

Les bases de la Cryptologie

Guénaël Renault

POLSYS LIP6/UPMC/INRIA

Part I

Définitions

La cryptologie

Définitions

La **cryptologie** est la *science du secret*. Elle se divise en deux disciplines :

- La **cryptographie** qui est l'étude des algorithmes permettant la protection d'informations (numériques). Ces algorithmes sont appelés **cryptosystèmes** ↗ design;
- la **cryptanalyse** qui est l'étude du niveau de sécurité des cryptosystèmes fournis par les cryptographes ↗ attack.

La cryptanalyse

Peut essentiellement se diviser en deux grandes familles d'attaques qui viseront deux cibles différentes :

- **algorithmes** : ici l'attaquant essaie de trouver des algorithmes efficaces pour résoudre les problèmes mathématiques sur lesquels reposent les cryptosystèmes ↗ Cours CRYPTO
- **implémentations matérielles** : ici l'attaquant utilisera les fuites d'information lors de l'exécution d'un cryptosystème pour retrouver les secrets ↗ Cours SCA.

La cryptanalyse

Peut essentiellement se diviser en deux grandes familles d'attaques qui viseront deux cibles différentes :

- **algorithmes** : ici l'attaquant essaie de trouver des algorithmes efficaces pour résoudre les problèmes mathématiques sur lesquels reposent les cryptosystèmes ↗ Cours CRYPTO
- **implémentations matérielles** : ici l'attaquant utilisera les fuites d'information lors de l'exécution d'un cryptosystème pour retrouver les secrets ↗ Cours SCA.

Cours CRYPTO : G. R. , Luk Bettale (Oberthur Tech.)

La cryptanalyse

Peut essentiellement se diviser en deux grandes familles d'attaques qui viseront deux cibles différentes :

- **algorithmes** : ici l'attaquant essaie de trouver des algorithmes efficaces pour résoudre les problèmes mathématiques sur lesquels reposent les cryptosystèmes ↗ Cours CRYPTO
- **implémentations matérielles** : ici l'attaquant utilisera les fuites d'information lors de l'exécution d'un cryptosystème pour retrouver les secrets ↗ Cours SCA.

Cours SCA : G. R. et P. Bazargan Sabet (UPMC)

E. Prouff , V. Lomné, A. Thillard, N. Robert, R. Benadjila,
J. Lopes-Esteves et Mathieu Renard, N. Robert (ANSSI)

David Vigilant (Gemalto)

Sylvain Guilley (Télécom ParisTech)

Luk Bettale (Oberthur)

Jessy Clédière (CEA LETI)

La cryptographie

La **cryptographie** protège l'information de différentes manières :

- **confidentialité** : pour s'assurer que l'information ne soit seulement accessible qu'à ceux dont l'accès est autorisé ;
- **authenticité** : vérifier l'identité d'une personne ou d'un matériel informatique ;
- **intégrité** : pouvoir affirmer que les données ont ou n'ont pas été modifiées ;

☞ Ces moyens doivent reposer sur des secrets

- **clé secrète** : cryptographie symétrique ;
- **clé publique/secrète** : cryptographie asymétrique



Principe de Kerckhoffs (Journal des sciences militaires, 1883)

La sécurité d'un cryptosystème ne doit reposer que sur le *secret de la clef*. En particulier un *attaquant est supposé connaître le cryptosystème* (Les cryptosystèmes militaires, vus comme des dispositifs physiques, peuvent tomber aux mains de l'ennemi).

La cryptographie

La **cryptographie** protège l'information de différentes manières :

- **confidentialité** : pour s'assurer que l'information ne soit seulement accessible qu'à ceux dont l'accès est autorisé ;
- **authenticité** : vérifier l'identité d'une personne ou d'un matériel informatique ;
- **intégrité** : pouvoir affirmer que les données ont ou n'ont pas été modifiées ;

Terminologie

Chiffrer : l'action de rendre un message clair M (*plaintext*) en un message C appelé cryptogramme ou message chiffré, illisible.

Déchiffrer : Action inverse du chiffrement.

Cryptosystème : L'algorithme (ou le dispositif physique) permettant de chiffrer des données.

Attaquer, Casser : Mettre à mal la sécurité d'un cryptosystème (retrouver M à partir de C sans connaître la clé, retrouver la clé).

La cryptographie: outils mathématiques

- Théorie de l'information (Shannon)
- Probabilité et statistique
- Arithmétique modulaire
- Arithmétique des polynômes
- Corps finis (Galois)
- Courbes elliptiques
- Théorie algorithmiques des nombres
- etc.

Part II

Chiffrement symétrique

Chiffrement symétrique : Modélisation

Définition

Un chiffrement est dit **symétrique** (ou à clé privée) si pour chiffrer et déchiffrer, la même clé secrète est utilisée.

☞ On s'intéresse uniquement aux **chiffrements par bloc** (pas aux chiffrements par flot).

Modélisation du cryptosystème

- \mathcal{P} et \mathcal{C} les alphabets pour écrire les messages clairs et les messages chiffrés respectivement.
- \mathcal{K} l'ensemble des clés possibles.
- Pour tout $K \in \mathcal{K}$ on peut définir deux applications $e_K : \mathcal{P} \rightarrow \mathcal{C}$ et $d_K : \mathcal{C} \rightarrow \mathcal{P}$ telles que $d_K(e_K(x)) = x$ pour tout $x \in \mathcal{P}$.

☞ les alphabets peuvent être composés de symboles uniques ou de blocs de symboles. Par exemple on peut choisir les 26 caractères non accentués de l'alphabet usuel.

Chiffrement d'un texte : Mode ECB

Chiffrement symétrique (la même clé K pour dé/chiffrer)

- \mathcal{P} et \mathcal{C} les alphabets clairs et chiffrés.
- \mathcal{K} l'ensemble des clés possibles.
- $K \in \mathcal{K} : e_K : \mathcal{P} \rightarrow \mathcal{C}$ et $d_K : \mathcal{C} \rightarrow \mathcal{P}$ avec $\forall x \in \mathcal{P} d_K(e_K(x)) = x$.
- Les applications e_K et d_K nécessitent la connaissance complète de K pour être définies.

En pratique : chiffrer avec la clé K

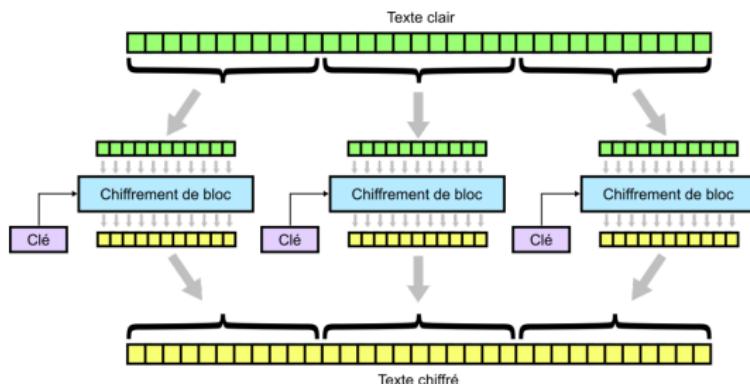
Pour chiffrer un texte P clair on procède comme suit

- ① On découpe P en blocs correspondant aux éléments de \mathcal{P}
- ② On applique e_K sur chacun des blocs de P
- ③ On obtient ainsi C un message correspondant à P et écrit à l'aide de \mathcal{C} .

Chiffrement d'un texte : Mode ECB

Chiffrement symétrique (la même clé K pour dé/chiffrer)

- \mathcal{P} et \mathcal{C} les alphabets clairs et chiffrés.
- \mathcal{K} l'ensemble des clés possibles.
- $K \in \mathcal{K} : e_K : \mathcal{P} \rightarrow \mathcal{C}$ et $d_K : \mathcal{C} \rightarrow \mathcal{P}$ avec $\forall x \in \mathcal{P} d_K(e_K(x)) = x$.
- Les applications e_K et d_K nécessitent la connaissance complète de K pour être définies.



Ce mode de chiffrement est à éviter.

Chiffrement d'un texte : Mode ECB

Chiffrement symétrique (la même clé K pour dé/chiffrer)

- \mathcal{P} et \mathcal{C} les alphabets clairs et chiffrés.
- \mathcal{K} l'ensemble des clés possibles.
- $K \in \mathcal{K} : e_K : \mathcal{P} \rightarrow \mathcal{C}$ et $d_K : \mathcal{C} \rightarrow \mathcal{P}$ avec $\forall x \in \mathcal{P} d_K(e_K(x)) = x$.
- Les applications e_K et d_K nécessitent la connaissance complète de K pour être définies.

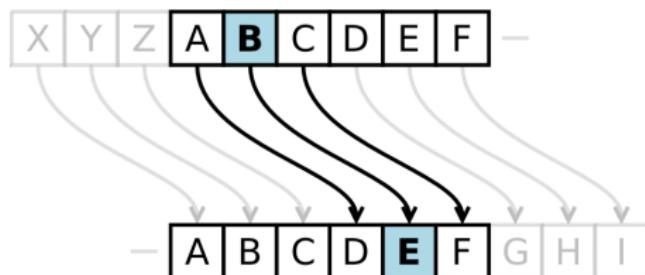
En pratique : déchiffrer avec la clé K

Pour déchiffrer le texte C on procède comme suit

- ① On découpe C en blocs correspondant aux éléments de \mathcal{C}
- ② On applique d_K sur chacun des blocs de C
- ③ On obtient ainsi P le clair de à C .

Période artisanale : Cesar

Le chiffrement de César (décrit précédemment dans le Kàma-sùtra -400), le point de départ de la cryptographie par substitution générale.

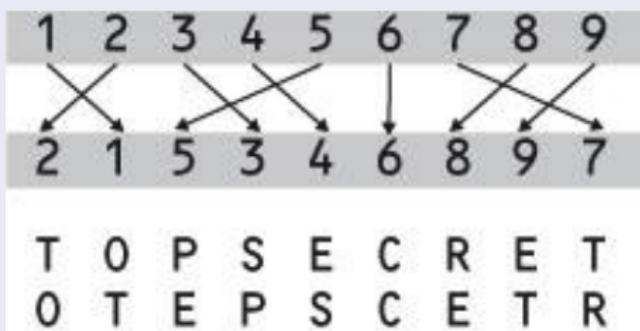


Implantation en shell d'un décalage de César :

```
tr a-zA-Z n-zA-MN-ZA-M
```

Période artisanale : Transposition/Substitution

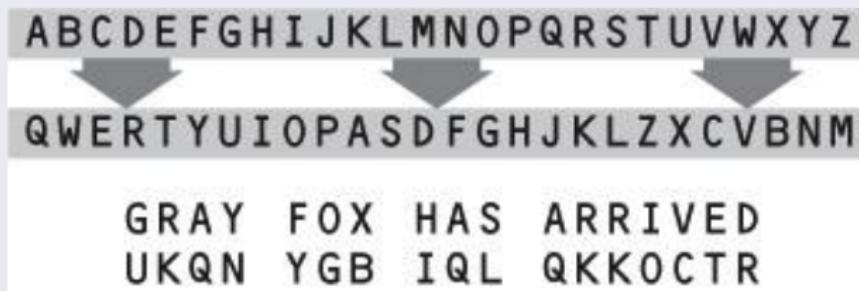
Chiffrement par transposition



Clef secrète est la permutation entre les positions des caractères.

Période artisanale : Transposition/Substitution

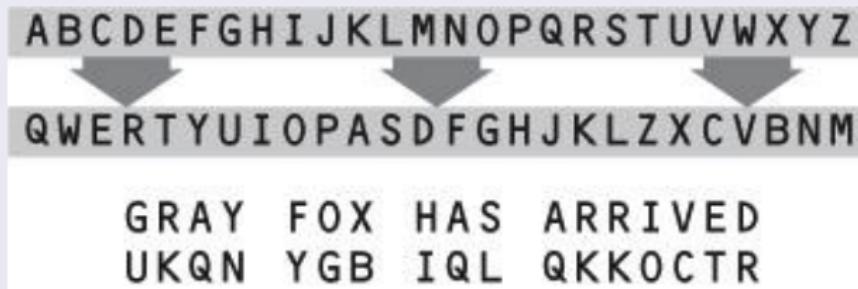
Chiffrement par substitution monoalphabétique



Clef secrète est la permutation entre les alphabets .

Période artisanale : Transposition/Substitution

Chiffrement par substitution monoalphabétique



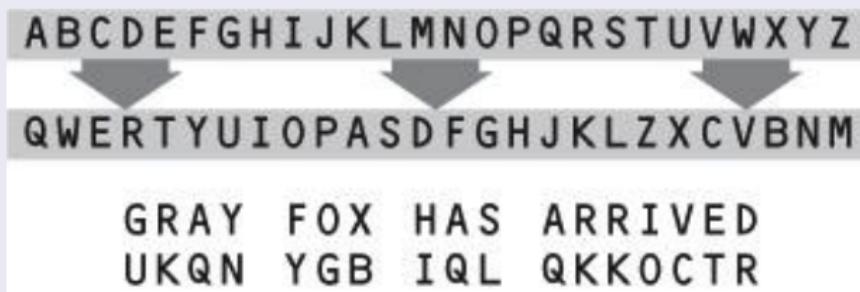
Clef secrète est la permutation entre les alphabets .

Cryptanalyse force brute :

- Substitution : $26!$ permutations possibles.
- Décalage : 26 décalages possibles.

Période artisanale : Transposition/Substitution

Chiffrement par substitution monoalphabétique



Clef secrète est la permutation entre les alphabets .

Cryptanalyse un peu plus intelligente : analyse des fréquences

Période artisanale : le point faible du monoalphabétique

- ☞ Al Kindi (~ 800) explique dans son ouvrage de cryptanalyse la méthode de l'étude des fréquences.
- ☞ Marie Stuart (1587) Reine d'Écosse *perd la tête* pour haute trahison. Elle utilisait un chiffrement monoalphabétique qui a cassé par un *man-in-the-middle*.



Période artisanale : Vigenère et le polyalphabétique

☞ Vigenère publie en 1587 le résultat de ses recherches sur des méthodes proposées par Alberti, Trithème et Porta. Il propose un cryptosystème polyalphabétique .



