking and defending the McEliece crytosystem* (Bernstein-Lange-Peters,2008) (**pdf**).

Quantum Resistant Random Linear Code Based Public Key Encryption Scheme RLCE (Wang, 2015)(**pdf**)

Wikipedia

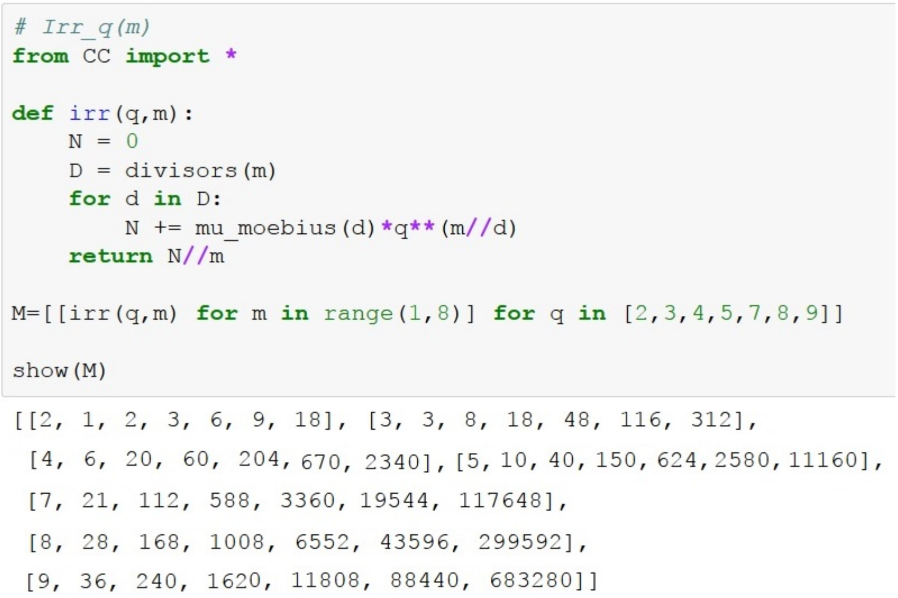
- Finite fields.
- Linear codes.
- Coding, decoding, and error-correcting capacity.
- Binary Goppa codes. Decoding algorithms.
- Private and public McEliece keys. Encryption and Decryption protocols.
- Comments on the analysis of the McEliece system.
- Code samples.
- A quotation.

- $\mathbb{F}_q = GF(q)$: finite field of $q = p^r$ elements (p prime, $r \geqslant 1$).
- The number of monic irreducible polynomials of degree m over $\mathbb{F}_q$ is given by *Gauss' formula*:

$$I_q(m) = \tfrac{1}{m} \sum_{d|m} \mu(m/d) q^d = \tfrac{q^m}{m} + \cdots$$

  Application: the probability that a random monic polynomial of degree m over $\mathbb{F}_q$ is prime is $\simeq 1/m$.

```
1     2     3     4      5       6       7
---------------------------------
2     1     2     3      6       9       18
3     3     8     18     48      116     312
4     6     20    60     204     670     2340
5     10    40    150    624     2580    11160
7     21    112   588    3360    19544   117648
8     28    168   1008   6552    43596   299592
9     36    240   1620   11808   88440   683280
```



```
# Irr_q(m)
from CC import *

def irr(q,m):
    N = 0
    D = divisors(m)
    for d in D:
        N += mu_moebius(d)*q**(m//d)
    return N//m

M=[[irr(q,m) for m in range(1,8)] for q in [2,3,4,5,7,8,9]]

show(M)
```

```
[[2, 1, 2, 3, 6, 9, 18], [3, 3, 8, 18, 48, 116, 312],
 [4, 6, 20, 60, 204, 670, 2340], [5, 10, 40, 150, 624, 2580, 11160],
 [7, 21, 112, 588, 3360, 19544, 117648],
 [8, 28, 168, 1008, 6552, 43596, 299592],
 [9, 36, 240, 1620, 11808, 88440, 683280]]
```

- An $[n,k]$ code over $\mathbb{F}_q$ is k-dimensional linear subspace C of $\mathbb{F}_q^n$ (n is the *length*, k the *dimension*, and $R = k/n$ the *rate*). We write $C \sim [n,k]$.
- A matrix G whose rows are a basis of C is called a *generating matrix* of C. It is an $k \times n$ matrix and we may write $C = \langle G \rangle$.
- The linear map $f : \mathbb{F}_q^k \to \mathbb{F}_q^n$, $\boldsymbol{u} \mapsto \boldsymbol{u}G$ is injective and its image is C. We say that f is an *encoding* for C, and $\boldsymbol{x} = f\boldsymbol{u}$ is called the *code vector* of $\boldsymbol{u}$.

