
Cryptography Engineering : Asymmetric Crypto

LECTURE 2 :

Poly-logarithm notations :

f (n) = O~(g(n)) iff f(n) = O(g(n) log^e g(n)) for some e > 0

Algo cost :

s = size of the input = log n (en base 2)

- Addition : O(s)
- Multiplication : $O(s^2)$ ou O~(s)
- Division : O~(s)
- GCD :

  euclidean algo : $O(s^2)$

  fast eucli algo : O~(s)

Evaluation interpolation :

: 2 points nécessitent un polynôme de degré 1 (une droite),

: 3 points nécessitent un polynôme de degré 2 (une parabole),

: n points nécessitent un polynôme de degré n-1

C'est un résultat fondamental en interpolation polynomiale : pour n points, il existe un unique polynôme de degré n−1 qui passe par ces points

FFT ⇔ Fast fourier transform

Permet de multiplier rapidement des nombres de grandes tailles.

La multiplication de 2 grands entiers peut être vu comme la multiplication de polynome.

O(n log n).

GCD ⇔ Greatest common divisor

LE GCD de a et b est le plus grand entier g qui divise à la fois a et b.

Si GCD(a,b) = 1 ← a et b cont coprimes

Bezout relation :

Si g = GCD(a,b), alors il existe u et v (entiers) coprimes tels que : g = ua + vb

properties :

GCD(a, b) = GCD(a, a – b))

GCD(a, b) = GCD(a, a mod b))

Z/nZ (finite ring and field)

Integers modulo n, équipé de l'addition et de la multiplication modulo n. Contient contient {0, 1, . . . , n − 1}.

Modular inverse : a^-1 mod n

PGCD(a,n) = 1 ⇔ ua + vn = 1 ⇔ ua = 1 mod n ⇔ a^-1 = u mod n

le produit de a et de a^-1, lorsqu'il est divisé par n donne un reste de 1 ⇔ a.a−1 ≡1 mod n

Corollary : Z/pZ is a field iff p is prime

CRT ⇔ Chinese Remainder Theorem :

Theorem : If p, q are coprime, $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \equiv \mathbb{Z}/(pq)\mathbb{Z}$.

Theorem : If m1, . . . ,mk are pairwise relatively prime, $\mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} \equiv \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z}$.

Utile pour RSA ou on va chercher p et q tel que pq = n

Group (Groupe) :

Caractéristiques plus bas. A group $(G, *, 1)$:

Ring (Anneau) :

Caractéristiques plus bas. A ring $(R, +, \times, 0, 1)$

Field (Corps) :

On a l'inverse pour la multiplication ← très important.

A field (F,+,×, 0, 1)

Résumé des différences

Propriété	Groupe	Anneau	Corps
Opérations	Une (addition ou multiplication)	Deux (addition et multiplication)	Deux (addition et multiplication)
Fermeture	Oui	Oui	Oui
Élément neutre	Oui pour l'addition	Oui pour l'addition, optionnel pour la multiplication	Oui pour les deux opérations
Inverse	Oui pour l'addition	Oui pour l'addition	Oui pour l'addition et la multiplication (pour les éléments non nuls)
Associativité	Oui	Oui	Oui
Commutativité	Optionnel	Optionnel	Oui (addition et multiplication)
Distributivité	N/A	Oui	Oui
Exemple	$(\mathbb{Z}, +)$	$\mathbb{Z}$	$\mathbb{Q}, \mathbb{R}, \mathbb{C}$

Lagrange :

Theorem : For a finite group (G, 1) and a ∈ G, a^#G = 1.

> **Théorème (Lagrange) : Pour un groupe fini G et un élément $a \in G$, il existe un entier k (appelé ordre de a) tel que $a^k = 1$, où 1 est l'élément neutre du groupe.**

Une formulation équivalente est :

> **$a^{\#G} = 1$, où $\#G$ est l'ordre (la taille) du groupe G, c'est-à-dire le nombre d'éléments dans G.**

Corollary :

Explication :

L'**ordre d'un élément** $a \in G$ (noté $o(a)$) est le plus petit entier positif k tel que $a^k = 1$. Le corollaire affirme que cet entier k divise la taille totale du groupe G, c'est-à-dire $\#G$. Cela découle directement du théorème de Lagrange, car $a^{\#G} = 1$, donc $o(a)$, qui est le plus petit k pour lequel $a^k = 1$, doit diviser $\#G$.

