8 février 2020 - Fondation Arts et Métiers à LIANCOURT (60)

# Cryptographie homomorphe : vers le vote électronique à grande échelle ?


***Pr Loÿs THIMONIER***

*https://www.researchgate.net/profile/Loys_Thimonier*

**algorithmes, informatique mathématique, cryptographie quantique**

au laboratoire **EPROAD** (**E**co**PR**océdés, **O**ptimisation, et **A**ide à la **D**écision)

*Faculté des Sciences de l'Université de Picardie – AMIENS*

# Motivations de cet exposé

## * Diffusion de la connaissance

- donner des clés introductives à **d'autres mondes de nombres courants en sciences modernes**
- esquisses élémentaires des aspects en jeu : *mathématiques, logiques*, *informatiques*…

## * Référendums réclamés lors des mouvements sociaux dès 2018 en France

**coûts élevés** (papier, impression, personnels) d'impact important sur le budget de l'État

=> penser au **vote électronique ?**

(vote "en ligne" depuis un ordinateur, ou la machine à voter électronique au bureau de vote)

## * Avantages du vote par Internet

- **bien moins onéreux**
- si vote papier => longs déplacements (exemple : Français de l'étranger dans les ambassades)
- si vote papier dans pays à faible densité de population (exemple : **Australie**)
- si nécessité apparue de consultation plus fréquente des électeurs…

# Motivations de cet exposé

*** Problème global de la confidentialité**

- posé par **vote papier** et **vote électronique**

- origine du **secret** du vote : **594 av. J.-C.** à Athènes

. une des premières formes connues de votes à bulletin secret

. tous les Athéniens (*sauf femmes, esclaves, étrangers*) => ***Ecclésia*** (assemblée citoyenne)

. vote des lois : souvent réalisés à main levée, mais aussi à **bulletins secrets**…

- systèmes en ligne à risque pour la **sécurité**, autres que le vote électronique :

*opérations bancaires, achats en ligne, remplissage des feuilles d'impôt…*

*** Plus de contraintes spécifiques** pour le vote électronique

exemple : informations sur compte bancaire pouvant se donner à un proche (pour achat, transfert…)

mais situation non concevable en cas de vote

# Motivations de cet exposé

## - Problème de la confidentialité entre client et fournisseur -

* le client stocke ses **données initiales** en clair sous forme de **données cryptées** numériques (auxquelles s'appliquent **additions** et **multiplications**)

* il les confie à un service extérieur (***cloud computing***) pour stockage et traitements à sa demande

* il souhaite des recherches (*moyennes, tris, fréquences*…) sur les données confiées
=> traitements (grâce au **cryptage homomorphe**) d'informations illisibles par le service qui transmet ensuite les résultats **cryptés** au client, seul à pouvoir les **décrypter** avec sa **clé secrète** devant être quasiment **impossible à découvrir**

* Problème considéré depuis toujours comme le **Graal** de la cryptographie !

# Motivations de cet exposé

## - Influence sur le problème du vote électronique -

* analogie : **stockage** des votes **secrets**, **traitements** des votes (comptage, calcul du vainqueur)

* progrès des recherches sur le **problème de la confidentialité entre client et fournisseur** :
**GENTRY** (**Stanford** University) trouve en **2009** le ***1er cryptage complètement homomorphe***
=> on est arrivé **théoriquement** à résoudre le problème de **confidentialité client fournisseur !**

* nombreux travaux actuels d'*optimisation* pour arriver à sa **mise en pratique effective** :
- cryptage utilisant des nombres de **plusieurs millions** de chiffres binaires !
- **clés nécessaires très longues** (implémentation initiale => clés de plus de **2 Go**, alors que le cryptage **AES** utilise des clés de **256 bits**), **algorithmes de traitement de données trop lents**

# - SOMMAIRE -

# A - VOTE ELECTRONIQUE ?

**A1 – Survol de propriétés à vérifier par le vote électronique**

**A2 - Historique du vote électronique dans le monde**

**A3 - Perspectives de progrès logiciels**

# A1 – Survol de propriétés à vérifier par le vote électronique

*** 2 propriétés essentielles à vérifier**

- la ***confidentialité*** (secret du vote), qui fait intervenir la ***cryptographie***
- la ***vérifiabilité***, qui assure un dépouillement tenant compte de tout bulletin

*** protection contre les fraudes**

- ***coercition*** : achat de votes, intimidation
- ***attaque virale*** du logiciel…

# A2 - Historique du vote électronique en France

*** Election présidentielle de 2007 :**

- machines à voter électroniques dans 83 villes pour 1,5 M d'électeurs (3 % du corps électoral)
- premier tour => problèmes techniques et juridiques
  (2 machines installées dans chaque bureau de vote de Reims pour écourter le temps d'attente