Example (The [7,4] Hamming binary encoder)

$$\boldsymbol{x} = \boldsymbol{u} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Let H be an $r \times n$ matrix of rank r. Then $C_H = \{\boldsymbol{x} \in \mathbb{F}^n : \boldsymbol{x} H^T = 0\}$ is an $[n, n-r]$ code. The matrix H is called a control matrix of C_H.

A generator matrix of C_H is obtained by choosing a basis of $\ker H^T$.

Example. The matrix $H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$ is a control matrix of the Hamming $[7, 4]$ code.

Indeed, $G = I_4|R$, $H = R^T|I_3$ and $GH^T = (I_4|R)\left(\frac{R}{I_3}\right) = R + R = 0$.

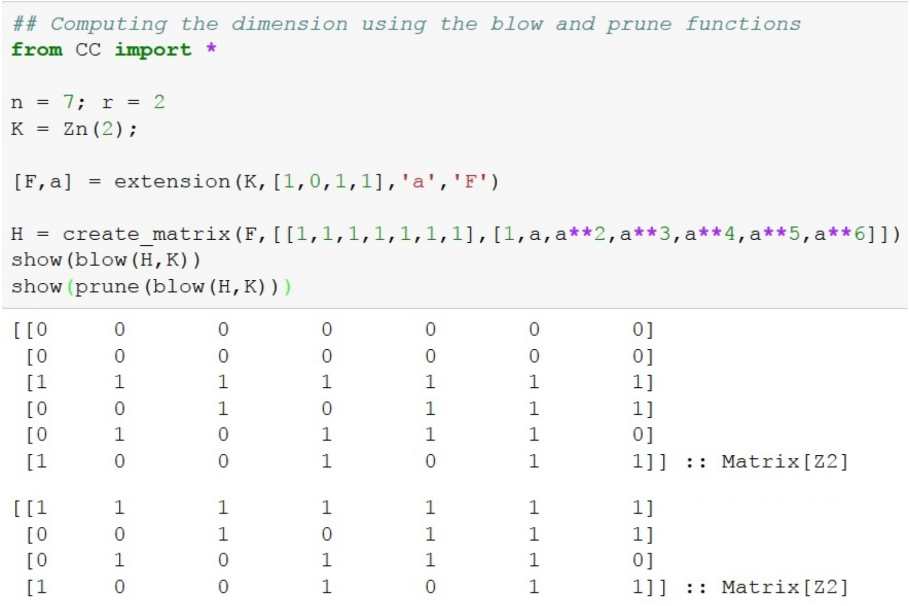
Remark. A generating matrix of the form $G = I_k|R$ is called *systematic* (with respect to the components $1, \ldots, k$). In this case, $H = (-R^T)|I_{n-k}$ is a control matrix of $\langle G \rangle$ ($H = R^T|I_{n-k}$ in the binary case).

Let H be an $r \times n$ matrix of rank r with entries in an extension field $\bar{K} = \mathbb{F}_{q^m}$ of $K = \mathbb{F}_q$. Then we can still define

$$C_H = \{\boldsymbol{x} \in \mathbb{F}^n : \boldsymbol{x} H^T = 0\},$$

but of its dimension k we only know, in general, a lower bound: $k \geqslant n - rm$.