Exemple :

Dans $\mathbb{Z}_4$, l'élément 2 a pour ordre 2 car $2 + 2 = 4 \equiv 0 \pmod 4$, et cet ordre 2 divise $\#\mathbb{Z}_4 = 4$.

Theorem v2 : If H est is a sub-group of G, then #H|#G

Explication :

- Un **sous-groupe** H d'un groupe G est un sous-ensemble de G qui est lui-même un groupe avec la même opération.
- Le théorème dit que l'ordre de H divise l'ordre de G, autrement dit, $\#H \mid \#G$. Cela découle du fait que G peut être partitionné en **classes à gauche** ou **cosets** du sous-groupe H, et que chaque classe contient exactement $\#H$ éléments.

Exemple :

Prenons $G = \mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$, avec l'addition modulo 6.

- Le sous-groupe $H = \{0, 3\}$ est un sous-groupe de G car $3 + 3 = 6 \equiv 0 \pmod 6$.
- L'ordre de H est 2 et l'ordre de G est 6, et $2 \mid 6$.

Euler :

Theorem : Let a, n ∈ Z. If GCD(a, n) = 1, then aϕ(n) = 1 mod n.

ϕ(n) = nombre d'entiers positifs inférieurs ou égaux à n qui sont premiers avec n

exemple :

ϕ(12)=4, car les entiers 1, 5, 7 et 11 sont premiers avec 12

Fermat : If p is prime, then ap = a mod p ∀a ∈ Z/pZ.

Exemple : for p = 7 ; a = 5

5^7 = 78125

78125 = 5 mod 7

Euler totient function : ϕ(n) = #(Z/nZ)∗

Le **sous-groupe multiplicatif** (Z/nZ)* est défini comme l'ensemble des entiers entre 0 et n−1 qui sont **premiers avec** n (leur plus grand commun diviseur, PGCD(x,n) est égal à 1).

Property :

▶ ϕ (p) = (p − 1) for p prime

▶ ϕ (pk) = (p − 1)p^(k−1) for p prime

▶ ϕ (mn) = ϕ (m) ϕ (n) for GCD(m, n) = 1

Galois field :

Les **Galois Fields** (ou **corps finis**) sont des structures mathématiques fondamentales en cryptographie. Ce sont des ensembles finis de nombres qui possèdent des propriétés algébriques spécifiques, et ils sont souvent notés $GF(p^n)$, où :

- p est un nombre premier appelé la **caractéristique** du corps.
- n est un entier positif, représentant le degré de l'extension du corps.

Les Galois Fields permettent de définir des opérations comme l'addition, la soustraction, la multiplication et l'inversion (division) avec certaines règles, tout en restant dans cet ensemble fini.

Propriétés fondamentales des Galois Fields

1. **Nombre fini d'éléments** :
   - Un Galois Field contient exactement p^n éléments. Par exemple, $GF(2)$ contient les éléments $\{0, 1\}$, et $GF(2^3)$ contient $2^3 = 8$ éléments.
2. **Clôture** :
   - Toute opération (addition, multiplication, etc.) entre deux éléments du corps donne un résultat qui appartient aussi au corps.
3. **Existence d'inverses** :
   - Dans un Galois Field, chaque élément $x \neq 0$ a un **inverse multiplicatif**, c'est-à-dire un élément y tel que $x \cdot y = 1$. Cela garantit que la division est définie (sauf par 0).
4. **Associativité, commutativité et distributivité** :
   - Les opérations d'addition et de multiplication dans un Galois Field respectent les lois usuelles de l'algèbre (associativité, commutativité, distributivité).
5. **Structure cyclique** :
   - Le groupe multiplicatif des éléments non nuls de $GF(p^n)$ est cyclique, ce qui signifie qu'il existe un élément générateur (ou primitif) tel que tous les autres éléments du corps peuvent être obtenus comme des puissances de cet élément.