*** Vote des Français à l'étranger aux élections législatives 2012 et consulaires 2014**

- mode : soit à l'urne au consulat, soit par correspondance (moins sécurisé que le vote en ligne!)
  avec bulletin papier et enveloppes pour ceux loin du consulat, soit en ligne par Internet
- vote en ligne aux législatives : 1 votant sur 2, n'a pas diminué le taux d'abstention

*** Rapport très réservé de 2 sénateurs en 2014**

- opacité du fonctionnement des machines à voter => interdiction demandée
- imperfections du vote en ligne => non en métropole, maintenu pour les Français de l'étranger

*** Poursuite du vote électronique depuis 2014 en cas d'enjeux et effectifs faibles**

- novembre 2014 : élection du président de l'**UMP**
- décembre 2015 : élections étudiantes à l'**I**nstitut **NA**tional des **L**angues et **C**ivilisations **O**rientales…

*** Législatives de 2017 : retour au vote papier pour les Français de l'étranger**

raisons : risque élevé de cyberattaques + suspicions sur les élections présidentielles américaines de **2016**

# A2 - Historique du vote électronique en Europe et aux USA

*** Machine à voter ou vote en ligne**

**machines à voter** utilisées dans de nombreux pays : manque de transparence => **problèmes**

*** Contre la machine à voter ou le vote en ligne**

**. Royaume-Uni**

vote en ligne testé lors d'élections locales de 2002 à 2007, manque de confiance => abandon

**. Irlande**

manque de fiabilité des machines à voter => abandon en 2004

**. Allemagne**

machines à voter déclarées illégales (exactitude non vérifiable par le citoyen)

**. Pays-Bas**

rupture de la confiance dans la fiabilité des résultats => abandon du vote en ligne en 2008

*** Pour la machine à voter ou le vote en ligne**

**. Suisse**

poursuite du vote en ligne depuis 2014

**. Estonie**

- choix possible du vote à l'urne avec poursuite du vote en ligne depuis 2013 (21 % des électeurs)
- services électroniques pour les citoyens : carte d'identité, impôts, police, services de santé, école

**. USA**

machines à voter très courantes : bulletins papier scannés, ou interaction directe sur écran tactile

=> . manque de transparence

. **représentativité des votes ?** (**constitution : 1787 !** élections présidentielles **2016**)

# A2 - Historique du vote électronique sur d'autres continents

## * Amérique latine

### - Venezuela

. depuis 2004, compagnie britannique ***Smartmatic*** : **SAES** (**S**martmatic **A**utomated **E**lection **S**ystem)

. dans l'isoloir, **machine à voter**, émettant un justificatif de vote à mettre dans une urne en carton

=> vérifications :

- la machine a bien enregistré le vote de l'électeur sans erreur sur le nom du candidat choisi
- les résultats correspondent bien à ceux donnés par les machines à voter

(dépouillement du contenu des urnes en carton lors d'audits postérieurs à l'élection)

- **mais pour la firme, le vote de 2017 d'élection de l'Assemblée Constituante a été manipulé**

(participation annoncée par les autorités supérieure de 1 million de votes / participation mesurée)

## * Australie

. vote électronique souhaité (grandes distances) et partiellement employé

. 2011 : une partie des votes a été mal enregistrée

. 2015 : faille sécuritaire trouvée dès le début de l'élection

(possibilité de modifier le vote d'1 électeur avant son recensement par la commission électorale)

# A3 - Perspectives de progrès logiciels

*** Enjeux et effectifs importants => réserve relative sur le vote électronique**

- on ne dispose pas encore de façon sûre de toutes les propriétés nécessaires (cf plus loin)
  au fonctionnement parfait nécessaire du vote électronique si **grandes élections politiques**
  car ***résultats totalement précis nécessaires***

- mais fonctionnement très suffisant pour des **consultations à toute échelle**
  car ***précision suffisante pour une possible prise de décision,***
  si marge de quelques % venant de quelques fraudes ou manipulations informatiques
  (analogie avec les techniques de marketing pour tester le choix du public entre des produits) :
  . **sondage** sur un gros échantillon significatif (milliers à centaines de milliers de personnes)
  . **référendum** (avec des millions de personnes)

*** Gros progrès réalisés avec des logiciels très avancés**

Logiciels réalisés pour s'approcher des propriétés nécessaires mettent en jeu la ***cryptographie homomorphe*** (cf plus loin) :