Vigenère ou multi-décalage

On associe aux lettres un entier dans $\{0, \dots, 25\}$ et on fixe un mot $C = (c_1, \dots, c_\ell)$ de ℓ lettres pour être la clé secrète . On découpe le message à envoyer en blocs $B_i = (x_{(i,1)}, x_{(i,2)}, \dots, x_{(i,\ell)})$ de ℓ lettres . Le chiffrement/déchiffrement par un multi-décalage sur les B_i :

$$E(x_{(i,j)}) = x_{(i,j)} + c_j \pmod{26}$$

$$D(x_{(i,j)}) = x_{(i,j)} - c_j \pmod{26}$$

ORDREDATTAQUER
+ CYPHERCYPHERCY mod 26

QPSYIUCRIHUMGP

Période artisanale : Vigenère et le polyalphabétique

☞ Vigenère publie en 1587 le résultat de ses recherches sur des méthodes proposées par Alberti, Trithème et Porta. Il propose un cryptosystème polyalphabétique .



Vigenère ou multi-décalage

On associe aux lettres un entier dans $\{0, \dots, 25\}$ et on fixe un mot $C = (c_1, \dots, c_\ell)$ de ℓ lettres pour être la clé secrète . On découpe le message à envoyer en blocs $B_i = (x_{(i,1)}, x_{(i,2)}, \dots, x_{(i,\ell)})$ de ℓ lettres . Le chiffrement/déchiffrement par un multi-décalage sur les B_i :

$$E(x_{(i,j)}) = x_{(i,j)} + c_j \pmod{26}$$

$$D(x_{(i,j)}) = x_{(i,j)} - c_j \pmod{26}$$

Cryptanalyse force brute : longueur ℓ fixée, on a 26^ℓ clés possibles !
Analyse des fréquences : Ne fonctionne plus directement !

Période artisanale : Vigenère et le polyalphabétique

☞ Vigenère publie en 1587 le résultat de ses recherches sur des méthodes proposées par Alberti, Trithème et Porta. Il propose un cryptosystème polyalphabétique .



Vigenère ou multi-décalage

On associe aux lettres un entier dans $\{0, \dots, 25\}$ et on fixe un mot $C = (c_1, \dots, c_\ell)$ de ℓ lettres pour être la clé secrète . On découpe le message à envoyer en blocs $B_i = (x_{(i,1)}, x_{(i,2)}, \dots, x_{(i,\ell)})$ de ℓ lettres . Le chiffrement/déchiffrement par un multi-décalage sur les B_i :

$$E(x_{(i,j)}) = x_{(i,j)} + c_j \pmod{26}$$

$$D(x_{(i,j)}) = x_{(i,j)} - c_j \pmod{26}$$

Cryptanalyse force brute : longueur ℓ fixée, on a 26^ℓ clés possibles !

Analyse des fréquences : Ne fonctionne plus directement !

Difficulté de son utilisation manuelle \Rightarrow impopulaire jusqu'au 19eme siècle . Mono-alphabétique persiste (homophonique)

Cryptanalyse (~ 1860): Vigenère et le polyalphabétique

- ☞ Charles Babbage (1792-1871)
- ☞ Friedrich Wilhelm Kasiski (1805-1881)



Longueur de la clé : Test de Kasiski

KQOWEFVJPUJUUNUKGLMEKJINMWUXFQMJKBGWRLFNFQHUD**WUU**MBSVLPS
NCMUEKQCTESWR**EKOYSS**IWCTUAXYOTAPXPLWPNTCGOJBGFQHTD**WXIZA**
YGFFNSXCSEYNCTSSPNTUJNYTGGWZGR**WUU**NEJUUQEAPYMEKQHUIDUXFP
GUYTSMTFFSH**NUOCZGM**RUWEYTRGKMEEDCTVRECDBDJQCUSWBPNLGOYL
SKMTEFVJJTWMMFMWPNMEMTMHRSPXFSSKFFST**NUOCZGMDOEOYEEKCPJR**
GPMURSKHFRSEIUEVGOYC**WXIZAY**GOSAANY**DOEOY**JLWUNHAMEBFELXYVL
WNOJNSIOFRWUCCESWKVI**GMU**CGOCRUWGNMAAFFVNSIUIDEKQHCEUCPFC
MPVSUDGAVEMNYMAMVLFMAOYFNTQCUAFVJNXKLNEIWCVODCCULWRIFT
W**GMU**SWOVMATNYBUHTCOCWFYTNMGYTQMKBBLGFBTWOJFTWGNTJEJKNEE
DCLDHWTBUVGFBIJG

- ☞ Le pgcd (probable) entre les différentes distances

Cryptanalyse automatique du chiffrement de Vigenère

- ① Retrouver la longueur de la clé
- ② Retrouver la clé

☞ Pour la seconde étape on peut voir cela comme une attaque d'un multi-chiffrement monoalphabétique.

Idée clé:

Plutôt que de réaliser une attaque brute force en t^ℓ on passe en $\ell \times t$ où t est la taille de l'alphabet et ℓ la longueur de la clé secrète.

Pour tester chaque caractère de la clé secrète on utilise un distinguateur statistique.

Cryptanalyse automatique : Utilisation des Statistiques

William F. Friedman (1891 - 1969)

- ☞ Utilisation des invariants statistiques pour distinguer un langage courant d'un flux aléatoire.
- ☞ Ne s'attache pas uniquement à la proba de chaque caractère pris indépendamment les uns des autres
- ☞ Considère la distribution dans sa généralité



Cryptanalyse automatique : Utilisation des Statistiques

Application Mono-Alphabétique

On s'intéresse à la corrélation entre la distribution d'un texte chiffré et celle d'un texte clair dans un langage donné.

Karl Pearson (1857 - 1936)



$$\rho(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

Cryptanalyse automatique : Utilisation des Statistiques

William F. Friedman (1891 - 1969)

Définition

L'indice de coïncidence d'un texte est la probabilité de tirer un couple de lettres identiques au hasard.

$$IC = \sum_{i=0}^{25} \frac{C_2^{n_i}}{C_2^n} = \sum_{i=0}^{25} \frac{n_i(n_i - 1)}{n(n - 1)}$$

où n_i est le nombre de caractère c_i dans le texte et n est la longueur total de ce dernier.



Cryptanalyse automatique : Utilisation des Statistiques

☞ $IC = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$ distingue l'aleatoire \Rightarrow attaque longueur de la clé

- Lorsque le texte est suffisamment long ($n \rightarrow \infty$) l'indice IC est donné par

$$IC \simeq \sum_{i=0}^{25} p_i^2 = \textcolor{blue}{0.074} \text{ pour l'alphabet français}$$

où p_i est la probabilité d'apparition de la lettre numérotée i dans un texte en français.

- Lorsque les lettres sont distribuées aléatoirement, l'indice de coïncidence est faible

$$IC \simeq \sum_{i=0}^{25} \left(\frac{1}{26}\right)^2 \simeq \textcolor{blue}{0.038}$$

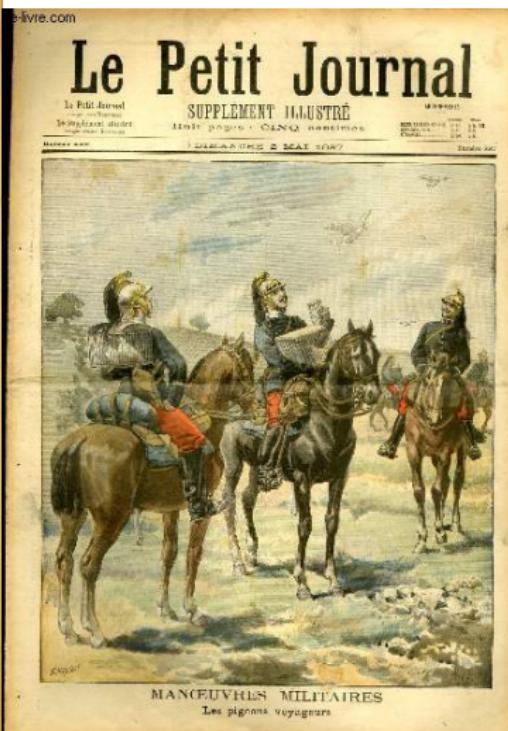
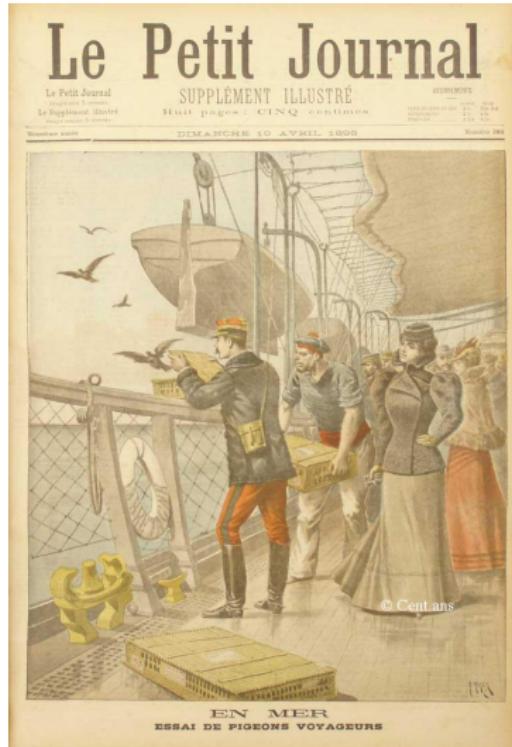
Cryptanalyse automatique : Utilisation des Statistiques

☞ $IC = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$ distingue l'aleatoire ⇒ attaque longueur de la clé

Key Length	Average Index	Individual Indices of Coincidence
4	0.038	0.034, 0.042, 0.039, 0.035
5	0.037	0.038, 0.039, 0.043, 0.027, 0.036
6	0.036	0.038, 0.038, 0.039, 0.038, 0.032, 0.033
7	0.062	0.062, 0.057, 0.065, 0.059, 0.060, 0.064, 0.064
8	0.038	0.037, 0.029, 0.038, 0.030, 0.034, 0.057, 0.040, 0.039
9	0.037	0.032, 0.036, 0.028, 0.030, 0.026, 0.032, 0.045, 0.047, 0.056

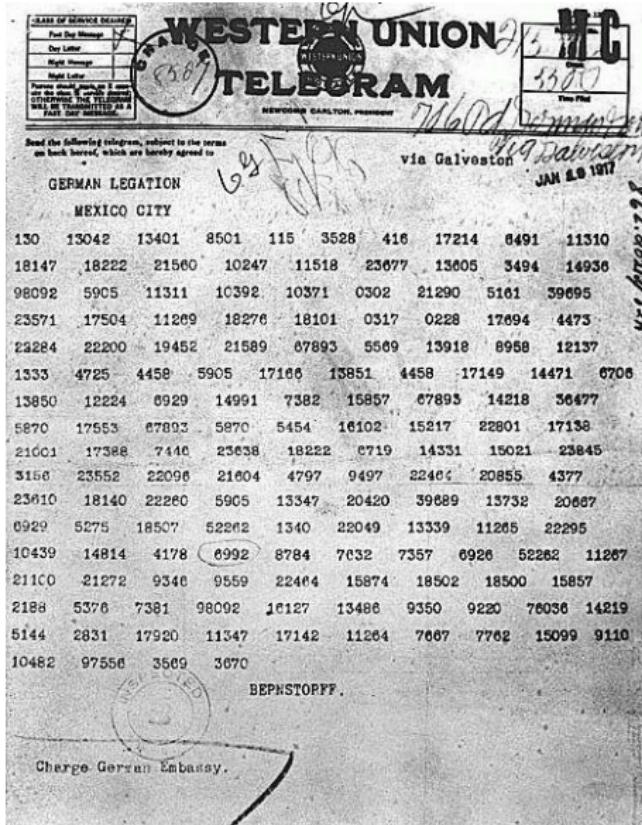
De l'artisanal à l'informatique via la mécanique

- ☞ Création de bureaux spéciaux de cryptanalyse
- ☞ Avant 1914, la crypto reste très proche de Vigenère.



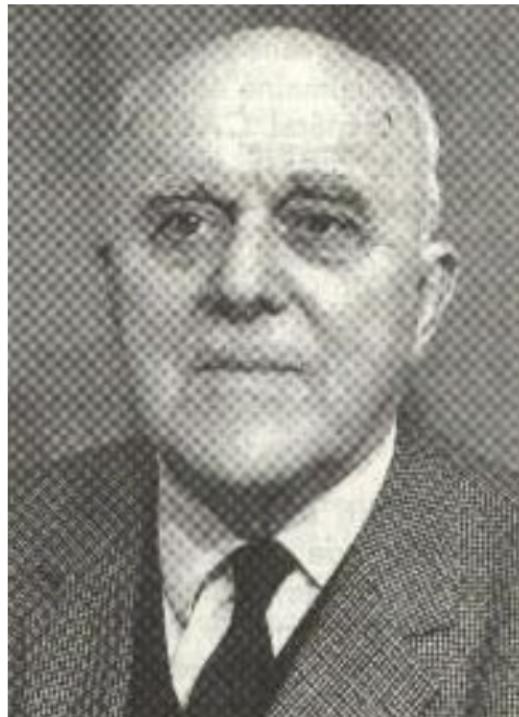
De l'artisanal à l'informatique via la mécanique

La cryptanalyse pousse les USA à entrer en guerre !



De l'artisanal à l'informatique via la mécanique

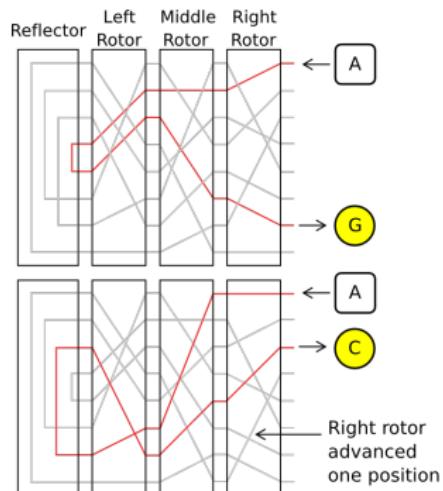
☞ La cryptanalyse d'un message allemand évite une défaite sur Paris !
Le chiffrement ADFGVX inventé par Nebel et cryptanalysé par Painvin.



Période intermédiaire *entre deux guerres* et WW2 : Enigma la mécanique cryptographique

- ☞ Invention de la télécommunication et croissance du nombre des échanges chiffrés

Enigma (Scherbius, 1919) \Leftrightarrow polyalphabétique mécanique



Période intermédiaire 1914-1945 : Enigma la mécanique cryptographique

- ☞ Le Biuro Szyfrów, bureau du chiffre polonais autour du M. Rejewski puis le centre de cryptanalyse anglais de Bletchley Park autour de A. Turing perce le secret d'Enigma.