Exemples pratiques

1. $GF(2)$:
   - Ce corps ne contient que $\{0, 1\}$, avec des règles d'addition et de multiplication modulaires simples :
     - $0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 0$ (addition modulo 2).
     - $0 \cdot x = 0$, $1 \cdot x = x$.
2. $GF(2^3)$:
   - Ce corps a 8 éléments et est construit à l'aide d'un **polynôme irréductible** comme $x^3 + x + 1$. Les éléments sont souvent représentés comme des polynômes de degré inférieur à 3 avec des coefficients dans $GF(2)$:
     - Par exemple, $\{0, 1, x, x + 1, x^2, x^2 + 1, x^2 + x, x^2 + x + 1\}$.
   - Les opérations d'addition et de multiplication suivent les règles de l'arithmétique modulaire.

Primitive elements / polynomials :

Bit packing :

Une représentation en bits

➔ utile dans le TP par exemple
➔ ex : x^3 + x + 1 = 00001011 (sur un byte / octet)

LECTURE 3 :

Goal of the attacker : BREAK ≫ OW ≫ IND ≫ NM

Si on peut se défendre contre quelqu'un qui a un but fort (tout détruire), alors on ne peut sans doute pas se défendre de celui qui a un but minime (récupérer un bit d'information)

BREAK = Decrypt any message (by finding the decryption key) :: Strong goal.

OW (oneway) = decrypt a given message

IND (indistinguability) = distinguish between two ciphertexts :: just need 1 bit of information.

NM (non malleability) = modify a ciphertext so that it becomes the ciphertext of another plaintext clair :: little power.

Means that an attacker can use : COA ≪ KPA ≪ CPA ≪ CCA

Peu de moyen (COA = quelques ciphertexts) alors que CCA représente bcp de moyen, l'attaquant peut décrypter des ciphertexts

COA (Ciphertext Only Attack) = knows only one/some ciphertexts

KPA (Known Plaintext Attack) = knows pairs (plaintext, ciphertext)

CPA (Chosen Plaintext Attack) = able to encrypt chosen plaintexts

CCA (Chosen Ciphertext Attack) = able to decrypt some ciphertexts (except the target / Lot of power)

- ➔ si quelqu'un avec peu de moyen peut faire bcp de dégats : très mauvais = BREAK-COA
- ➔ si quelqu'un avec bcp de moyen fait peu de dégats : niveau max de sécurité : NM-CCA

asymetric crypto

- La clé publique de bob est publique
- Alice peut donc l'utiliser pour encrypter un text : ciphertext
- Bob peut décrypter ce texte avec sa clé privée (qui est reliée à la clé publique)

➔ Tout le monde peut écrire à Bob

➔ Mais il n'y a que Bob qui peut lire

One way functions : facile à encrypter mais difficile à décrypter si on a pas "l'indice" (trapdoor) qu'il faut.

La clé privée est entièrement déterminée par la clé publique donc on ne peut pas utiliser d'aléatoire → on cherche la difficulté de calcul. (want a trapdoor).

The exponentation :

Calculer le membre d'un groupe à la puissance k :

calculer y = x^k, où x∈G et k est un entier.

- ➔ Cost : O(log2k) (log en base 2)
- ➔ O(log2 |G|) (log en base 2)

The discrete logarithm :

La réciproque de l'exponentiation : on cherche k (l'exposant).

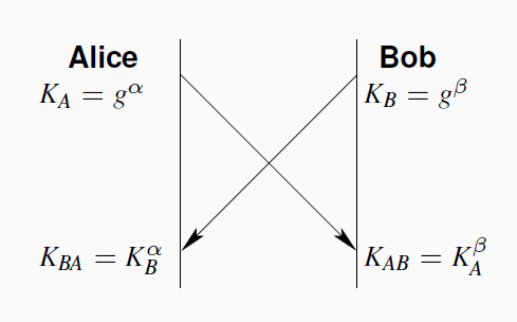
- ➔ From y, x ∈ G, compute k ∈ {1 . . . |G|} such that y = x^k

One way fonction :