- **HELIOS** : Université de Harvard aux USA, Université de Louvain en Belgique
- **BELENIOS** variante de **HELIOS** : en France, à l'**INRIA**
  (**I**nstitut **N**ational de la **R**echerche en **I**nformatique et **A**utomatique)

# B – CRYPTOGRAPHIE ?

**B1 - Quelques définitions et notations**

**B2 - Cryptographie ancienne et contemporaine**

**B3 - Codage de CÉSAR**

# B1 - Cryptographie : quelques définitions et notations

*** Etymologie**

= "*écriture secrète*" en grec ancien

(*kruptos* : **κρυπτός** = "caché" - *graphein* : **γράφειν** = "écrire")

*** Entrée en jeu de branches des mathématiques et de l'informatique**

- **algèbre**
- **arithmétique** :
  . arithmétique des grands nombres premiers
  . traitement **informatique** des très grands entiers (problèmes de taille mémoire)
  . arithmétique binaire associée à la logique classique
  . **arithmétique modulaire** sur l'ensemble **Z/nZ** des **restes "modulo n"** : {**0**, **1**,…, **n-1**}
- **probabilités** :
  . calcul des probabilités
  . génération aléatoire **informatique** d'entiers

# B1 - Cryptographie : quelques définitions et notations

## * But

rendre incompréhensible le message via une fonction **C** , la ***clé de cryptage*** (*chiffrement*)

si on ne possède pas sa **fonction réciproque** $D = C^{-1}$ , la ***clé de décryptage*** (*déchiffrement*) ,

assurant ***confidentialité*** et ***vérifiabilité***

## * Fonction réciproque d'une fonction

- fonction **f** : **x** **entier** (ou **réel**) → **y = f (x) entier** (ou **réel**)

fonction **réciproque** $f^{-1}$ : **y** **entier** (ou **réel**) → **x =** $f^{-1}$ **(y)** tel que **y = f (x)**

(souvent nommée fonction **inverse**, mais rien à voir avec l'inverse multiplicatif…)

$f \circ f^{-1} = f^{-1} \circ f = \text{identité}$

- **exemples**

. **f** = carré : **x** réel → $y = x^2 \geq 0$

=> $f^{-1}$ = racine carrée : $y \geq 0 \rightarrow x = \sqrt{y}$

. **f** exponentiation à base **a** : **x** réel ⟶ $a^x > 0$ (noté $\exp_a(x)$)

=> $f^{-1}$ = logarithme à base **a** : $y > 0 \longrightarrow \log_a(x)$

# B2 - Cryptographie ancienne et contemporaine

*** Ancienne cryptographie traditionnelle**

La **clé de décryptage D = $C^{-1}$** doit se déduire aisément de la **clé de cryptage C** qui doit rester **secrète** : **problème crucial de transmission,** aggravé au sein d'un groupe ( **n** personnes => prévoir autant de clés que de couples (non ordonnés) : **n(n-1)/2** clés )

*** Cryptographie contemporaine**

- le cryptage introduit une **clé publique de cryptage C**, une **clé secrète de décryptage D = $C^{-1}$** (qui doit être impossible à déduire de **C** par un tiers), ce qui conduit à utiliser des **notions très avancées d'arithmétique**

- ainsi un problème mathématique souvent **très difficile** se pose :

*si* **C** *candidate à être fonction de* **cryptage**,

*existence et formulation de la fonction de* **décryptage D = $C^{-1}$** ?

# B2 - Cryptographie contemporaine : exemple très élémentaire

*** Juste pour voir, jamais utilisé dans la réalité !**

. message = **"À CE SOIR "**

. **codage** de chaque lettre en son rang alphabétique

=> message **codé** = **| 1 | 3 | 5 | 19 | 15 | 9 | 18 |**

. ***clé de cryptage*** **C** = ***carré*** : **x** → **C (x) = $x^2$**

. message **crypté** = **| 1 | 9 | 25 | 361 | 225 | 81 | 324 |**

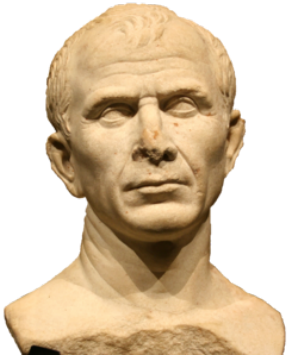
. ***clé de décryptage*** **D** = fonction ***réciproque*** **$C^{-1}$** de **C** = ***racine carrée*** : **y** → **D (y) =** $\sqrt{y}$

. => message **décrypté** = **| 1 | 3 | 5 | 19 | 15 | 9 | 18 |**, puis **décodé** : **"À CE SOIR "**