Indeed, to obtain a generating matrix, we have to expand H into an $rm \times n$ matrix $\bar{H}$ by substituting each entry by its expression as a length m vector with entries in K, and take its kernel: its dimension is $k = n - \mathrm{rank}(\bar{H}) \geqslant n - rm$.



```
## Computing the dimension using the blow and prune functions
from CC import *

n = 7; r = 2
K = Zn(2);

[F,a] = extension(K,[1,0,1,1],'a','F')

H = create_matrix(F,[[1,1,1,1,1,1,1],[1,a,a**2,a**3,a**4,a**5,a**6]])
show(blow(H,K))
show(prune(blow(H,K)))
```

```
[[0      0      0      0      0      0      0]
 [0      0      0      0      0      0      0]
 [1      1      1      1      1      1      1]
 [0      0      1      0      1      1      1]
 [0      1      0      1      1      1      0]
 [1      0      0      1      0      1      1]] :: Matrix[Z2]

[[1      1      1      1      1      1      1]
 [0      0      1      0      1      1      1]
 [0      1      0      1      1      1      0]
 [1      0      0      1      0      1      1]] :: Matrix[Z2]
```

A *decoder* is a surjective map $g : \mathbb{F}^n \to C \sqcup \mathcal{E}$, where $\mathcal{E}$ is a set of *error messages*, such that $g(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in C$.

The set $D = g^{-1}(C) \subseteq \mathbb{F}^n$ is the set of g-*decodable vectors*. The set $\mathbb{F}^n - D = g^{-1}\mathcal{E}$ is the set of *error vectors*. The decoder is *complete* if $D = \mathbb{F}^n$ (in which case $\mathcal{E} = \emptyset$).

The decoder has *error-correcting capacity* t if we have $g(\boldsymbol{y}) = \boldsymbol{x}$ for any vector $\boldsymbol{y} \in \mathbb{F}^n$ such that $|\boldsymbol{y} - \boldsymbol{x}| \leqslant t$, where $|\boldsymbol{z}|$ is the number of non-zero components of $\boldsymbol{z}$ (*Hamming weight* of $\boldsymbol{z}$).

If we let $B(\boldsymbol{x}, t) = \{\boldsymbol{y} \in \mathbb{F}^n : |\boldsymbol{y} - \boldsymbol{x}| \leqslant t\}$ (*Hamming ball of center* $\boldsymbol{x}$ *and radius* t), then we have $\cup_{\boldsymbol{x} \in C} B(\boldsymbol{x}, t) \subseteq D$ and $g(B(\boldsymbol{x}, t)) = \{\boldsymbol{x}\}$.

- $K = \mathbf{Z}_2$, $\bar{K} = \mathbb{F}_{2^m}$, $g \in \bar{K}[T]$ irreducible polynomial of degree t.
- Let $n = 2^m$, $\alpha_1, \ldots, \alpha_n$ the elements of $\bar{K}$, and $h_j = 1/g(\alpha_j)$.

  Then $\Gamma_g = C_H$, where $H = \begin{pmatrix} h_1 & \cdots & h_n \\ h_1\alpha_1 & \cdots & h_n\alpha_n \\ \vdots & & \vdots \\ h_1\alpha_1^{t-1} & \cdots & h_n\alpha_n^{t-1} \end{pmatrix}$, is the (classical) *Goppa code* associated to g. We have seen that $\dim \Gamma_g \geqslant n - tm$.

Theorem. Γ_g can correct t (or fewer) errors.

Remark. Γ_g is a special case of *alternant code* and so it has efficient decoders, as BMS (Berlekamp-Massey-Sugiyama, $O(nt)$) and PGZ (Petterson-Gorenstein-Zierler).

- Select m and t.
- Pick g at random. This is done by repeatedly choosing at random a monic polynomial $g \in \bar{K}[T]$ of degree t until g is irreducible. We know that on average t trials will be sufficient. Testing for irreducibility is done by means of the following criterion:

  g is irreducible if and only if $T^{2^t} \mod g = T$ and $\gcd(T^{2^{t/d}} - T, g) = 1$ for all $d|t$, d prime.
- Let G be a generating matrix of Γ_g, S a random non-singular $k \times k$ matrix, and P an $n \times n$ permutation matrix.
- *Private key*: G, S, and P.
- *Public key*: SGP and t.

Encryption:

- Let $\boldsymbol{u} \in \mathbb{F}^k$ (*information vector*)
- Generate a random vector $\boldsymbol{e}$ of weight t.
- Send $\boldsymbol{x} = \boldsymbol{u}SGP + \boldsymbol{e}$.