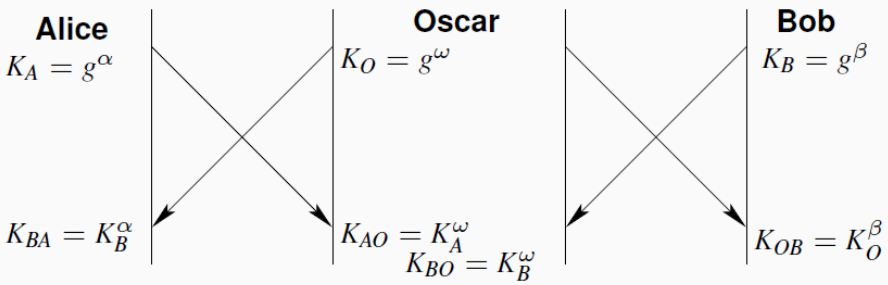
- Expx (exponentation de x) : facile (en temps polynomial), facile à encrypter
- Logx : hard (on ne connait pas d'algo en temps polynomial).

Elliptic curve arithmetic : ECC

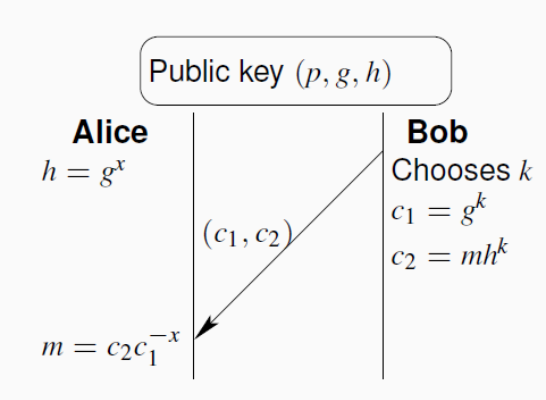
Diffie hellman key exchange protocol :



- Weakness : *Man in the middle attack*



El Gamal :



How to decipher : $c_2 c_1^{-x} = m h^k g^{-kx} = m g^{kx} g^{-kx} = m$

Integer factorization :

Retrouver 2 nombres tel que n = p * q

Mais bien que ce soit très simple pour des petits nombres c'est très dur pour de très grand nombres (d'autant plus quand c'est une factorisation entre 2 nombres premiers).

➔ One way fonction à la base de la sécurité de RSA (peut rappeler le CRT)

Theorem Rivest Shamir Adelman 78 :

Le théorème dit que pour tout entier a dans Z/nZ et pour tout entier k, il existe une relation :

$$a^{1+k(p-1)(q-1)} \equiv a \mod n$$

Propriété de modularité : Si vous élevez un élément du groupe multiplicatif à un puissance équivalente à son ordre (ou un multiple de cet ordre), le résultat est congruent à l'élément.

RSA :

- Let $e < \varphi(n)$ coprime with $\varphi(n)$: $\Rightarrow \exists d$ s.t. $ed = 1 \mod \varphi(n)$

$$ed = 1 + k\varphi(n)$$

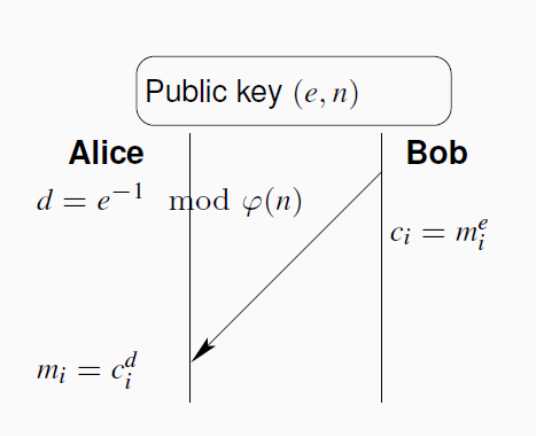
The RSA cryptosystem:

- Public key: (e, n), encryption: $c = m^e \mod n$
- Private key: (d, n), decryption: $c^d \mod n$