*** Problème : pas de secret, car *clé de décryptage* accessible à tous**

Il faudrait une ***clé de décryptage*** **secrète** et introuvable…

# B3 - Codage de Jules CÉSAR



(**Jules CESAR** : tête découverte à Arles dans le Rhône en 2007)

## * Cryptage par substitution monoalphabétique

- clé de **cryptage C** = **permutation circulaire** des lettres de l'alphabet :
  chaque lettre du message est décalée ***à droite*** dans l'alphabet d'une distance fixe **l**
  à transmettre au destinataire informé du codage

- dernières lettres => reprendre au début
  exemple : si **l = 5 , x → c**

## * Décryptage

**clé de décryptage D = $C^{-1}$** déduite de la **clé de cryptage C** par décalage **inverse : *à gauche***

# B3 - Exemple de codage de CÉSAR

Décalage à droite de **l = 13** lettres (*dans le message* : **d** → **q**) :

**message = v e n i v i d i v i c i**

(*"Je suis venu, j'ai vu, j'ai vaincu"*)

**=> codage = i r a v i v q v i v p v**


**=> décodage en faisant tourner la roue CÉSAR**

# B3 - Sécurité du cryptage par substitution monoalphabétique ?

*** Codage de CÉSAR => aucune sécurité de communication**

**permutation circulaire** :

essai successif des **26** clés possibles => **déchiffrage !**

*** Sécurité beaucoup plus grande pour une permutation quelconque**

. **$26 ! \approx 4 . 10^{26}$** clés à essayer !

. code "*cassé*" si **analyse de fréquences** :

langue naturelle => lettres plus fréquentes, ainsi en français **E** est la plus fréquente

# C - CRYPTAGE SYMÉTRIQUE

**C1 - Principe du cryptage symétrique**

**C2 - L'addition modulo 2 ⊕ (ou opération logique XOR)**

**C3 - Unités de mesure de quantité d'information numérique**

**C4 - Codage préalable de message en suite de bits**

**C5 - Le masque jetable de VERNAM**

**C6 - Aperçu sur *Enigma* la machine portable *nazie* de cryptage**

# C1 - Principe du cryptage symétrique

## * Formes

. forme ancienne : cryptage déjà connu des Égyptiens en **2000 av. J.-C.**

. formes modernes les plus connues :

- le **masque jetable** de **VERNAM** en **1918**
- **Enigma**, la **machine nazie** électromécanique portable (invention allemande de **1923**)
- l'algorithme **AES** (**A**dvanced **E**ncryption **S**tandard) en **2000**

## * Principe

cryptage et décryptage des messages à l'aide d'**une seule clé secrète γ** :

**C** est une fonction ***involutive*** , c.a.d. égale à sa fonction réciproque **D** = $C^{-1}$

## * Utilisation

. La plupart des cryptages symétriques utilisent un certain nombre **d'opérateurs logiques XOR** (voir plus loin)

. cryptage **rapide à l'exécution** :

=> utilisé pour la **transmission de données en masse**

# C2 - L'addition modulo 2 : ⊕

## * Ensemble des restes "modulo 2"

division d'un entier **a** par **2** => reste = **0** si **a** pair**, = 1** si **a** impair

**0** représente les nombres **pairs** : ensemble des nombres de reste **0**

**1** représente les nombres **impairs** : ensemble des nombres de reste **1**

## * Addition "modulo 2" ⊕

elle se définit naturellement sur l'ensemble **Z / 2Z** = {**0** , **1**} des restes "**modulo 2**" :

**0** ⊕ **0** = **0** (*pair* + *pair* = *pair*), **1** ⊕ **0** = **1** (*impair* + *pair* = *impair*) => **x** ⊕ **0** = **x**

**0** ⊕ **0** = **0** (*pair* + *pair* = *pair*), **1** ⊕ **1** = **0** (*impair* + *impair* = *pair*) => **x** ⊕ **x** = **0**

**0** ⊕ **1** = **1** ⊕ **0** = **1** (*pair* + *impair* = *impair* + *pair* = *impair*)

**associativité** (comme l'addition usuelle) : (**x** ⊕ **y**) ⊕ **z** = **x** ⊕ (**y** ⊕ **z**)

## * Table de ⊕

| ⊕ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# C2 - Exercice : non distributivité de l'addition modulo 2

**⊕ n'est pas distributif**

on va le voir en calculant $(x \oplus y) \oplus y$ :

. $(x \oplus y) \oplus y = x \oplus (y \oplus y) = x \oplus 0 = x$

. si c'était distributif :