- ☞ Identifier un certain type de permutations dans le cryptosystème, utilisation des premiers ordinateurs électro-mécaniques (Bombes).
- ☞ Reste classifiée jusqu'en 1974.

Shannon et la théorie de l'information

Claude Shannon (1916 - 2001) a publié deux articles de recherche en 1948 et 1949 donnant les fondations de la **théorie de l'information** et, plus généralement, de la cryptologie moderne. Il donne les premières preuves de sécurité d'un cryptosystème en se basant sur des principes de probabilité et de statistique.

Définitions importantes

- Chiffrement parfait
- Entropie d'un langage



Chiffrement parfait

Intuition

Un cryptosystème sera dit **chiffrement parfait** lorsque la donnée d'un message chiffré ne révèle aucune fuite d'information sur la clé ou le message clair correspondant et aucune information non plus sur les textes chiffrés futurs.

Caractérisation

Supposons qu'un cryptosystème vérifie

$$\#\mathcal{K} = \#\mathcal{P} = \#\mathcal{C}$$

alors (c'est aussi une condition nécessaire) il sera chiffrement parfait
ssi les deux conditions suivantes sont vérifiées:

- Toutes les clés sont utilisées avec même probabilité
- Pour tout couple $(m, c) \in \mathcal{P} \times \mathcal{C}$ il existe une unique clé k telle que $e_k(m) = c$.

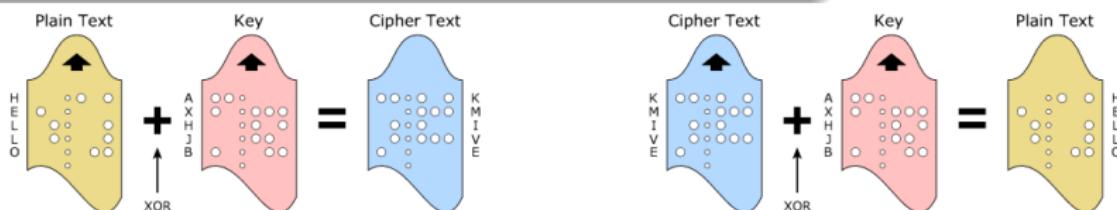
- ☞ Beaucoup de contraintes sur le cryptosystème
- ☞ Peu fréquent en pratique

Exemple : Vernam's One Time Pad

Gilbert S. Vernam (1890-1960), proposa le cryptosystème qui porte son nom en 1917 et fût déposé un brevet le concernant jusqu'en 1919 (US PATENT 1310719).

Le principe est simple : l'utilisation du XOR ! Les messages clairs et chiffrés, les clés seront des suites de bits de même longueur.

$$C[i] = M[i] \oplus K[i] \text{ et } M[i] = C[i] \oplus K[i]$$



Exemple : Vernam's One Time Pad

- C'est le seul cryptosystème à chiffrement parfait !
 - Très peu pratique !
 - Utilisé dans la cryptographie Top Secrète (téléphone rouge, valise diplomatique, militaire (Atomique)).
 - Le principe est utilisé pour faire des chiffrements symétrique dépendant de générateur aléatoire.

5 Bit One-time tape XOR Encryption Principle

XOR (Exclusive OR) = \oplus

$$1 \oplus 1 = 0$$

$$1 \oplus 0 = 1$$

$$0 \oplus 1 = 1$$

$$0 \oplus 0 = 0$$

TELEX Signal



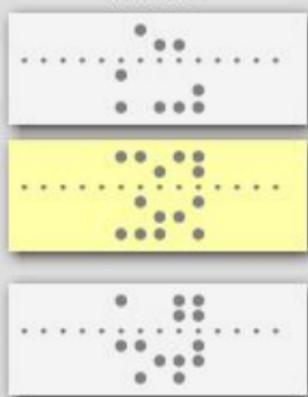
HELI 10

Plain

⊕

Key

Cipher



Exemple : Vernam's One Time Pad

- La clé doit être aussi longue que le message
- Elle doit être aléatoire
- Elle doit être utilisée une unique fois
- Projet VENONA des USA pour écouter les discussions Russes utilisant un Two-Time Pad ⇒ faiblesse !

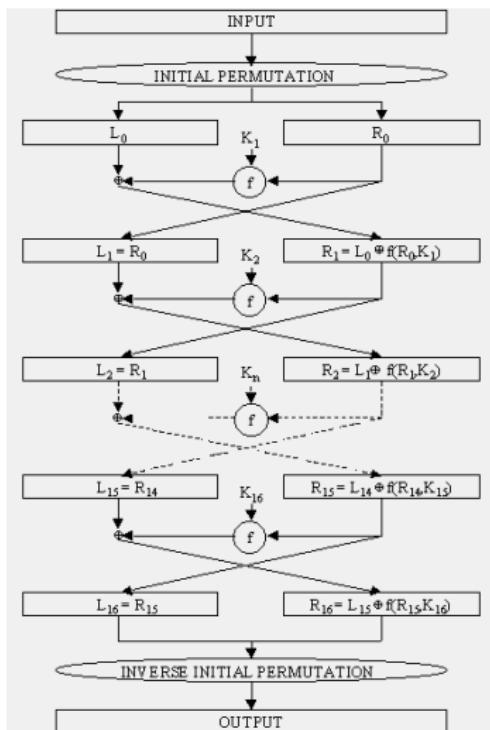


Début ~1970 : besoin de standardisation

Basé sur deux concepts introduits par Shannon, le DES est proposé en 1977 comme standard de chiffrement par bloc.

- Confusion: chaque bit du chiffré dépend de manière hautement non-linéaire en les bits du clair et de la clé.
- Diffusion: chaque bit du clair ou de la clé affecte un grand nombre de bits du chiffré.

☞ Shannon propose ces conditions comme nécessaires pour obtenir un chiffrement sûr.

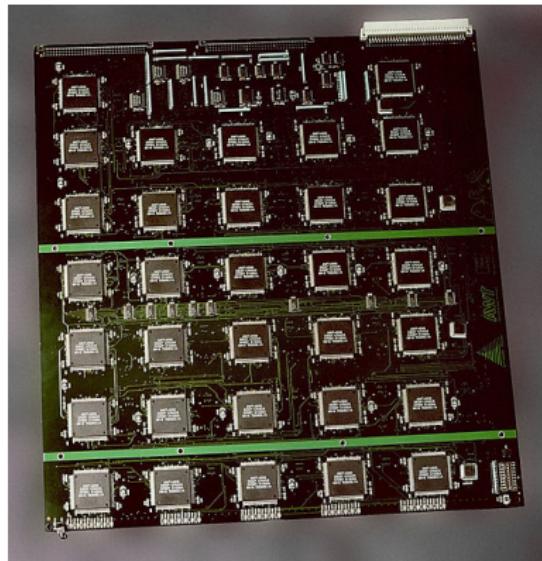


La fin du DES

Très tôt le DES est critiqué pour la taille de clé trop petite (64 bits mais seulement 56 utiles).

Ordinateurs construits pour casser le DES par recherche exhaustive :

- 1977 : 20 million \$, DES casser th. en 24h (Diffie Hellman)
- 1993 : 1 million \$, DES casser th. en 7h (Wiener)
- 1998 : 250 K\$, DES **casser** en 3 jours (Deep crack de EFF)



Attaques théoriques

- Cryptanalyse différentielle (Biham, Shamir, 1988) en 2^{48}
- Cryptanalyse linéaire (Matsui, Gilbert, 1993) en 2^{43}

☞ On passe au 3DES pour assurer une bonne sécurité !

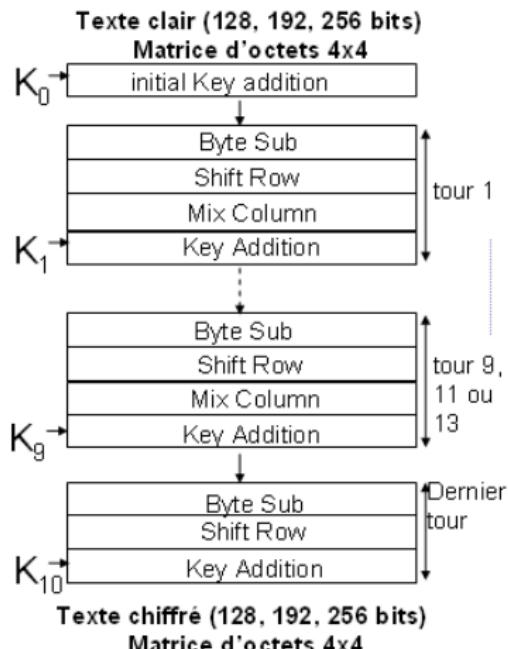
Le remplaçant du DES : AES

En 1997 une compétition est organisée pour trouver un remplaçant au DES

Rijndael (Daemen, Rijmen) gagne le concours en 2001

- S-Box basée sur arithmétique modulaire assure la confusion
- Une permutation en plusieurs étapes assure la diffusion
- Une expansion de clé simple assure l'efficacité

☞ En 2003 la NSA déclare AES comme moyen de chiffrement pour des données classifiées SECRET ou TOP SECRET.



☞ Représentation matricielle

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]

    state = in

    AddRoundKey(state, w[0, Nb-1])           // See Sec. 5.1.4

    for round = 1 step 1 to Nr-1
        SubBytes(state)                   // See Sec. 5.1.1
        ShiftRows(state)                 // See Sec. 5.1.2
        MixColumns(state)                // See Sec. 5.1.3
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for

    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

    out = state
end
```

Figure 5. Pseudo Code for the Cipher.¹

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp

    i = 0

    while (i < Nk)
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while

    i = Nk

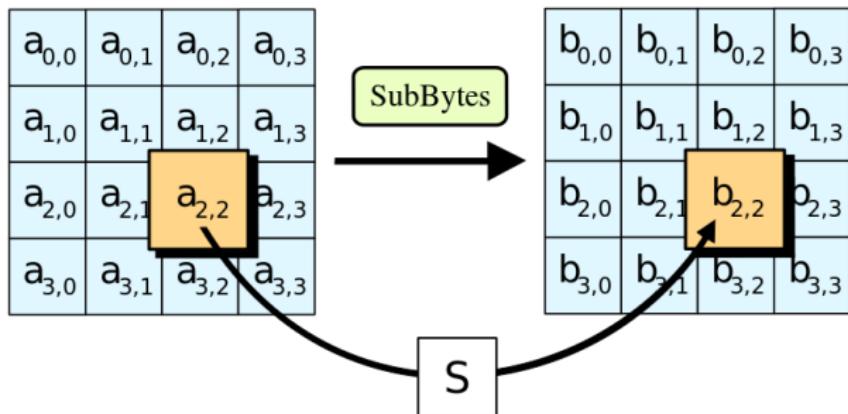
    while (i < Nb * (Nr+1))
        temp = w[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
end
```

Note that $Nk=4$, 6 , and 8 do not all have to be implemented; they are all included in the conditional statement above for conciseness. Specific implementation requirements for the Cipher Key are presented in Sec. 6.1.

Figure 11. Pseudo Code for Key Expansion.²

AES : Byte Sub

Représentation matricielle



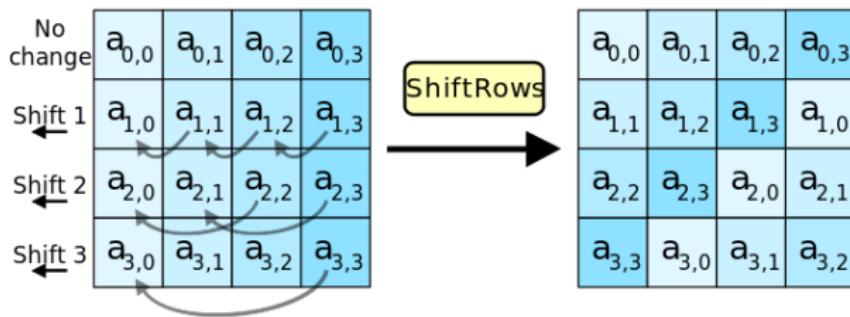
AES : Byte Sub (S-Box)

☞ Boite-S définit sur 8 bits (description arith. plus loin)

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

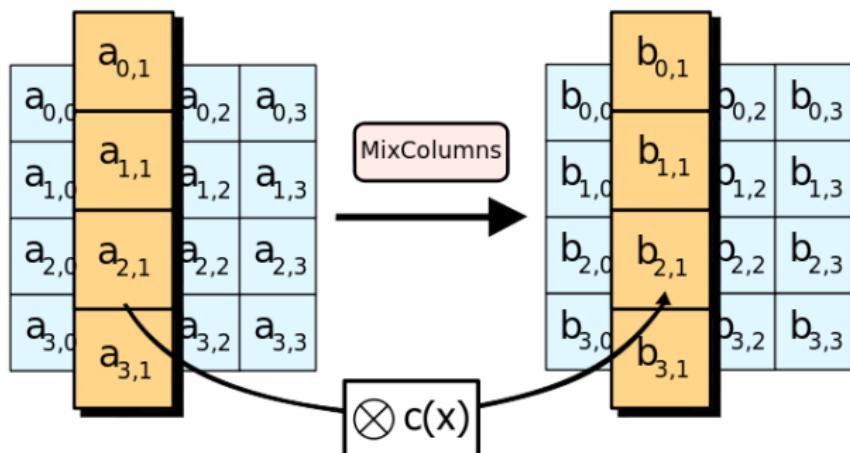
AES : Permutation (les lignes)

☞ Représentation matricielle



AES : Permutation (les colonnes)