RSA Theorem: $c^d = m^{ed} = m^{1+k\varphi(n)} = m \mod n$

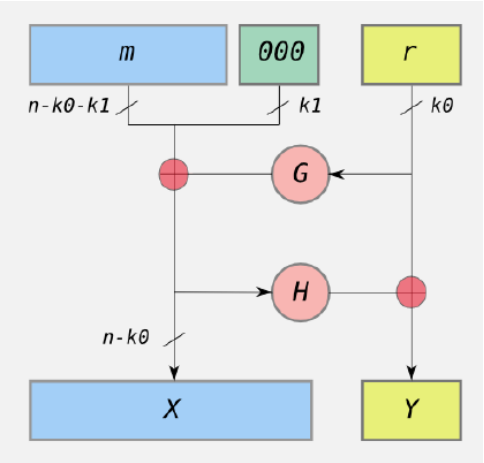
Warning:

- $\varphi(n)$ must remain private (since inverting $e \mod \varphi(n)$ is easy)
- p, q must remain private



Randomized padding :

RSA is not IND-CPA = not semantically secure. → use random padding on each block :



RSA algorithm complexity :

- Computing p,q primes → prime number generator → O~($\log^2 n$)
- Computing n, ϕ (n) = (p-1)(q-1) → integer multiplication → O~(logn)
- Computing d = e^-1 mod ϕ (n) → extended euclidean algo → O~(logn)
- Encryption : c = m^e mod n → modular exponentiation → O~($\log^2 n$)
- Decryption : m = c^d mod n → modular exponentiation → O~($\log^2 n$)

CRT also use for RSA

Attack by fault injection on CRT-RSA :

Using a laser : on can flip a bit in a aregister of the chip.

Moyen de faire tomber le chiffrement (comme n = p * q, le chiffrement est divisé en 2 → facilite les choses pour l'attaquant).

Baby step / Giant step (solve DLP) :

Given $x, y \in G$, *find* $k = k_0 + mk_1$ *such that* $y = g^k = g^{k_0}(g^m)^{k_1}$

Avoir 2 loops :

- Une pour les baby steps, de taille m où l'on va tout stocker)
- Une autre pour les giant steps où l'on va comparer avec les valeurs de babystep jusqu'à trouver un match.

Possible de stocker dans un simple tableau ou une hash table.

Cost : $O(\sqrt{|G|})$ time and $O(\sqrt{|G|})$ space

Rho Pollard's algorithm (integer factorization / DLP) :

Index Calculus Algo (DLP) :

Quadratic sieve :

Complexity (of algo) :

The Ln[α , c] notation :

- Ln[0, c] = polynomial in the size log n
- Ln[1, c] = exponential in the size log n

Au plus The α est proche de 1 → au plus l'algo est exponentiel (de 0 au plus il est polynomial).

- Rho pollard : Lp[1, 1/2] ← exponential time
- Index Calculus : Lp[1/2, c] ← sub-exponential time
- Quadratic sieve : Lp[1/2, c] ← sub-exponential time
- Number field sieve : Lp[1/3, c] ← sub-exponential time

	Factorisation	Discrete logarithm
Rho Pollard	$L_p[1, 1/2]$	$L_p[1, 1/2]$
Index Calculus	N/A	$L_p[1/2, c]$
Quadratic Sieve	$L_p[1/2, c]$	N/A
Number Field Sieve	$L_p[1/3, c]$	$L_p[1/3, c]$

Seurity parameters :

Security level 66 : considered has broken with an amount of effort : brut forcing 2^66 problems (= RSA 768).

Security level 106 : considered secure (RSA 2048).

LECTURE 4 :

Provable security :

How to prove the One-way-ness ? ← analyse the time complexity / provide a lower bound.

= Raisonner sur un algo que l'on ne connait pas mais on estime qu'il doit être dur à résoudre ou non.

Contradiction proof , by reduction ⚠

⚠ Que sur les problèmes décisionels : résultat = oui ou non.

Class P :

A problem is in the complexity class **P** if there is an algorithm A(x) that solves it on every instance x in time polynomial in |x| (the size of x).

Les problèmes solvables efficacement.

Class NP ⇔ Non Deterministic Polynomial time :

A decision problem F is in the complexity class NP if there is a polynomial time algorithm V(x, y) (== verifier) such that :

- F(x) = 1 ⇒ ∃z of size poly in |x| s.t. V(x, z) = 1 z = certificate = extradata
- F(x) = 0 ⇒ ∀yV(x, y) = 0

L'oracle donne un certificat qui aide à la vérification du problème.