$(x \oplus y) \oplus y = (x \oplus y) \oplus (y \oplus y) = (x \oplus y) \oplus 0 = x \oplus y$

# C2 - L'addition modulo 2 ⊕ s'identifie à l'opération logique XOR

*** Le XOR en logique classique**

une variable **x** de la logique classique peut prendre 2 valeurs : **VRAI**, **FAUX**

le **OU exclusif** (**XOR** : e**X**clusive **OR**), ou **somme disjonctive**, **disjonction exclusive**

défini par : **x XOR y = VRAI** si et seulement si **x** et **y** ont 2 valeurs différentes

(soit **x** est **VRAI**, soit **y** est **VRAI**)

*** Table de XOR = Table de ⊕**

| XOR | V | F |
|---|---|---|
| V | F | V |
| F | V | F |

| ⊕ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# C3 - Unités de mesure de quantité d'information numérique

## * Codage de base sur 8 bits (octet)

□ □ □ □ □ □ □ □

**8 bits** => $2^8 = 256$ nombres de **0** à **255** représentant tous un symbole

(cas du code **ASCII** en informatique : lettres, majuscules, chiffres, ponctuation…)

## * Multiples d'octets

quantité de bits (valeur 0 ou 1) contenus dans un support de stockage =>

unités de mesure standardisées par l'**I**nternational **E**lectrotechnical **C**ommission en **1998** :

1 **kilooctet** (ko) = $10^3$ octets - 1 **mégaoctet** (Mo) = $10^3$ Ko = $10^6$ octets

1 **gigaoctet** (Go) = $10^3$ Mo = $10^9$ octets - 1 **téraoctet** (To) ) = $10^3$ Go = $10^{12}$ octets

(1 **pétaoctet**(Po) = $10^3$ To, 1 **exaoctet**(Eo) = $10^3$ Po, 1 **zettaoctet**(Zo) = $10^3$ Eo, 1 **yottaoctet**(Yo) = $10^3$Zo)

## * Capacités moyennes de divers supports de stockage

**mémoire** : 1 clé USB = **32 Go**, 1 DD = **1 To**

**son** : 1 mn de son MP3 = **1 Mo**, 1 CD = **700 Mo**

**image** : 1 mn de vidéo à 720 pixels de résolution = **60 Mo**, 1 DVD = **5 Go**

**envoi de fichiers depuis un site d'Email** : espace mémoire alloué par fournisseur = **20 Mo**

# C4 - Codage préalable de message en suite de bits

## * Numération binaire

- nombre en numération décimale : $326 = 3*10^2 + 2*10^1 + 6*10^0$
- nombre en numération binaire : $26 = 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 11010$

## * Codage élémentaire sur 2 bits utilisé ici

numéro de lettre dans l'alphabet écrit en **système binaire** :

**A = 1 = $2^0$ → 1, B = 2 = $2^1$ → 10, C = 3 = $2^1 + 2^0$ → 11, … , Z = 26 → 11010**

=> message codé **m** = suite de **bits**

## * Exemple

message = "**C O M E** " :

**C = 3 → 11 , O = 15 → 1111 , M = 13 → 1101 , E = 5 → 101**

=> **m = 1 1 1 1 1 1 1 0 1 1 0 1**

# C4 – Codage en suite de bits : écriture d'un nombre n en système binaire

## * Division par 2

**n = 2q + r**

**q** quotient dans la division par **2** (en *informatique*, fonction : **div**)

**r** reste : **0 ≤ r < 2** (en *informatique*, fonction : **mod**)

## * Algorithme d'écriture en système binaire

**n = 5** = en binaire ?

**5 = 2 * 2 + 1 - 2 = 2 * 1 + 0 - 1 = 2 * 0 + 1 => 5 en binaire = 101**

**Stop**

**dès que q = 0**

## * Exercice : addition à 1 en binaire

**5 + 1 = 6** :

**1 0 1** (**5** en binaire)

\+ **0 0 1** (**6** en binaire)

———

**1 1 0** (retenue de 1 dans le **1** + **1**)

# C5 - Le masque jetable de VERNAM

* **Idées de VERNAM (ingénieur américain, 1918)**

- utiliser une fonction de cryptage **C** simple et **involutive** ($C = C^{-1}$) utilisant l'opération logique **XOR** et agissant **bit à bit**, unique pour chaque message, donc **jetable** après décryptage

- si le message codé est **m** avec **n bits** consécutifs $m_i$ , fabriquer une clé secrète **c** de **n bits** $c_i$ **générés aléatoirement** et à l'émission réception synchronisée (=> *difficultés techniques*) entre expéditeur et destinataire

- appliquer à **m** la fonction de cryptage **C** définie par l'action de $\oplus$ **bit à bit** :

$$C(m_i) = m_i \oplus c_i = m'_i$$

* **Explications un peu plus loin**

- pourquoi **C** est-elle **involutive** ($C = C^{-1}$), c.a.d. **cryptage symétrique** ?