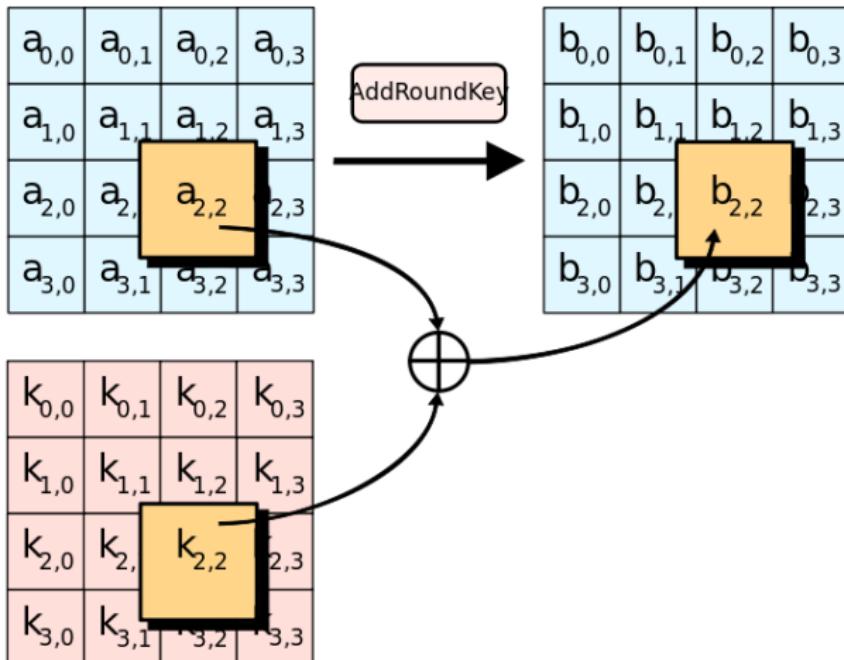
Représentation matricielle



$$c(x) = 3x^3 + x^2 + x + 2 \pmod{x^4 + 1} \text{ dans } \mathbb{F}_8[x]$$

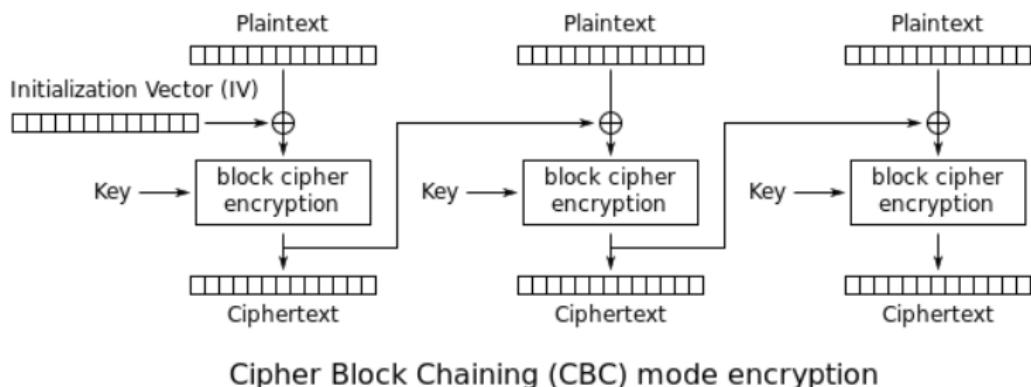
AES : Addition de la sous-clé

Représentation matricielle



AES : Un mode opératoire plus secure

☞ Rappelez vous, ne jamais utiliser ECB !

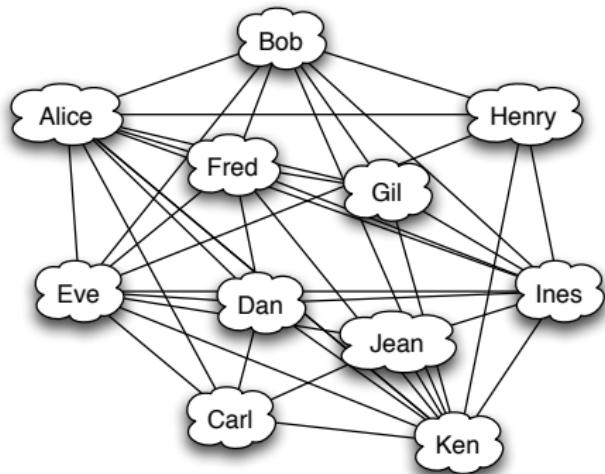


Part III

De l'échange de clés à la clé publique

Problématique

- ☞ Le nombre énorme de clés à gérer !



- ☞ Des échanges chiffrés avec n amis nécessitent n clés !

Échanger des clés

☞ 1976 [Diffie, Hellman et Merkle](#) publient le premier schéma d'échange de clés. Ils proposent (enfin) la notion de clé publique.

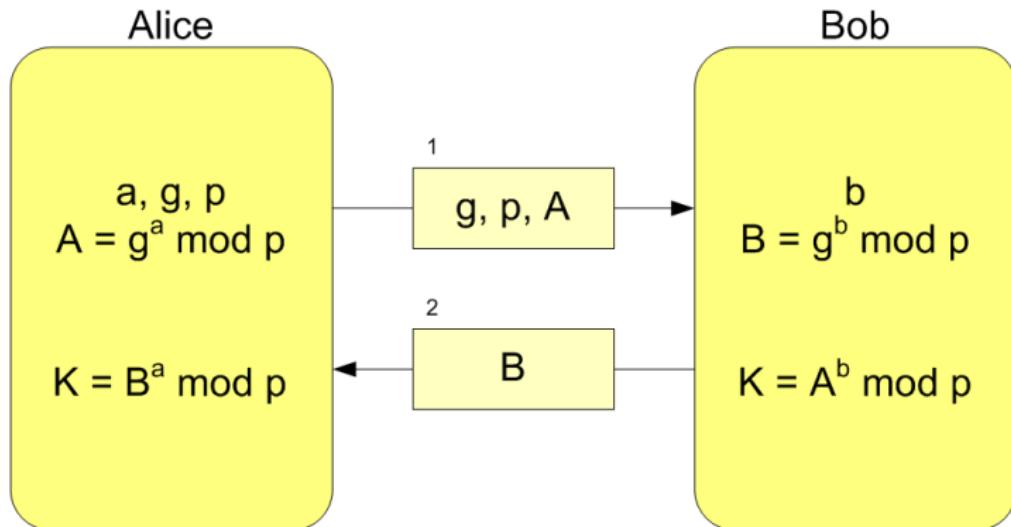


1 INTRODUCTION

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where

Échanger des clés

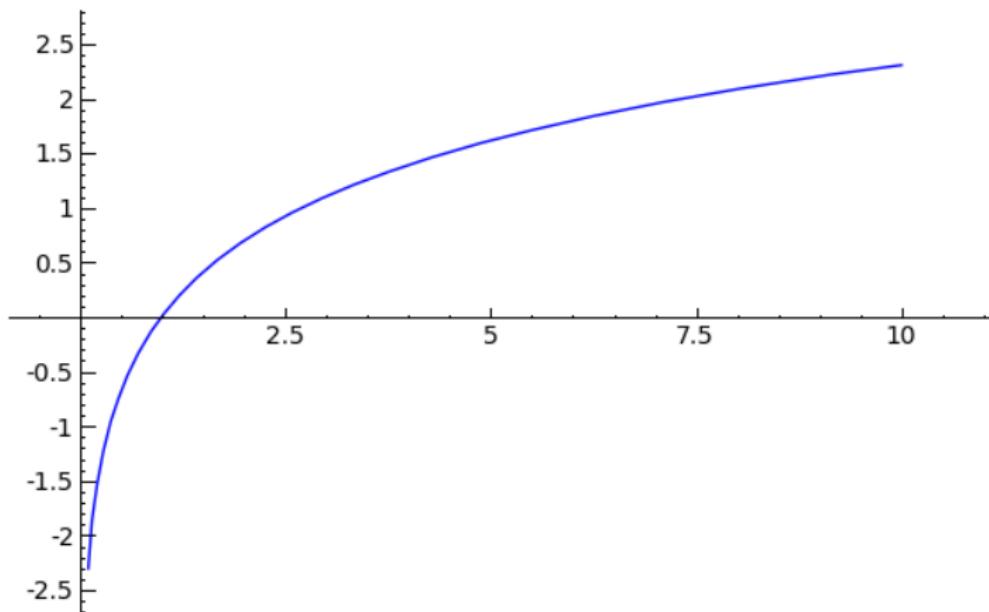
- 1976 Diffie, Hellman et Merkle publient le premier schéma d'échange de clés. Ils proposent (enfin) la notion de clé publique.
- Basé sur la difficulté du logarithme discret.



$$K = A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = g^{ab} \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = B^a \text{ mod } p$$

Logarithme discret : du plan continu au discret

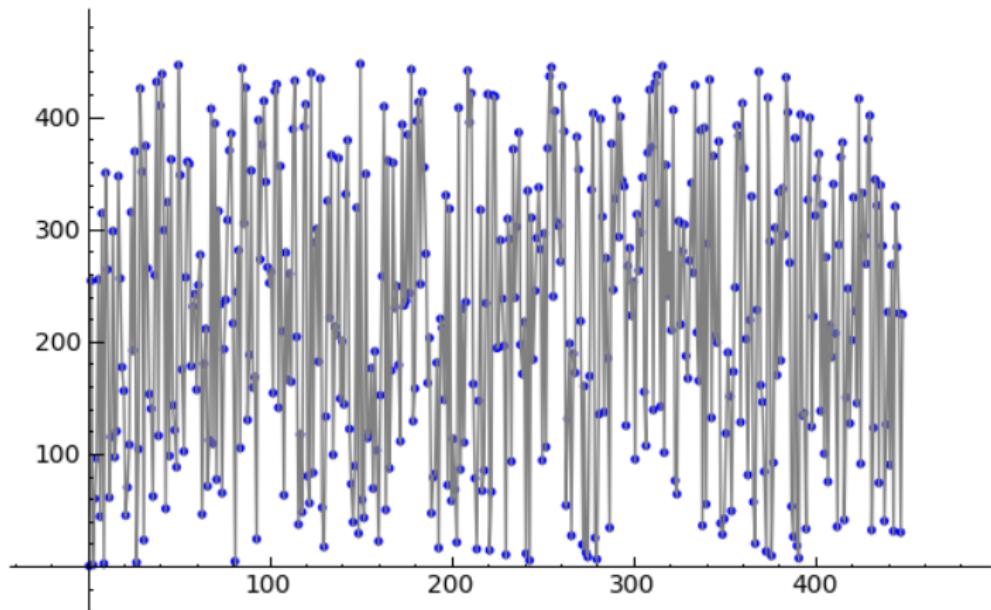
Étant donné e^x



Il est facile de calculer x

Logarithme discret : du plan continu au discret

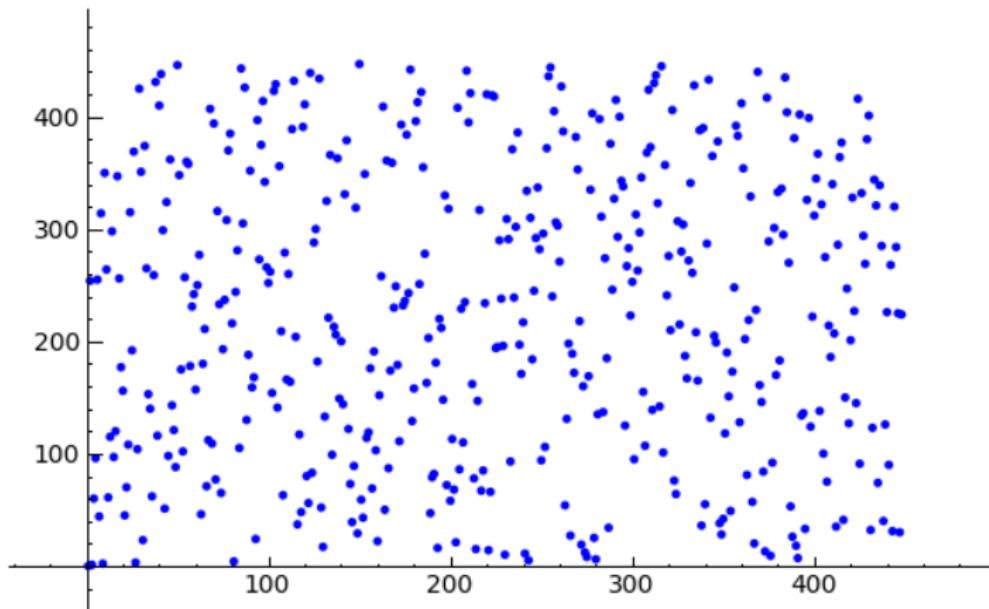
Étant donné $192 = g^x \pmod{449}$ ($g = 3$ générateur du groupe cyclique)



Il est **difficile** de calculer x

Logarithme discret : du plan continu au discret

Étant donné $192 = g^x \pmod{449}$ ($g = 3$ générateur du groupe cyclique)



Il est **difficile** de calculer x

La trappe de Diffie-Hellman

- ☞ Le nombre croissant d'échanges augmente le nombre de clés secrètes nécessaires.
- ☞ En 1975 ils proposent les grandes lignes de leur idée théorique pour éviter ce problème.

Définition

Une fonction $F : X \rightarrow Y$ est à sens unique avec trappe , si :

- F est à sens unique, i.e. elle est difficile à inverser, facile à calculer.
- La connaissance d'une information supplémentaire, appelée la trappe, rend le calcul pour tout $y \in \text{Im}(F)$ de $x \in X$ tel que $F(x) = y$ réalisable en temps polynomial.

Clé publique : F

Clé secrète : la trappe

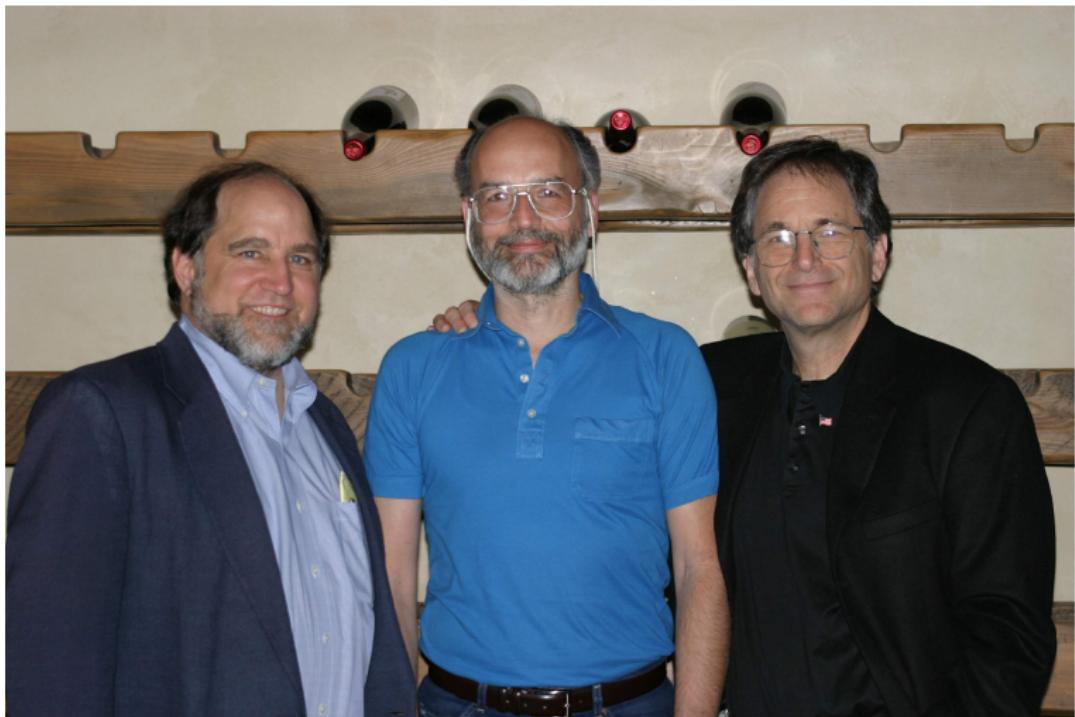
Rivest, Shamir, Adleman

1977 première instance d'un cryptosystème à clé publique : RSA.