Les problèmes où la réponse OUI peut être vérifié (avec le verifier) en un temps polynomial.

NP = Set of problems efficiently verifiable.

Def : Karp reduction :

Notion de réduction entre problèmes de décision. A est réductible à un problème de décision B ssi il existe un algo en temps polynomial qui transforme un input x pour A en un input y pour B de sorte que A(x) = B(x).

Def : Turing / cook reduction :

Permet de faire plusieurs appels à un oracle pour le problème B.

On dit qu'un problème A est réductible à un problème B ssi existe un algorithme capable de résoudre A(x) en un nombre polynomial d'opérations (par rapport à la taille de l'entrée |x|) et en faisant appel un nombre polynomial de fois à un oracle pour le problème B.

Ainsi, un algorithme pour résoudre A peut effectuer des calculs supplémentaires, mais a également la possibilité de poser des questions à un oracle qui résout B en temps constant.

And all properties :

$If\ A \leq B\ Then$

- $B \in \mathbf{P} \Rightarrow A \in \mathbf{P}.$
- $A \notin \mathbf{P} \Rightarrow B \notin \mathbf{P}.$

1er = useless : Si B appartient à P mais que A est inférieur ou égal : A dans P

2nd = Si A n'est pas dans P mais qu'il est inférieur ou égal à B, alors B n'est pas dans P.

$\leq_P^{(Turing)}$ *and* $\leq_P^{(Karp)}$ *are transitive*

Permet d'utiliser des chaines de réductions pour prouver que des problèmes sont au plus aussi difficile que d'autres.

NP-hard :

Un problème B est dit NP-hard si tout problème dans la classe NP est réductible à B par une réduction en temps polynomial. Cela signifie que B est au moins aussi difficile que n'importe quel problème dans NP. (notation avec le Karp reduction).

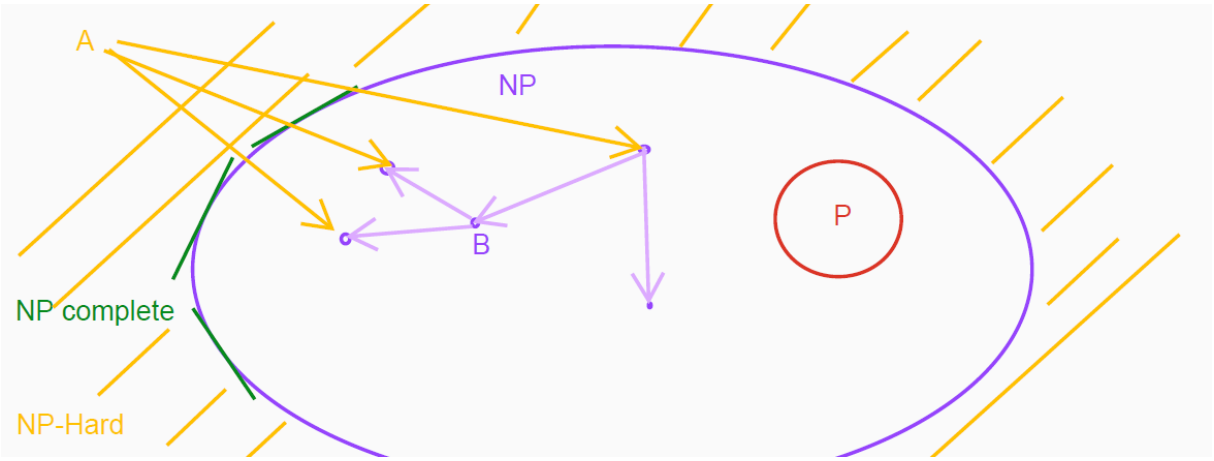
NP-complete :

NP-complete = NP-hard Union NP

À la fois dans NP hard et NP.

NP-intermediate :

Problems that are neither in P or NP-complete



One way trapdoor functions :

Easy to compute : P

Hard to invert : NP : on veut que l'inverse soit computationnaly hard.

One way fonction exist if P n'est pas égal à NP.