- pourquoi **masque jetable** ?

# C5 - Exemple de fonctionnement du masque jetable

*** Éléments du cryptage**

message : "**COME**" → **m =** **1 1 1 1 1 1 1 0 1 1 0 1**

clé secrète aléatoire : **c =** **1 0 1 0 1 1 1 0 0 0 1 1 0**

*** Cryptage bit à bit (pas de retenue)**

$m'_i = m_i \oplus c_i$ =>

**1 1 1 1 1 1 1 0 1 1 0 1**

$\oplus$ **1 0 1 0 1 1 1 0 0 0 1 1 0**

---

**m' = 0 1 0 1 0 0 0 1 0 1 0 1 1**

# C5 - Décryptage avec la même clé, à jeter ensuite

*** L'addition modulo 2 rend le cryptage symétrique (involutif : $C = C^{-1}$)**

- **résultat utile (*)** : $(a \oplus x) \oplus x = a \oplus (x \oplus x) = a \oplus \underline{0} = a$

  (propriétés rencontrées de $\oplus$ : **associativité**, $x \oplus x = \underline{0}$ , $x \oplus \underline{0} = x$ )

- cryptage **bit à bit** : $C(m_i) = m_i \oplus c_i = m'_i$

  décryptage **bit à bit** : $C(m'_i) = m'_i \oplus c_i = (m_i \oplus c_i) \oplus c_i = m_i$ d'après (*)

*** Déchiffrement bit à bit sur l'exemple**

$$
\begin{array}{rl}
m' \oplus c = & 0\,1 \quad 0\,1\,0\,0 \quad 0\,1\,0\,1 \quad 0\,1\,1 \\
\oplus & 1\,0 \quad 1\,0\,1\,1 \quad 1\,0\,0\,0 \quad 1\,1\,0 \\
\hline
 & 1\,1 \quad 1\,1\,1\,1 \quad 1\,1\,0\,1 \quad 1\,0\,1 \quad = m
\end{array}
$$

*** Masque jetable (*one time pad*)**

après décodage, **clé jetée** :

**nouvelle clé** pour chaque **nouveau message**, de même longueur que celui-ci

# C5 - Le système de VERNAM seul cryptage théoriquement sûr

*** Bits modifiés par cryptage**

**preuve probabiliste** simple => en moyenne, **1 bit sur 2** n'est pas modifié

*** Caractère théoriquement sûr du système de VERNAM**

- seul cryptage à être **théoriquement sûr** : si la clé, de même longueur que le message et utilisée une seule fois pour crypter, est **totalement aléatoire**

- **preuve probabiliste** (*toutes les clés - suites de bits aléatoires - sont possibles*) utilisant :

. la **théorie de l'information** (**SHANNON**, **1948**), concept de physique mathématique

. l'**entropie** de **SHANNON**, concept thermodynamique mesurant la quantité d'information issue d'une source (***entropie*** *: mesure du degré de désordre d'un système microscopique*)

*** Inconvénient majeur**

préalable à toute communication : échanger une clé très longue, et ne jamais la réutiliser

*** Utilisation toujours actuelle**

utilisé durant la "guerre froide" : codage du téléphone rouge reliant Moscou et Washington

# C6 - Aperçu sur *Enigma* la machine portable *nazie* de cryptage

**1. Machine électromécanique avec substitution polyalphabétique**

**2. Le brouilleur pièce maîtresse d'*Enigma***

**3. Nombre total de clés possibles**

**4. Vues sur la machine *Enigma***

# C6
# 1. Machine électromécanique avec substitution polyalphabétique

## * Substitution polyalphabétique

. **chaque lettre frappée sur clavier donne lieu à un décalage différent**

. ***Enigma*** : machine à chiffrer électromécanique portable (invention allemande : **1923**)
**1930** à **1945** : plus de 30 000 telles machines dans l'armée allemande
système cryptographique alors le plus sûr au monde, codant les **messages nazis**

## * Fonctionnement complexe avec 4 éléments reliés par circuit électrique

- **clavier** pour entrer le texte clair

- **brouilleur** du texte clair avec des **rotors**

- **réflecteur**
. lettre **minuscule** en cours de chiffrage => renvoyée par courant à travers rotors et connexions jusqu'au **tableau lumineux**