Rivest, Shamir, Adleman

☞ 1977 première instance d'un cryptosystème à clé publique : RSA.



☞ 1977 première instance d'un cryptosystème à clé publique : RSA.

RSA

Soient p et q deux nombres premiers et $n = pq$ et deux entiers e, d tels que $ed = 1 \pmod{\phi(n)}$. L'instance $F_{(n,e)} : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z}$ qui à un élément x fait correspondre son exponentiation modulaire $x^e \pmod{n}$.

- F est bien à sens unique, difficulté basée sur la factorisation de n .
- La trappe est la connaissance de p, q et d .
- Chiffrement $x : x^e \pmod{n}$
- Déchiffrement $y : y^d \pmod{n}$

Complexités modulo N

☞ Tous les calculs se font modulo $N \Rightarrow$ les opérandes sont toutes majorées par N !

Complexité des algos naïfs

- Addition : $\mathcal{O}(\log(N))$
- Multiplication : $\mathcal{O}(\log(N)^2)$
- Division : comment divise-t-on modulo N ?

☞ Les complexités sont au plus quadratique en la taille de N mais que coûte le calcul d'un inverse ?

Complexités modulo N

☞ Tous les calculs se font modulo $N \Rightarrow$ les opérandes sont toutes majorées par N !

Complexité des algos naïfs

- Addition : $\mathcal{O}(\log(N))$
- Multiplication : $\mathcal{O}(\log(N)^2)$
- Division : comment divise-t-on modulo N ?

☞ Les complexités sont au plus quadratique en la taille de N mais que coûte le calcul d'un inverse ?

☞ Coût du calcul du PGCD ?

Calculer le PGCD : définition

Par définition on a

$$\text{PGCD}(a, b) := \prod_{p \in P} p^{\min(v_p(a)v_p(b))}$$

- ☞ Dépend de la factorisation de a et de b
- ☞ Le meilleure algorithme permettant de factoriser des entiers est de complexité sous-exponentielle. Le problème de la factorisation est donc assez difficile.
- ☞ Si l'on ne connaissait pas d'autres algorithmes permettant le calcul du PGCD il en serait de même...

Calculer le PGCD : Euclide

On cherche le plus grand diviseur entre 14 et 8.

- ☞ On a $14 = 8 + 6$ ainsi comme on cherche d divisant 8 et 14 il suffit de chercher d divisant 8 et 6, on recommence le procédé avec 8 et 6 etc.
- ☞ On s'arrête dès que l'un des entiers considéré est nul.

Lemmes immédiats

- Si d divise a et c et que $(a + b) = c$ alors d divise b .
- Le pgcd d'un élément $a \neq 0$ et 0 est a .

Calculer le PGCD : dans un anneau Euclidien

L'anneau A possède donc une fonction d permettant d'ordonner les éléments de cet anneau et de définir une division. Pour calculer le pgcd de deux éléments a et b tels que $d(b) < d(a)$.

Algorithme d'Euclide

Posant $r_0 = a$ et $r_1 = b$ on calcule jusqu'à obtenir un **reste nul**.

1 si $r_1 \neq 0$ on a $r_0 = r_1 q_1 + r_2$ avec $d(r_2) < d(r_1)$

2 si $r_2 \neq 0$ on a $r_1 = r_2 q_2 + r_3$ avec $d(r_3) < d(r_2)$

⋮

$n - 1$ si $r_{n-2} \neq 0$ on a $r_{n-2} = r_{n-1} q_{n-1} + r_n$ avec $d(r_n) < d(r_{n-1})$

n si $r_{n-1} \neq 0$ on a $r_{n-1} = r_n q_n + r_{n+1}$ avec $r_{n+1} = 0$ et $d(r_{n+1}) < d(r_n)$

PGCD par Euclide

Le **dernier reste non nul** r_n est égal au **pgcd de a et b** .

Algorithme d'Euclide Étendu

La démonstration du théorème de Bachet Bézout permet de déduire une formule de récurrence pour calculer les coefficients de Bézout de a et b et le $\text{pgcd}(a, b)$. Cette formule se traduit sous la forme d'un algorithme :

Algorithme d'Euclide étendu

Initialisation :

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \end{cases}$$

Calcul $i \geq 1$ tant que $r_i \neq 0$:

- $q_i = r_{i-1} \text{ div } r_i$
- $u_{i+1} = u_{i-1} - q_i u_i, v_{i+1} = v_{i-1} - q_i v_i, r_{i+1} = r_{i-1} - q_i r_i$

☞ Représentation matricielle :

$$\begin{pmatrix} u_{i+1} & v_{i+1} & r_{i+1} \\ u_i & v_i & r_i \end{pmatrix} = \begin{pmatrix} -q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_i & v_i & r_i \\ u_{i-1} & v_{i-1} & r_{i-1} \end{pmatrix}$$

Algorithme d'Euclide Étendu

Exemple : $a = 1234$ et $b = 896$.

i	r_i	r_{i+1}	q_i	u_i	u_{i+1}	v_i	v_{i+1}
0	1234	896	*	1	0	0	1

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \end{cases}$$

Algorithme d'Euclide Étendu

Exemple : $a = 1234$ et $b = 896$.

i	r_i	r_{i+1}	q_i	u_i	u_{i+1}	v_i	v_{i+1}
0	1234	896	*	1	0	0	1
1	896	338	1	*	*	*	*

Pour $i \geq 1$:

$$q_i = r_{i-1} \text{ div } r_i, \quad r_{i+1} = r_{i-1} \text{ mod } r_i$$

Algorithme d'Euclide Étendu

Exemple : $a = 1234$ et $b = 896$.

i	r_i	r_{i+1}	q_i	u_i	u_{i+1}	v_i	v_{i+1}
0	1234	896	*	1	0	0	1
1	896	338	1	0	1	1	-1

$$\begin{pmatrix} u_{i+1} & v_{i+1} & r_{i+1} \\ u_i & v_i & r_i \end{pmatrix} = \begin{pmatrix} -q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_i & v_i & r_i \\ u_{i-1} & v_{i-1} & r_{i-1} \end{pmatrix}$$

Algorithme d'Euclide Étendu

Exemple : $a = 1234$ et $b = 896$.

i	r_i	r_{i+1}	q_i	u_i	u_{i+1}	v_i	v_{i+1}
0	1234	896	*	1	0	0	1
1	896	338	1	0	1	1	-1
2	338	220	2	1	-2	-1	3
3	220	118	1	-2	3	3	-4
4	118	102	1	3	-5	-4	7
5	102	16	1	-5	8	7	-11
6	16	6	6	8	-53	-11	73
7	6	4	2	-53	114	73	-157
8	4	2	1	114	-167	-157	230
9	2	0	2	-167	448	230	-617

La relation de Bézout :

$$1234 \times (-167) + 896 \times 230 = 2$$

Nombre de boucles : le pire cas (pour une taille donnée) !

☞ Quand on ne peut pas faire mieux qu'utiliser la version simple d'Euclide (géométrique) : dans ce cas la division euclidienne est remplacée par une soustraction.

Exemple : pour $a = 13$ et $b = 8$:

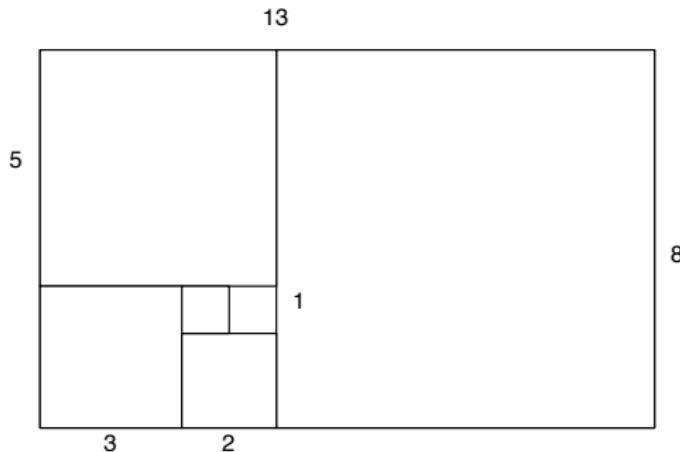
$$\begin{array}{rcl} 13 & = & 1 \times 8 + 5 \\ 8 & = & 1 \times 5 + 3 \\ 5 & = & 1 \times 3 + 2 \\ 3 & = & 1 \times 2 + 1 \\ 2 & = & 2 \times 1 + 0 \end{array}$$

☞ Le nombre de calcul est **maximal** car les quotients sont tous (sauf le dernier) réduits à 1.

Nombre de boucles : le pire cas (pour une taille donnée) !

☞ Quand on ne peut pas faire mieux qu'utiliser la version simple d'Euclide (géométrique) : dans ce cas la division euclidienne est remplacée par une soustraction.

Plus généralement : cette **propriété est vraie** pour tout couple F_{n+2}, F_{n+1} de **nombres de Fibonacci** successifs



☞ Par **définition**, ces couples sont les **pires cas**.

Nombre de boucles : les nombres de Fibonacci

Définition

$$\begin{cases} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_{n+1} + F_n \end{cases}$$

Propriétés

- $F_n = \frac{(\phi^n - \hat{\phi}^n)}{\sqrt{5}}$ avec $\phi = \frac{1+\sqrt{5}}{2}$ et $\hat{\phi} = \frac{1-\sqrt{5}}{2}$.
- Le calcul de $\text{pgcd}(F_{n+2}, F_{n+1})$ par l'algorithme d'Euclide demande exactement n itérations .
- De plus $\text{pgcd}(F_{n+2}, F_{n+1}) = 1$

Nombre de boucles : le cas général

Théorème

Soit $a > b > 0$. Si le calcul $\text{pgcd}(a, b)$ demande n itérations alors

$$a \geq F_{n+2} \text{ et } b \geq F_{n+1}$$

Corollaire (Théorème de Lamé)

$$n + 2 < \mathcal{O}(\log(a)) \text{ et } n + 1 < \mathcal{O}(\log(b))$$

- ☞ Le théorème de Lamé donne la meilleure constante possible dans le \mathcal{O} (5 dans le cas des chiffres décimaux).
- ☞ On se rappellera que le nombre d'itérations est de l'ordre de $\log(b)$ i.e la taille (le nombre de chiffres) de la plus petite opérande.

Algorithme d'Euclide : résultat final

Retour sur la complexité de l'algorithme d'Euclide

Pour $a \geq b > 0$

- Il y a $\mathcal{O}(\log(b))$ itérations de la boucle.
- Le coût d'une boucle est majoré par celui de la division
- En notant $t_i = \log(r_i)$ la taille de r_i le coût d'une boucle est

$$\mathcal{O}\left(\log\left(\frac{r_i}{r_{i+1}}\right)\log(r_{i+1})\right) = c_i t_{i+1} (t_i - t_{i+1})$$

Algorithme d'Euclide : résultat final

Retour sur la complexité de l'algorithme d'Euclide

Pour $a \geq b > 0$

- Il y a $\mathcal{O}(\log(b))$ itérations de la boucle.
- Le coût d'une boucle est majoré par celui de la division
- En notant $t_i = \log(r_i)$ la taille de r_i le coût d'une boucle est

$$\mathcal{O}\left(\log\left(\frac{r_i}{r_{i+1}}\right) \log(r_{i+1})\right) = c_i t_{i+1} (t_i - t_{i+1})$$

En notant n le nombre de boucles, le coût total du pgcd est majoré par la somme suivante (rappel : $t_i \geq t_{i+1}$)

$$\sum_{i=0}^n c_i t_{i+1} (t_i - t_{i+1}) \leq c t_1 \sum_{i=0}^n (t_i - t_{i+1}) \leq c t_1 t_0 \leq \mathcal{O}(\log(a) \log(b))$$

☞ complexité quadratique en la taille des entrées

Exponentiation modulaire

- entrée $a \in \{0, \dots, n - 1\}$ et $k \in \mathbb{N}$.
- sortie $a^k \pmod{n}$

Algorithme Naïf (pour ne pas dire crétin)

on fait n multiplications de a dans \mathbb{Z} et on réduit \pmod{n}

- complexité proportionnelle à k pas $\log(k) \Rightarrow$ exponentielle
- on utilise l'arithmétique sur les grands entiers pour réduire à la fin
 \Rightarrow il faut absolument penser à réduire pendant les calculs

☞ dans la suite on oublie le caractère modulaire, on commence par s'occuper de l'**exposant k** .

Exponentiation : square-and-multiply

Pour calculer une exponentiation, réfléchissons ($t_k = \ell(k) - 1$)

$$a^k = a^{k_0 2^0 + \dots + k_i 2^i + \dots + k_{t_k} 2^{t_k}}$$

avec $k_i \in \{0, 1\}$ en particulier on a deux cas possibles pour k_0 .