LECTURE 5 :

Shor's algorithm :

Solve “Order finding problem” with quantum computer in polynomial time.

Factorization with quantum computer :

Integer factorization can be solved by a quantum computer in time O(log2 N log log N).

DLP with quantum computer :

The Discrete logarithm problem can be solved by a quantum computer in time $O(\log^2 N \log \log N)$.

Error model :

problème : message reçu n'est pas toujours le message envoyé→ but : même avec un problème ce soit bien reçu (en comblant les erreurs).

Coding theory :

- ➔ Detect
- ➔ Correct

Tool : add redundancy / ou autres : utiliser des mots complets pour les lettres :

- beaucoup plus de places nécessaire que pour le même message
- marche pas si mal

A code is :

- t-detector, if any set error on t symbols can be detected
  Si il peut détecter tout erreur affectant jusqu’à t symboles dans un message codé.
- t-corrector, if any set error on t symbols can be corrected
  Si il peut détecter mais aussi corriger toute erreur affectant jusqu’à t symboles.

1st category of code : Stream codes :

Online processing of the stream of information

2nd category of code : Block codes :

Cutting information in blocks and applying the same treatment to each block

Linear codes :

Un code linéaire est un type de code correcteur d'erreurs où les mots de code sont des vecteurs d'un espace vectoriel et où les opérations de codage sont linéaires.

- Taille : n ← length of the vector carrying the information
- Dimension : k = Dim(C) ← Dimension of the code
- Rate : k / n ← Measure of efficiency

Un code linéaire de distance minimale d peut **corriger jusqu'à** $\left\lfloor \frac{d-1}{2} \right\rfloor$ erreurs et **détecter jusqu'à** $d-1$ erreurs.

Distance of a code :

Hamming weight : $w_H(x)$ Nombre de positions dans laquelle le mot contient 1 (nb erreurs).

Hamming distance : $d_H(x,y) = w_H(x-y)$ Nb de positions dans laquelle les 2 nombres diffèrent.

Minimum distance of a code δ Min des Hamming distances.

Perfect code :

Un code est appelé parfait s'il remplit complètement l'espace des mots de code possibles avec des sphères de Hamming de rayon t, où t est le nombre d'erreurs que le code peut corriger.

Cela signifie que chaque mot possible dans l'espace des mots de longueur n se trouve à une distance t d'un unique mot de code.

Chaque mot reçu est soit un mot de code valide, soit il se trouve dans une "sphère" autour d'un mot de code valide. Si un mot reçu n'est pas le mot de code exact, il se trouve dans la région d'influence (sphère) d'un mot de code valide, ce qui permet la correction.

Il n'y a pas d'ambiguïté dans la correction d'erreurs jusqu'à t erreurs.

➔ 3-reptition is perfect

Generator matrix :

matric G est utilisée pour coder les messages.

Chaque message m est transformé en un mot de code c en multipliant par la matrice génératrice G : c = mG

Parity check matrix :

matrice H : utilisée pour décoder les messages et détecter les erreurs.

Lorsqu'un mot de code c est reçu, on vérifie qu'il appartiennent bien au code Hc.

Si Hc = 0 :: le mot est correct sinon des erreurs sont détectées.

Hamming code :

Theorem Interpolation :

Lagrange

Reed Solomon codes :

Minimal distance of Reed-Solomon codes :

$\delta = n - k + 1$ ← minimum distance separable code.

Correct up to (n-k) / 2 errors.

Solving the key equation :

Rational Fraction Reconstruction (RFR) :

Generalized Reed Solomon codes :

$$\mathcal{C}_{GRS}(n,k,\mathbf{x},\mathbf{v}) = \{(v_1 f(x_1),\ldots,v_n f(x_n)), f \in K_{<k}[X]\}$$

- Same dimension and minimal distance ⇒MDS
- Existence of a dual GRS code in the same evaluation points: There is a vector $\mathbf{w}$ such that

$$\mathcal{C}_{GRS}(n,k,\mathbf{x},\mathbf{v})^{\perp} = \mathcal{C}_{GRS}(n,n-k,\mathbf{x},\mathbf{w})$$

i.e.

$$G_{\text{GRS}}(\mathbf{x},\mathbf{w})G_{\text{GRS}}(\mathbf{x},\mathbf{v})^{T} = 0$$

Alternant codes :

Motivation: workaround the limitatoin of GRS codes: $n \leq q$
⇒allow for arbitrary length n given a fixed field $\mathbb{F}_q$.