- **tableau lumineux**
. la lettre cryptée s'y affiche en **majuscule**
. cryptage **symétrique** ($C = C^{-1}$) : si **b** tapée, **C**(**b**) = **M** affichée – si **m** tapée, **C**(**m**) = **B** affichée

# C6
# 2. Le brouilleur pièce maîtresse d'*Enigma*

## * Brouilleur du texte clair composé de 3 rotors

. 3 **rotors cylindriques** mobiles interchangeables, avec chacun **26** positions (lettres) différentes

. lettre frappée sur **clavier** => courant électrique dans le **rotor** suivant le câblage interne

. le **1er** rotor **tourne d'un cran** (=> change les connexions électriques) à chaque lettre frappée, revient au début après **26** frappes ; alors le **2e** rotor **tourne d'un cran** ; le **1er** rotor refait un tour complet ; le **2e** rotor **tourne d'un cran** à nouveau, etc…
**=> si lettre à coder plusieurs fois, ce n'est jamais pareil**

## * Tableau de connexions

. tableau de **connexions à 6 fiches** (simili-prises jack) entre le clavier et le 1er rotor
=> interversion possible de 6 fois 2 lettres reliées 2 à 2 avant l'arrivée au rotor de la lettre tapée (***10*** *paires de lettres permutées* ***chaque jour*** *par les nazis*)

. chaque jour définition d'1 nouvelle clef, avec un **ensemble de spécifications**
(*ordre de disposition des rotors, orientation des rotors, branchement des connexions*)

## * Emission - réception

- texte **crypté** transmis via un opérateur radio
- **décryptage** par le destinataire à l'aide d'une machine ***Enigma*** similaire et de l'**ensemble de spécifications** indiquant les positions du jour

# C6
# 3. Exercice : calcul du nombre total de clés possibles

*** Calcul du nombre de combinaisons possibles : produit de 3 nombres a, b, c**

**- orientation des rotors**

3 rotors de 26 lettres chacun => 26 positions pour chaque rotor => **a** = $26^3$ **= 17576**

**- ordre des 3 rotors**

3 rotors (1, 2, 3) placés selon 6 permutations possibles : 123, 132, 213, 231, 312, 321=> **b = 6**

**- nombre de branchements possibles**

le tableau de connexions à fiches, placé avant le 1er rotor, apparie 6 fois 2 lettres parmi 26 en utilisant 6 câbles avant l'entrée de lettres dans le 1er rotor

$\mathbf{C_n^2 = n(n-1)/2}$ = nombre des combinaisons de **n** objets **2** à **2** => nombre de connexions :

$C_{26}^2.C_{24}^2.C_{22}^2.C_{20}^2.C_{18}^2.C_{16}^2$ = (13.12.11.10.9.8) . (25.23.21.19.17.15) = 72 282 089 880 000

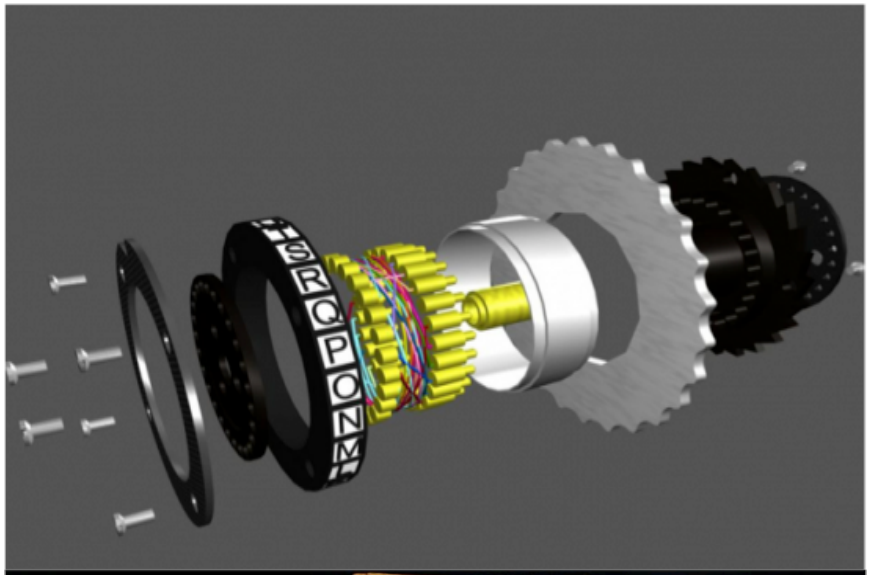
ordre des 6 câbles sans importance => diviser par 6 ! = **720** (nombre de permutations possibles)