Exponentiation : square-and-multiply

Pour calculer une exponentiation, réfléchissons ($t_k = \ell(k) - 1$)

Si $k_0 = 0$ (i.e. k est pair) alors

$$a^{k_0 2^0 + \dots + k_{t_k} 2^{t_k}} = a^{k_1 2^1 + \dots + k_{t_k} 2^{t_k}}$$

$$= a^{2(k_1 2^0 + \dots + k_{t_k} 2^{t_k-1})}$$

$$= (a^{k_1 2^0 + \dots + k_i 2^{i+1} + \dots + k_{t_k} 2^{t_k-1}})^2 \quad \text{Square}$$

Exponentiation : square-and-multiply

Pour calculer une exponentiation, réfléchissons ($t_k = \ell(k) - 1$)

Si $k_0 = 1$ (i.e. k est impair) alors

$$a^{k_0 2^0 + \dots + k_i 2^i + \dots + k_{t_k} 2^{t_k}} = a^{1 + k_1 2^1 + \dots + k_{t_k} 2^{t_k}}$$

$$= (a^{k_1 2^1 + \dots + k_{t_k} 2^{t_k}}) \cdot a \quad \text{Multiply}$$

Exponentiation : square-and-multiply

Pour calculer une exponentiation, réfléchissons ($t_k = \ell(k) - 1$)

Square and Multiply dépendant de la valeur de k_0 :

$$(a^{\sum_{i=0}^{t_k-1} k_{i+1} 2^i})^2 \quad \text{Square si } k_0 = 0$$

$$(a^{\sum_{i=1}^{t_k} k_i 2^i}) \cdot a \quad \text{Multiply si } k_0 = 1$$

Algorithme récursif (écrire une version séquentielle de cet algorithme est un bon exercice)

- ☞ on ne va faire que $\log(k)$ multiplications
- ☞ Beaucoup mieux !

Dans le cas modulaire

On fait les calculs modulo un entier n .

- ☞ On peut considérer $k \leq n - 1$, pourquoi ?
- ☞ On fait tous les calculs modulo n donc toutes les opérandes sont de taille $\log(n)$

Conclusion

Complexité de l'exponentiation modulaire

$$\mathcal{O}(\log(n)\mathcal{M}(\log(n))) = \mathcal{O}(\log(n)^3)$$

Vérifications faciles

- Étant donné n, a, b on peut vérifier $n = a \times b$ en **multipliant** !
- Étant donné n, a, b, k on peut vérifier facilement $b = a^k \pmod n$ en calculant une **exponentiation modulaire** !

Problèmes difficiles correspondants

- FACT : Etant donner un grand entier n , donner sa factorisation ?
- DLP : Etant donner b, a, n , comment retrouver l'exposant k tel que $b = a^k \pmod n$?

☞ On ne connaît pas d'algorithme de complexité polynomiale résolvant ces problèmes !

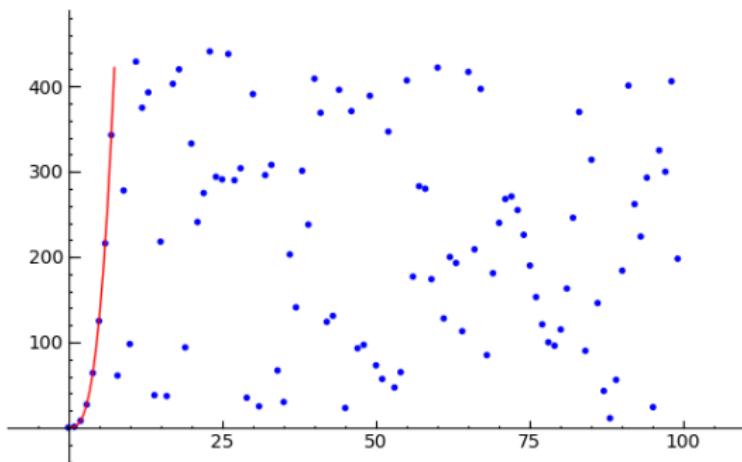
Part IV

Implémentation efficace de RSA

RSA

Basé sur la difficulté de calculer une racine e -ième mod un entier N de factorisation inconnue

Si on sait factoriser facilement alors ce problème se résout tout aussi facilement.



RSA

Soient p et q deux nombres premiers et $n = pq$. On choisit deux entiers e, d tels que $ed = 1 \pmod{\varphi(n)}$. L'application

$$F_{(n,e)} : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z} \text{ définie par } x \mapsto x^e \pmod{n}$$

- F est bien à sens unique, difficulté basée sur le calcul de la racine e -ième modulaire.
- La trappe est la connaissance de d (ou de manière équivalente p et q).

☞ on chiffre en faisant $x^e \pmod{n}$ on déchiffre en calculant $y^d \pmod{n}$

Preuve : On a $|\mathbb{Z}/n\mathbb{Z}^\times| = \varphi(n) = (p-1)(q-1) \Rightarrow ed = 1 + k \times \varphi(n) \dots$
Les messages qui ne sont pas inversibles mod n sont dangereux !

RSA : implémentation chiffrement

☞ Exponentiation modulaire

Square and Multply

Basé sur l'écriture binaire de l'exposant $e = \sum_{i=0}^{l-1} e_i 2^i$:

$$x^e = x^{\sum_{i=0}^{l-1} e_i 2^i} = \prod_{i=0}^{l-1} x^{e_i 2^i}$$

☞ On réduit la complexité de l'exponentiation modulo n à $\mathcal{O}(\log^3(n))$ (voir cours précédent).

RSA : implémentation déchiffrement

☞ dans ce cas on connaît p et q , idée couper les calculs en deux !

RSA et CRT1

On pré-calcule les valeurs :

- $u_p = q \times (q^{-1} \bmod p) \bmod n$
- $u_q = p \times (p^{-1} \bmod q) \bmod n$
- $d_p = d \bmod (p - 1)$
- $d_q = d \bmod (q - 1)$

Pour déchiffrer y et obtenir le message x d'origine on utilise le *CRT* :

- 1 $x_p = y^{d_p} \bmod p$
- 2 $x_q = y^{d_q} \bmod q$
- 3 $x = x_p u_p + x_q u_q \bmod n$

Théorème chinois des restes

Théorème

Soit m_1 et m_2 deux éléments de A un anneau euclidien. Si $\text{pgcd}(m_1, m_2) = 1$ alors nous avons l'isomorphisme d'anneau

$$A/m_1A \times A/m_2A \simeq A/(m_1m_2)A$$

Démonstration existentielle (à lire chez vous) :

Le morphisme d'anneau

$$\begin{aligned}\phi : A &\longrightarrow A/m_1A \times A/m_2A \\ e &\longmapsto (e \bmod m_1, e \bmod m_2)\end{aligned}$$

est surjectif.

Théorème chinois des restes

Théorème

Soit m_1 et m_2 deux éléments de A un anneau euclidien. Si $\text{pgcd}(m_1, m_2) = 1$ alors nous avons l'isomorphisme d'anneau

$$A/m_1A \times A/m_2A \simeq A/(m_1m_2)A$$

Démonstration existentielle (à lire chez vous) :

Le noyau de ϕ est donné par

$$\{e \in A \text{ tq } e = 0 \pmod{m_2} \text{ et } e = 0 \pmod{m_1}\}$$

ainsi le noyau est l'ensemble des multiples de m_1 et m_2 . Comme $\text{pgcd}(m_1, m_2) = 1$ on obtient le résultat.

Théorème chinois des restes

Théorème

Soit m_1 et m_2 deux éléments de A un anneau euclidien. Si $\text{pgcd}(m_1, m_2) = 1$ alors nous avons l'isomorphisme d'anneau

$$A/m_1A \times A/m_2A \simeq A/(m_1m_2)A$$

Démonstration constructive:

Comment reconstruire l'élément e dans $A/(m_1m_2)A$ à partir de son image (e_1, e_2) dans $A/m_1A \times A/m_2A$? (l'autre direction est triviale)

Théorème chinois des restes

Théorème

Soit m_1 et m_2 deux éléments de A un anneau euclidien. Si $\text{pgcd}(m_1, m_2) = 1$ alors nous avons l'isomorphisme d'anneau

$$A/m_1A \times A/m_2A \simeq A/(m_1m_2)A$$

Démonstration constructive:

Comment reconstruire l'élément e dans $A/(m_1m_2)A$ à partir de son image (e_1, e_2) dans $A/m_1A \times A/m_2A$? (l'autre direction est triviale)

Pour cela, nous utilisons la relation de Bézout entre m_1 et m_2 :

$$um_1 + vm_2 = 1$$

Théorème chinois des restes

Théorème

Soit m_1 et m_2 deux éléments de A un anneau euclidien. Si $\text{pgcd}(m_1, m_2) = 1$ alors nous avons l'isomorphisme d'anneau

$$A/m_1A \times A/m_2A \simeq A/(m_1m_2)A$$

Démonstration constructive:

Comment reconstruire l'élément e dans $A/(m_1m_2)A$ à partir de son image (e_1, e_2) dans $A/m_2A \times A/m_1A$? (l'autre direction est triviale)

En notant $M = m_1m_2$ il vient alors

$$e = e_1vm_2 + e_2um_1 \mod M$$

Théorème chinois des restes : algorithme général

On applique récursivement le procédé de résolution sur les équations du système. Il faut que $\text{pgcd}(m_i, m_j) = 1$ pour $i \neq j$.

$$\left\{ \begin{array}{l} x = e_1 \pmod{m_1} \\ x = e_2 \pmod{m_2} \\ \vdots \\ x = e_k \pmod{m_k} \end{array} \right\} \xrightarrow{\text{CRT}} \left\{ \begin{array}{l} x = c \pmod{M = m_1 m_2} \\ x = e_3 \pmod{m_3} \\ \vdots \\ x = e_k \pmod{m_k} \end{array} \right\}$$

Théorème chinois des restes : exemple sur \mathbb{Z}

Soit à résoudre le système

$$\begin{cases} x \equiv 2 \pmod{3} \\ x \equiv 3 \pmod{7} \\ x \equiv 8 \pmod{19} \end{cases}$$

Les entiers 3, 7 et 19 sont premiers deux à deux. CRT possible.

Initialisation :

- $c \leftarrow 2$
- $M \leftarrow 3$

Calculs :

- $c \leftarrow c \times u \times m_i + e_i \times v \times M$
- $M \leftarrow m_i \times M$
- $c \leftarrow c \pmod{M}$

avec $um_i + vM = 1$ (relation de Bézout):

i	e_i	m_i	u	v	c	M	$c \pmod{M}$
1					2	3	17
2	3	7	1	-2	$2 \times 1 \times 7 + 3 \times -2 \times 3 = -4$	$7 \times 3 = 21$	
3	8	19	10	-9	$17 \times 19 \times 10 + 8 \times -9 \times 21 = 1718$	$19 \times 21 = 399$	122

Les solutions sont de la forme $122 + 399k$ pour $k \in \mathbb{Z}$.

CRT et RSA avec p et q de même taille

Lors du déchiffrement, on connaît p et q et l'on doit calculer

$$x = y^d \mod n$$

Cette opération prend $\simeq \log(d) \log(n)^2 \simeq \log(n)^3$ mais on ne se sert pas de la connaissance de p et q !

Mais d'après le CRT on a

$$\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$$

CRT et RSA avec p et q de même taille

Mais d'après le CRT on a

$$\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$$

On peut donc faire les calculs à droite :

$$x_p = y^d \pmod p = y^{d \pmod{(p-1)}} \pmod p \text{ et } x_q = y^{d \pmod{(q-1)}} \pmod q$$

et on reconstruit la solution modulo n :

$$x = x_p u_p + x_q u_q \pmod n$$

Ici on fait $\simeq 2 \times \log(\sqrt{n})^3 \simeq \frac{1}{4} \log(n)^3$ pour le calcul des exponentiations et $2 \times \log(\sqrt{n})^2 + \log(n)$ pour la reconstruction.

☞ $\log(n)^3 >> \frac{1}{4} \log(n)^3$

☞ Temps d'exécution de 2 à 4 fois plus rapide avec le CRT !

Part V

Corps finis et Applications

Corps finis : Propriétés

A partir des corps premiers $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ on veut pouvoir construire des corps de même caractéristique p mais de cardinal plus grand.

☞ On utilise la rupture

Définition

Soit P un polynôme irréductible de degré n

$$\mathbb{F}_{p^n} := \mathbb{F}_p[X]/(P)$$

Corps finis : Exemple

Théorèmes

- \mathbb{F}_{p^n} est de cardinal p^n .
- Pour tout p premier et tout entier $n > 0$ il existe un polynôme irréductible de degré n dans \mathbb{F}_p .

Exemple : $\mathbb{F}_p = \mathbb{Z}/2\mathbb{Z}$ et $P = X^2 + X + 1$

$$\mathbb{F}_4 = \mathbb{F}_2[X]/(X^2 + X + 1)$$

Si on note ω une racine de P (image de X dans $\mathbb{F}_2[X]/P$) on a :

$$\mathbb{F}_4 = \{0, 1, \omega, \omega + 1\}$$

Corps finis : Exemple

Exemple : $\mathbb{F}_p = \mathbb{Z}/2\mathbb{Z}$ et $P = X^2 + X + 1$

$$\mathbb{F}_4 = \mathbb{F}_2[X]/(X^2 + X + 1)$$

Si on note ω une racine de P (image de X dans $\mathbb{F}_2[X]/P$) on a :

$$\mathbb{F}_4 = \{0, 1, \omega, \omega + 1\}$$

Pour calculer la table de multiplication dans \mathbb{F}_4 on revient au calcul dans $\mathbb{F}_2[X]/P$:

\times	0	1	ω	$\omega + 1$
0	0	0	0	0
1	0	1	ω	$\omega + 1$
ω	0	ω	$\omega + 1$	1
$\omega + 1$	0	$\omega + 1$	1	ω

☞ Applications Crypto Asymétrique

Théorème

Le groupe multiplicatif $\mathbb{F}_{p^n}^\times$ est **cyclique** d'ordre $p^n - 1$.

- ☞ Il existe un $\omega \in \mathbb{F}_{p^n}$ tel que tous les éléments de $\mathbb{F}_{p^n}^\times$ puissent s'écrire sous la forme ω^i .
- ☞ Ce groupe multiplicatif peut être utilisé comme support de DLP et donc pour faire de l'échange de clés.

Corps finis : Application

☞ Applications Crypto Symétrique

La boite S de AES est définie comme une fonction de \mathbb{F}_{2^8} dans lui-même avec

$$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$$

et S définie en deux temps : $S(x) = A(\text{Inv}(x)) + C$ avec

$$\text{Inv}(x) = x^{-1} \text{ si } x \neq 0 \text{ et } \text{Inv}(0) = 0$$

A est une matrice et C un vecteur (application affine).