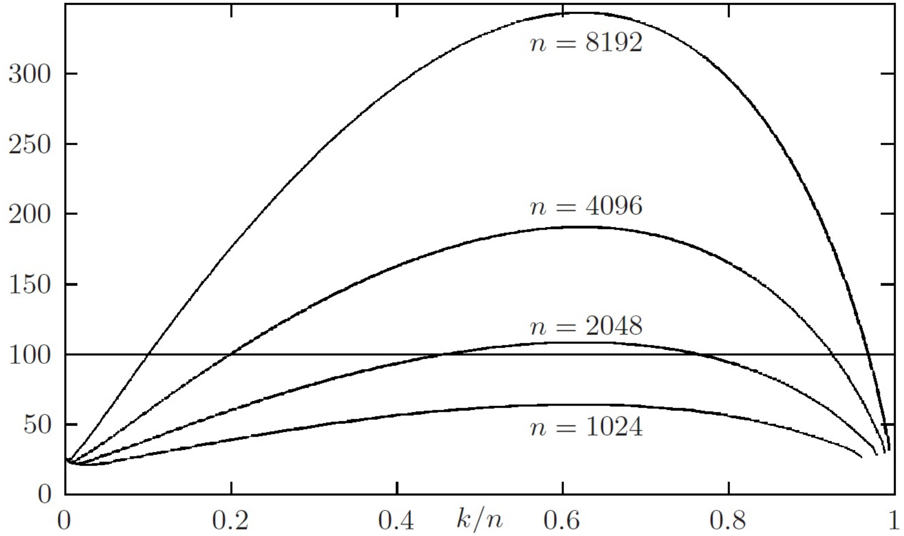
Decryption:

- $\boldsymbol{y} = \boldsymbol{x}P^{-1} = \boldsymbol{u}SG + \boldsymbol{e}P^{-1}$.
- The decoding of $\boldsymbol{y}$ produces the code vector $\boldsymbol{x}' = \boldsymbol{u}SG$.
- Solve this linear relation for $\boldsymbol{u}S$.
- Get $\boldsymbol{u} = (\boldsymbol{u}S)S^{-1}$.

We assume that E, the eavesdropper, knows not only t and SGP, but also $\boldsymbol{x}$ (after all this is the purpose of evesdropping!).

- The general problem of decoding $[n, k]$ codes is NP complete (Berlekamp-McElice-van Tilborg, 1978). So finding $\boldsymbol{u}$ from $\boldsymbol{x}$ and SGP is hard if $[n, k]$ are large enough.
- Select k coordinates from $\boldsymbol{x}$ at random. If they are not in error, then we get $\boldsymbol{u}$ with about k^3 operations. Since the probability that there are no errors is $(1 - t/n)^k$, the number of expected operations to succeed is $k^3(1 - t/n)^{-k}$.

Binary work factor (log scale)



PyECC

```
def permutation_matrix(n):
    N = list(range(n))
    p = rd_choice(N,n)
    P = create_matrix(ZZ(),n,n)
    for j in range(n):
        P[j,p[j]] = 1
    return P
```

```
def scramble_matrix(A,k):
    U = create_matrix(A,k,k)
    L = create_matrix(A,k,k)
    for i in range(k):
        U[i,i] = L[i,i] =1
        for j in range(i+1,k):
            U[i,j] = rd(A)
            L[j,i] = rd(A)
    return U*L
```

```
F5 = Zn(5)
# Creation of F25, with generator x
[F25,x] = extension(F5,[1,0,-2],'x','F25')
# Creation of the polynomial ring F25[T]
[A,T] = polynomial_ring(F25,'T')
g = T**6 + T**3 + T +1
a = Set(F25)[1:] # The non-zero elements of F25
a = [t for t in a if evaluate(g,t)!=0]
C = Goppa(g,a)
# generate a random error pattern of weight 3
e = rd_error_vector(Z5,n,3)
>> e = [0,1,0,0,0,3,0,4,0,0,0,0,0,0,0,0,0,0,0]
# Use the PGZ decoder for C
PGZ(e,C)
>>PGZ: Error positions [1,5,7], error values [1,3,4]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] :: Vector[Z5]
```

The McEliece cryptosystem was proposed by McEliece in 1978 [10] and the original version, using Goppa codes, remains unbroken. Quantum computers do not seem to give any signicant improvements in attacking code-based systems, beyond the generic improvements possible with Grover's algorithm, and so the McEliece encryption scheme is one of the interesting candidates for post-quantum cryptography. Bernstein-Lange-Peters-2008.