Idea: use a GRS over an extension $\mathbb{F}_{q^m}$, and restrict to $\mathbb{F}_q$.

Let

- $K = \mathbb{F}_q, \bar{K} = \mathbb{F}_{q^m}$ and $\mathbf{x} \in \bar{K}^n, \mathbf{v} \in (\bar{K}^*)^n$
- $\mathcal{C}_{\bar{K}} = \mathcal{C}_{GRS}(n,k,\mathbf{x},\mathbf{v})$ over $\bar{K}$ with minimum distance $D = n-k+1$

Then

$$\mathcal{C}_{Alt} = \mathcal{C}_{\bar{K}} \cap \mathbb{F}_q^n$$

- Dimension: $\geq n-(D-1)m = n-(n-k)m$
- Minimum distance: $\geq D$ by design

Goppa codes :

- An instance of a broad class of Algebraic Geometric Codes (AG-codes).
- Can be viewed as an alternant code for some special multiplier vector $\mathbf{v}$.

Let

- $K = \mathbb{F}_q, \bar{K} = \mathbb{F}_{q^m}$ and $\mathbf{x} \in \bar{K}^n$
- $f \in \mathbb{F}_{q^m}[X]$, $\deg f = r$ and $mr < n$
- $\mathbf{v} = (\frac{f(x_i)}{\prod_{j \neq i}(x_j - x_i)})$
- $\mathcal{C}_{\bar{K}} = \mathcal{C}_{GRS}(n, n-r, \mathbf{x}, \mathbf{v})$ over $\bar{K}$ with parameters $(n, n-r, r+1)$

Then

$\mathcal{C}_{Goppa} = \mathcal{C}_{\bar{K}} \cap \mathbb{F}_q^n$

- Dimension: $\geq n - rm$
- Minimum distance: $\geq r + 1$
- Case $q = 2^e$, with f square free
  $\Rightarrow$Minimum distance: $= 2r + 1$

McEliece / advantages :

A code based cryptosystem : designing a one-way function with a trapdoor.

$$\text{message} \times \boxed{G} + \text{rand. error} = \text{codeword}$$

Encryption: is easy (matrix-vector product)

Decryption: decoding a received word

- easy for known codes
- NP-complete for random linear codes

Trapdoor: efficient decoding when the code familiy is known

KeyGen

- Select an (n, k) binary linear code $\mathcal{C} \in \mathcal{F}$ correcting t errors, having an efficient decoding algorithm $\mathcal{A}_{\mathcal{C}}$,
- Form $G \in \mathbb{F}_q^{k \times n}$, a generator matrix for $\mathcal{C}$
- Sample uniformly a $k \times k$ non-singular matrix S
- Select uniformly an n-dimensional permutation P.
- $\hat{G} = SGP$

Public key: $(\hat{G}, t)$

Private key: (S, G, P)

Encrypt

$$E(\mathbf{m}) = \mathbf{m}\hat{G} + \mathbf{e} = \mathbf{m}SGP + \mathbf{e} = \mathbf{y}$$

where $\mathbf{e}$ is an error vector of Hamming weight at most t.

Decrypt

1. $\mathbf{y}' = \mathbf{y}P^{-1}$ $= \mathbf{m}SG + \mathbf{e}P^{-1}$
2. $\mathbf{m}' = \mathcal{A}_{\mathcal{C}}(\mathbf{y}')$ $= \mathbf{m}S$
3. $\mathbf{m} = \mathbf{m}'S^{-1}$

Security

Based on two assumptions:

- decoding a random linear code is hard (NP complete reduction)
- the generator matrix of a Goppa code, perturbed by S and P looks random (indistinguishability)

Pros:

- faster encoding/decoding algorithms than RSA, ECC (for a given security parameter)
- Post quantum security: still robust against quantum computer attacks

Cons:

- harder to use for signature (non determinstic encoding)
- large key size