Part VI

Signer un message avec RSA et le DLP

Signer un message

- ☞ La signature permet d'authentifier l'expéditeur d'un message

Problème de l'authentification

Étant donné un message m et sa signature s , on doit être capable de vérifier l'expéditeur de m à partir de données publiques.

- ☞ Les problèmes RSA et DLP peuvent être utilisés à cet usage.

Signer avec RSA

- ☞ La signature permet d'authentifier l'expéditeur d'un message

La signature RSA

Soit $N = pq$ un entier RSA et (e, d) un couple d'exposants de chiffrement/déchiffrement choisis par l'expéditeur.

- L'expéditeur publie N et d
- Un message $m \in (\mathbb{Z}/N\mathbb{Z})^\times$ envoyé par l'expéditeur aura pour signature

$$s_m = m^e \mod N$$

- Le destinataire recevra le couple (m, s) et authentifiera l'expéditeur à partir ses clés publiques (N, d) en vérifiant

$$m = s_m^d \mod N$$



ATTENTION il ne faut pas mélanger chiffrement et signature RSA n'importe comment.

Fonctions de hachage

Problème signature RSA : la signature est de la taille du message !

Fonction de hachage cryptographique

Une fonction de $0, 1^*$ dans $[0, 2^t - 1]$ (avec $t = 160, 256, 512$ etc) est dite de hachage si elle vérifie

- difficile à inverser (reconstruire un message à partir d'une signature)
- difficile de reconstruire un second message de même signature (seconde préimage)
- la probabilité de trouver deux messages de même signature est très faible (collision)

Exemples : [MD5](#), SHA1, SHA2 (SHA-256, SHA-512), [SHA3](#), ...

- ☞ Permet de vérifier l'intégrité d'un fichier (paquets linux par exemple)
- ☞ Plutôt que signer un message, on signe son empreinte après application d'une fonction de hachage (compression).

- ☞ Elle repose sur une modification du chiffrement de ElGamal
- ☞ Nécessite l'utilisation d'une fonction de hachage de l'ensemble des messages dans le groupe support du DLP.

La signature DSA sur les corps finis

La génération des clés :

- une fonction de hachage H (e.g. SHA1)
- p, q deux premiers de tailles respectives 1024 et 160 bits tels que $p - 1$ est un multiple de q
- g un entier d'ordre q modulo p
- un problème DLP : $h = g^t \pmod p$ (le groupe support est d'ordre q)

Signer avec DLP

- Elle repose sur une modification du chiffrement de ElGamal
- Nécessite l'utilisation d'une fonction de hachage de l'ensemble des messages dans le groupe support du DLP.

La signature DSA sur les corps finis

L'expéditeur publie les clés publiques (p, q, g, h) et conserve t comme clé secrète. La fonction de hachage H est aussi une donnée publique. Pour signer un texte m , l'expéditeur procède comme suite

- ❶ Il choisit un entier k_m au hasard (pour chaque message)
- ❷ $r = (g^{k_m} \bmod p) \bmod q$
- ❸ $s = k_m^{-1}(H(m) + t \times r) \bmod q$
- ❹ si r et s sont non nuls alors la signature de m est le couple (r, s) sinon retourner à la première étape.

➤ ATTENTION k_m doit vraiment être choisi au hasard ! (voir TD)

- ☞ Elle repose sur une modification du chiffrement de ElGamal
- ☞ Nécessite l'utilisation d'une fonction de hachage de l'ensemble des messages dans le groupe support du DLP.

La signature DSA sur les corps finis

Le destinataire vérifiera l'authenticité de l'expéditeur en vérifiant

$$v = r \mod q$$

où v est calculé comme suit :

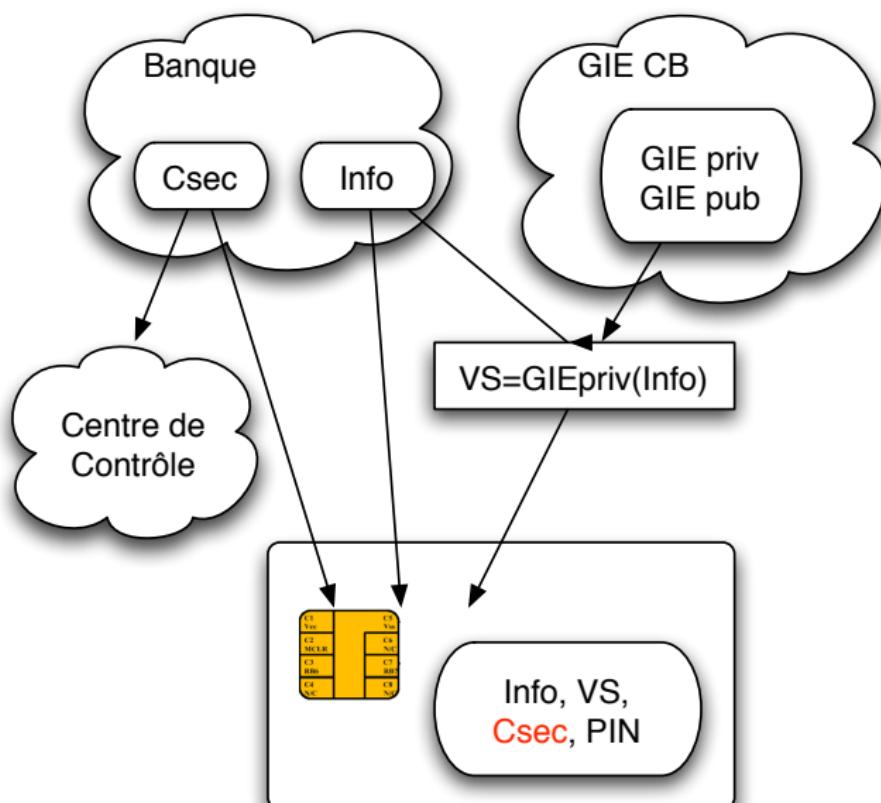
- ① $u_1 = H(m) \times s^{-1} \mod q$
- ② $u_2 = r \times s^{-1} \mod q$
- ③ $v = (g^{u_1} h^{u_2} \mod p) \mod q$

Part VII

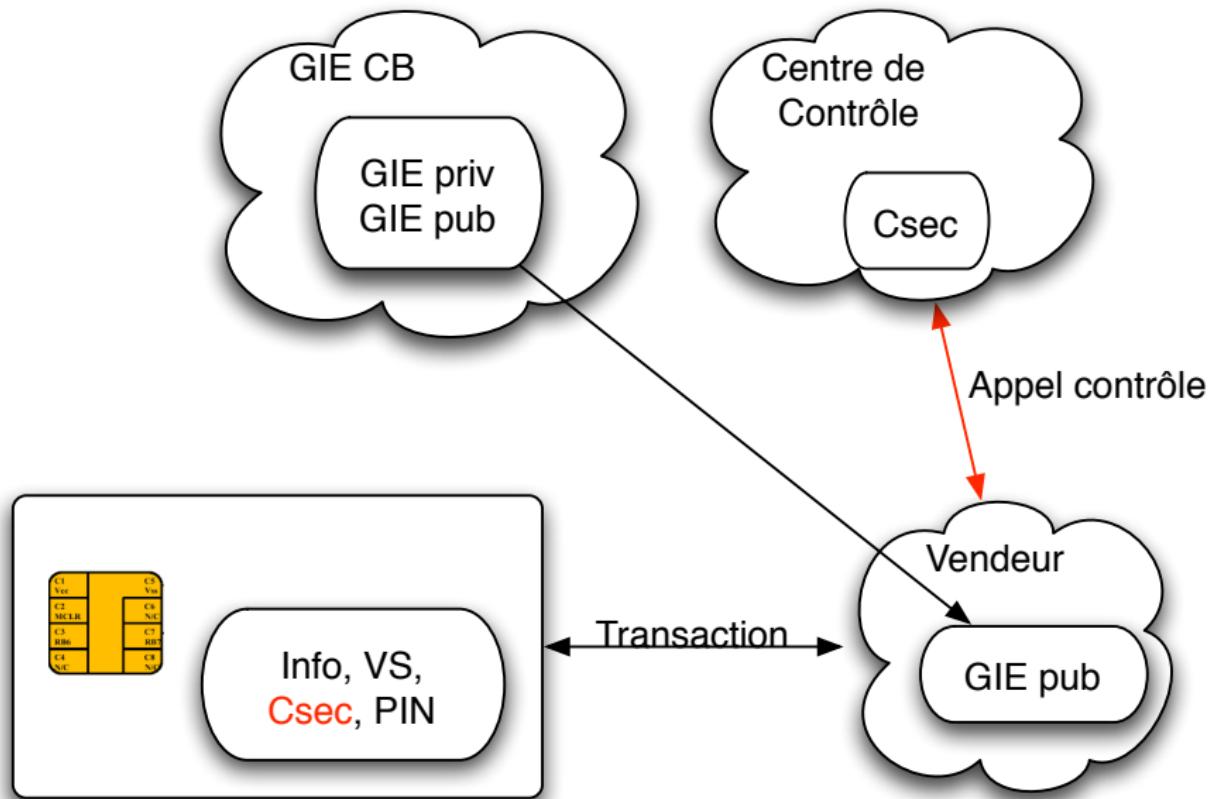
RSA dans des cartes bancaires (ancien protocole B0 utilisé avant 2004)

Création de la carte bancaire

- La signature RSA permet d'authentifier une transaction



Utilisation de la carte bancaire



Utilisation de la carte bancaire

- **Authentification de la carte** : le terminal lit Info et VS sur la carte et vérifie la signature à l'aide de GIpub.
- **PIN** : le PIN est envoyé à la carte pour calcul en place et vérification.
- **Authentification par Csec** : appel téléphonique à l'organisme de contrôle pour vérifier de la présence la clé secrète.



Attaque Serge Humpich (1997, factorisation 32 bits), divulgation des YES card (2000)

Part VIII

Une fausse bonne nouvelle idée ?

OTP par carte bancaire

- ☞ CAP (Chip Authentication Protocol) permet une authentification forte



OTP par carte bancaire

☞ CAP (Chip Authentication Protocol) permet une authentification forte



Attaque Side-Channel possible ?

Part IX

Groupes des points rationnels sur
courbes elliptiques

Rappel sur le DLP

- ☞ sécurité basée sur la difficulté de résoudre le problème de calculer un logarithme discret

On a vu comment utiliser le groupe multiplicatif de:

- Corps premiers : $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$
- Corps finis généraux : \mathbb{F}_q



Les meilleures attaques connues sont sous-exponentielles pour les corps premiers. Une attaque de janvier 2013 implique le rejet total du cas des corps finis généraux de petites caractéristiques.

☞ Un groupe plus sûr ?

Courbes elliptiques

☞ Lieu géométrique !

Sur $\mathbb{K} = \mathbb{F}_{p^k}$ avec $p > 3$

Pour a et b dans \mathbb{K} vérifiant

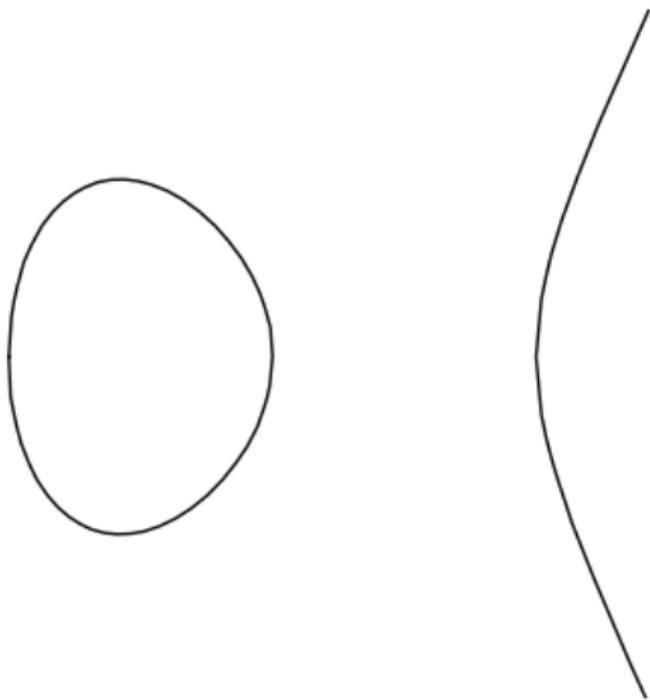
$$\Delta := -16(4a^3 + 27b^2) \neq 0$$

On définit la **courbe elliptique** $\mathcal{E}_{a,b}$ sur \mathbb{K} par l'ensemble des points $(x, y) \in \mathbb{K}^2$ vérifiant l'équation:

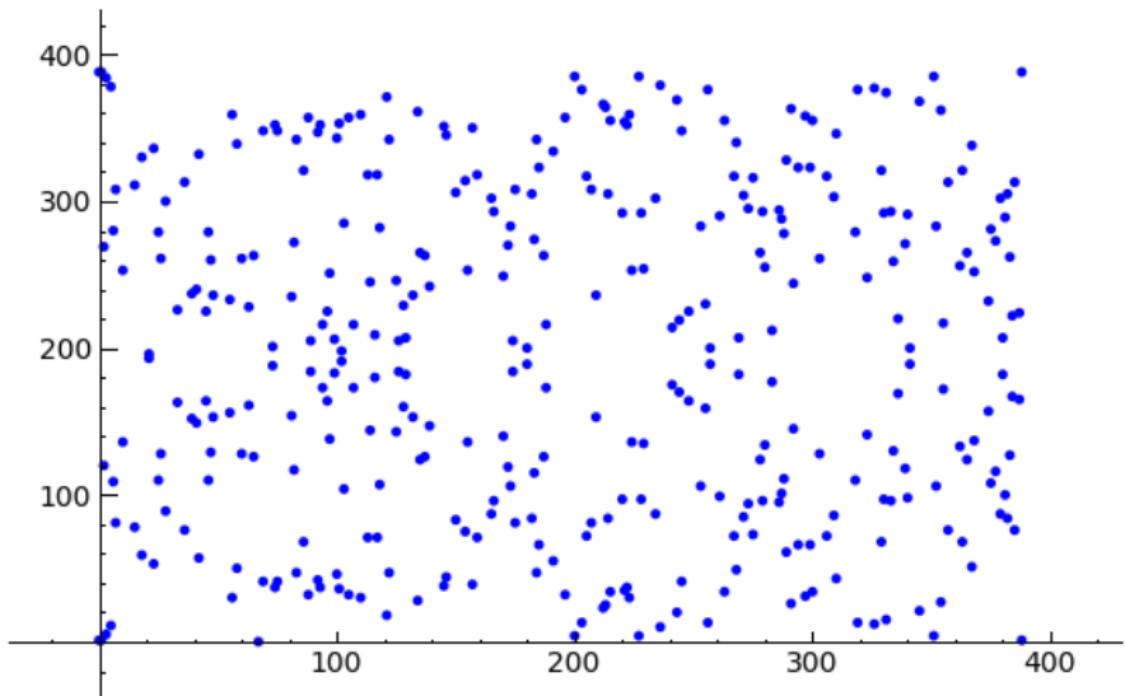
$$y^2 = x^3 + ax + b$$

☞ Définition équivalente dans le cas de la caractéristique 3 et 2.

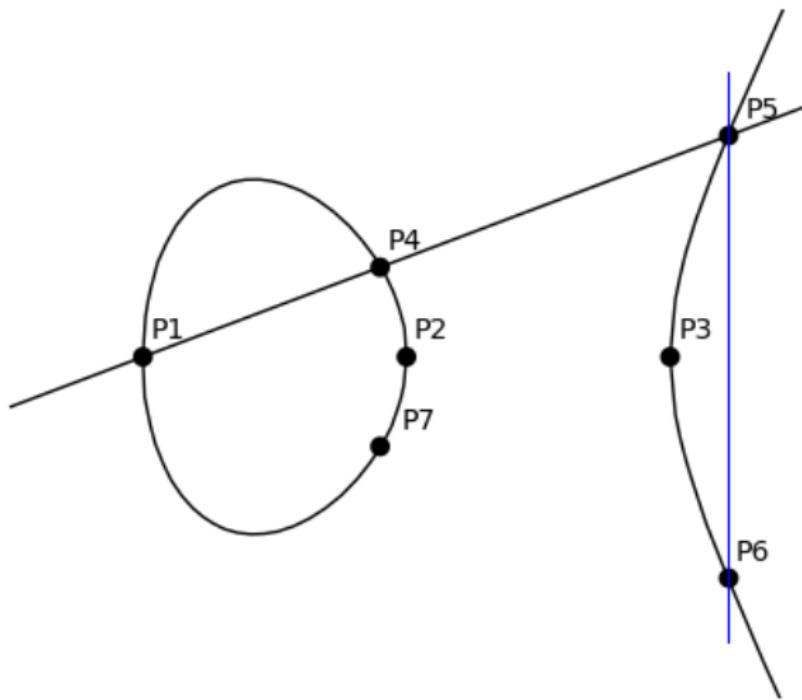
Courbes elliptiques sur \mathbb{R}^2



Courbes elliptiques sur \mathbb{F}_p , $p = 389$, $y^2 = x^3 - x + 1$



Structure de groupe $E_{a,b} = \mathcal{E}_{a.b} \cup \mathcal{O}$



Structure de groupe $E_{a,b} = \mathcal{E}_{a,b} \cup \mathcal{O}$

Groupe additif commutatif de la courbe

Soit $P_1 = (x_1, y_1), Q = (x_2, y_2) \in \mathcal{E}_{a,b}$

- Point à l'infini \mathcal{O} est l'élément neutre :

$$P + \mathcal{O} = \mathcal{O} + P = P$$

- Si $x_2 = x_1$ et $y_2 = -y_1$ alors $P + Q = \mathcal{O}$
- $P + Q = (x_3, y_3)$ avec

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

et $\lambda = \frac{3x_1^2 + a}{2y_1}$ si $P = Q$ et $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ sinon.

☞ Ne pas oublier de rajouter le point à l'infini pour former le groupe !

Un exemple concret

On étudie le groupe construit avec $\mathbb{K} = \mathbb{F}_{11}$ et $\mathcal{E}_{1,6}$: $y^2 = x^3 + x + 6$. Pour chaque $x \in \{0, \dots, 10\}$ on cherche à résoudre une équation quadratique modulo 11. Pour faciliter ce travail, on peut commencer par lister l'ensemble des carrés dans \mathbb{K} .

y	$z = y^2 \pmod{11}$
0	0
± 1	1
± 2	4
± 3	9
± 4	5
± 5	3

Un exemple concret

On étudie le groupe construit avec $\mathbb{K} = \mathbb{F}_{11}$ et $\mathcal{E}_{1,6}$: $y^2 = x^3 + x + 6$. Pour chaque $x \in \{0, \dots, 10\}$ on cherche à résoudre une équation quadratique modulo 11. Pour faciliter ce travail, on peut commencer par lister l'ensemble des carrés dans \mathbb{K} .

y	$z = y^2 \pmod{11}$	x	$z = x^3 + x + 6 \pmod{11}$	$\left(\frac{z}{11}\right)$	y
0	0	0	6	-1	
± 1	1	1	8	-1	
± 2	4	2	5	1	± 4
± 3	9	3	3	1	± 5
± 4	5	4	8	-1	
± 5	3	5	4	1	± 9
		6	8	-1	
		7	4	1	± 9
		8	9	1	± 3
		9	7	-1	
		10	4	1	± 9

Un exemple concret

On étudie le groupe construit avec $\mathbb{K} = \mathbb{F}_{11}$ et $\mathcal{E}_{1,6} : y^2 = x^3 + x + 6$.

Pour chaque $x \in \{0, \dots, 10\}$ on cherche à résoudre une équation quadratique modulo 11. Pour faciliter ce travail, on peut commencer par lister l'ensemble des carrés dans \mathbb{K} .

y	$z = y^2 \pmod{11}$
0	0
± 1	1
± 2	4
± 3	9
± 4	5
± 5	3

x	$z = x^3 + x + 6 \pmod{11}$	$(\frac{z}{11})$	y
0	6	-1	
1	8	-1	
2	5	1	± 4
3	3	1	± 5
4	8	-1	
5	4	1	± 9
6	8	-1	
7	4	1	± 9
8	9	1	± 3
9	7	-1	
10	4	1	± 9

☞ 12 points rationnels + le point à l'infini = un groupe d'ordre 13, donc cyclique engendré par tout élément autre que le neutre.

Exemple concret

Calculons explicitement avec $\mathbb{K} = \mathbb{F}_{11}$, $\mathcal{E}_{1,6}$: $y^2 = x^3 + x + 6$.

Soit $P = (2, 7)$ on veut calculer $[2]P = P + P$ et $[3]P$.

$$\begin{aligned}\lambda &= (3 \times 2^2 + 1)(2 \times 7)^{-1} \pmod{11} \\ &= 2 \times 3^{-1} \pmod{11} \\ &= 2 \times 4 \pmod{11} \\ &= 8\end{aligned}$$

$$[2]P = (8^2 - 2 - 2 \pmod{11}, 8(2 - 5) - 7 \pmod{11}) = (5, 2)$$

Pour $[3]P$ on a $[3]P = [2]P + P$, on calcule l'autre λ

$$\begin{aligned}\lambda &= (7 - 2)(2 - 5)^{-1} \pmod{11} \\ &= 5 \times 8^{-1} \pmod{11} \\ &= 5 \times 7 \pmod{11} \\ &= 2\end{aligned}$$

$$[3]P = (2^2 - 5 - 2 \pmod{11}, 2(5 - 8) - 2 \pmod{11}) = (8, 3)$$

Carrés modulo p , symbole de Legendre

- Etant donné un entier a on ne connaît pas d'algorithme déterministe qui calcule b tel que $a = b^2 \pmod{p}$ en temps polynomial (en $\log p$).
- On peut décider qu'un élément est *résidu quadratique* (un carré modulo p) en temps polynomial

Définition : Symbole de Legendre

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{si } a \text{ est un résidu quadratique } \pmod{p} \\ -1 & \text{si } a \text{ n'est pas un résidu quadratique } \pmod{p} \\ 0 & \text{si } p \text{ divise } a \end{cases}$$

Propriétés

$$\left(\frac{a_1 a_2}{p}\right) = \left(\frac{a_1}{p}\right) \left(\frac{a_2}{p}\right) \text{ et } \left(\frac{a}{p}\right) = \left(\frac{a \pmod{p}}{p}\right)$$

Carrés modulo p , symbole de Legendre

- ☞ Etant donné un entier a on ne connaît pas d'algorithme déterministe qui calcule b tel que $a = b^2 \pmod{p}$ en temps polynomial (en $\log p$).
- ☞ On peut décider qu'un élément est *résidu quadratique* (un carré modulo p) en temps polynomial

Réciprocité quadratique

Soit p et q deux premiers distincts > 2 (le cas $p = 2$ est trivial)

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & \text{si } p \equiv 1 \pmod{4} \\ -1 & \text{si } p \equiv -1 \pmod{4} \end{cases},$$

$$\left(\frac{2}{p}\right) = \begin{cases} 1 & \text{si } p \equiv \pm 1 \pmod{8} \\ -1 & \text{si } p \equiv \pm 3 \pmod{8} \end{cases}$$

$$\left(\frac{p}{q}\right) = \begin{cases} \left(\frac{q}{p}\right) & \text{si } p \equiv 1 \pmod{4} \text{ ou } q \equiv 1 \pmod{4} \\ -\left(\frac{q}{p}\right) & \text{si } p \equiv 3 \pmod{4} \text{ et } q \equiv 3 \pmod{4} \end{cases}$$

Structure de groupe pour $\mathbb{K} = \mathbb{F}_q$ avec $q = p^k$ et $p > 3$

Théorème de Hasse

Soit $q = p^k$.

$$q + 1 - 2\sqrt{q} \leq |E_{a,b}| \leq q + 1 + 2\sqrt{q}$$

Théorème de structure

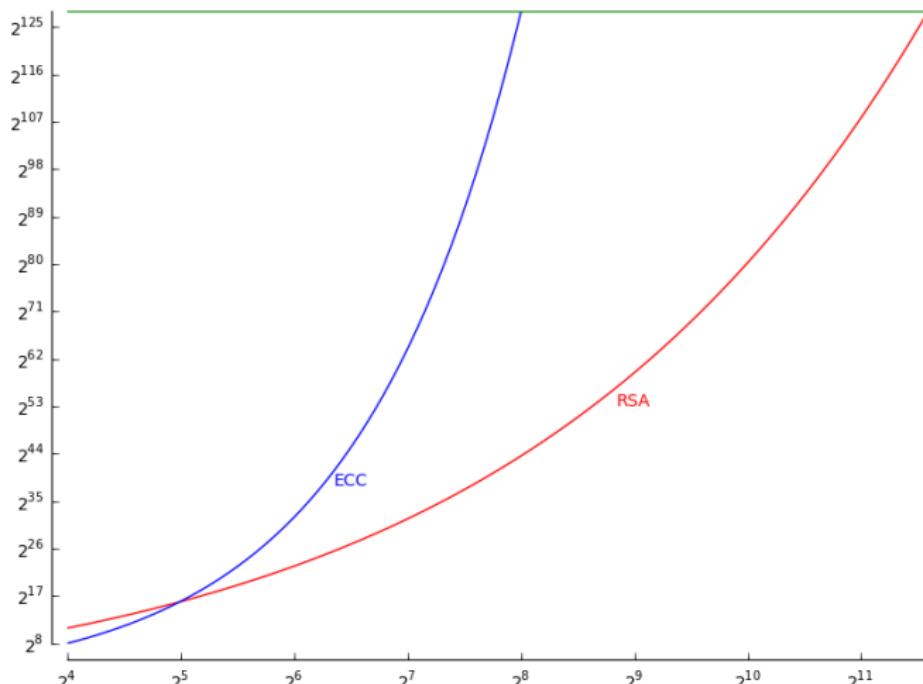
$$(E_{a,b}, +) \simeq (\mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z}, +)$$

où d_1 divise d_2 et d_1 divise $q - 1$. En d'autres termes, le groupe construit à partir d'une courbe elliptique est soit cyclique (le cas $d_1 = 1$), soit le produit de deux groupes cycliques.

☞ Pour la cryptographie on choisit des courbes elliptiques telles que $(E_{a,b}, +)$ soit cyclique (le cas $d_1 = 1$) ou au moins contient un grand groupe cyclique (Pohlig-Hellman).

Avantages des courbes elliptiques

- Pas d'attaque connue de complexité sous-exponentielle
- L'arithmétique peut être rendue très efficace
- Clés plus petites pour des sécurités équivalentes



La signature DSA sur EC

La génération des clés :

- une fonction de hachage H (SHA-1, 2)
- une courbe \mathcal{C} support
- un point G d'ordre q un grand premier
- un problème DLP : $H = [t]G$ (le groupe support est d'ordre q)

La signature DSA sur EC

L'expéditeur publie les clés publiques (\mathcal{C}, G, H) et conserve t comme clé secrète. La fonction de hachage H est aussi une donnée publique. Pour signer un texte m , l'expéditeur procède comme suite

- 1 Soit z les $\log(q)$ bits de poids forts de $H(m)$
- 2 Il choisit un entier $1 \leq k_m < q$ au hasard (pour chaque message)
- 3 $(x_1, y_1) = [k_m]G$
- 4 $r = x_1 \pmod q$
- 5 $s = k_m^{-1}(z + t \times r) \pmod q$
- 6 si r et s sont non nuls alors la signature de m est le couple (r, s) sinon retourner à la première étape.

ATTENTION k_m doit vraiment être choisi au hasard !

La signature DSA sur EC

Le destinataire vérifiera l'authenticité de l'expéditeur en vérifiant

$$v = r \mod q$$

où v est calculé comme suit :

- ➊ Soit z les $\log(q)$ bits de poids forts de $H(m)$
- ➋ $u_1 = z \times s^{-1} \mod q$
- ➌ $u_2 = r \times s^{-1} \mod q$
- ➍ $(x_1, y_1) = [u_1]G + [u_2]H$
- ➎ $v = x_1 \mod q$

mais parfois mal utilisées...

En janvier 2011 , une équipe de hackers allemands **fail0verflow** ont démontré qu'une très grande entreprise informatique utilisait des cryptosystèmes sans les comprendre ... ERROR!



mais parfois mal utilisées...

➡ En janvier 2011 , une équipe de hackers allemands  ont démontré qu'une très grande entreprise informatique utilisait des cryptosystèmes sans les comprendre ... ERROR!

Sony's ECDSA code

```
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
              // guaranteed to be random.
}
```