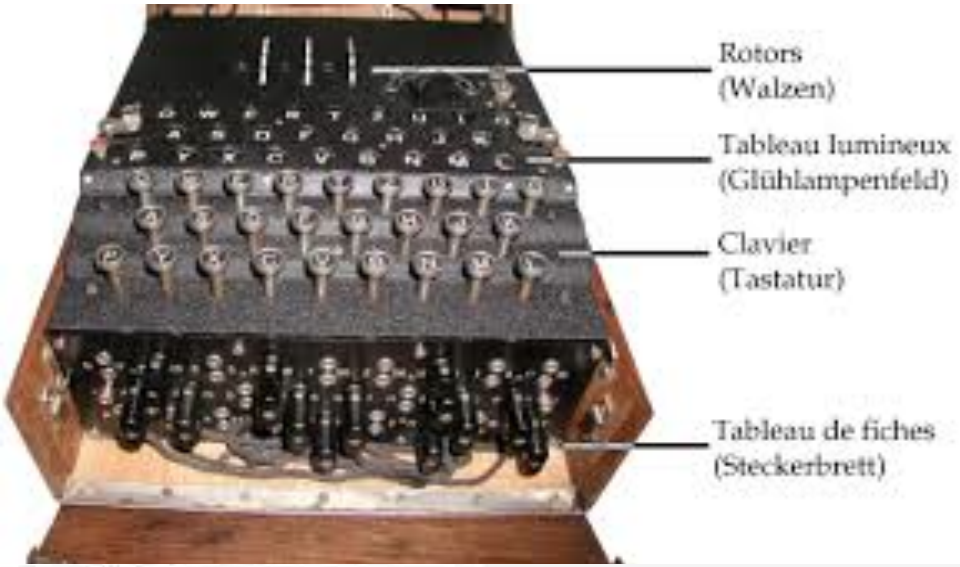
=> **c = 100 391 791 500**

*** Nombre de combinaisons possibles voisin de 10 millions de milliards**

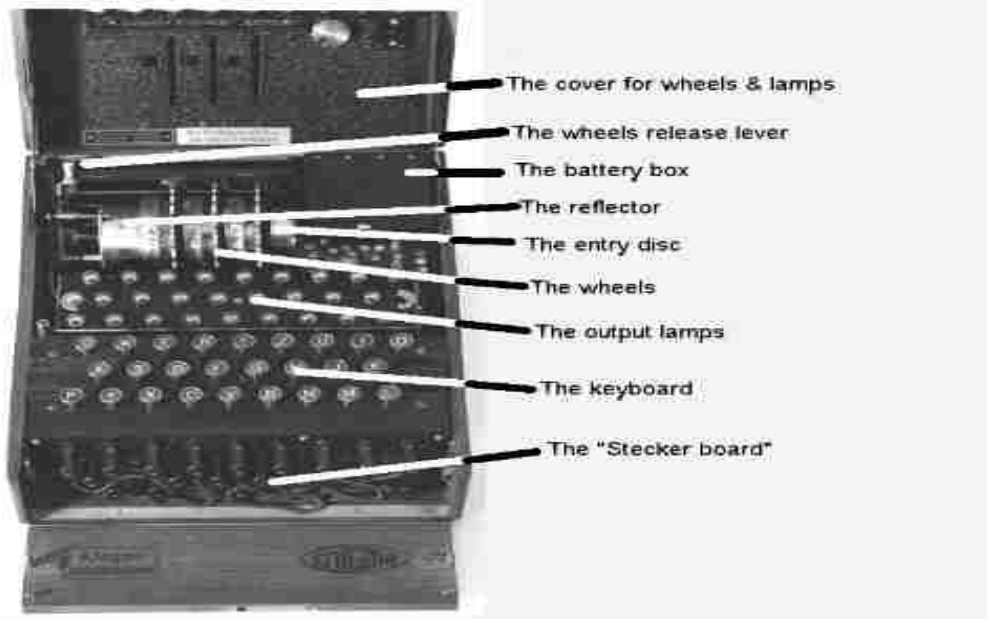
en effet, c'est donc : **a . b . c = 17576 . 6 . 100 391 791 500 = 1,06 . $10^{16}$**

# C6
# 4. Vues sur la machine *Enigma*







# D - Cryptage asymétrique : l'exemple de RSA

**D1. Principe du cryptage asymétrique**

**D2. Le contexte d'arithmétique avancée autour de RSA**

**D3. Fabrication de clé secrète et clé publique personnelles dans RSA**

**D4. Fonctionnement de RSA sans authentification**

**D5. Fonctionnement de RSA avec signature électronique**

# D1. Principe du cryptage asymétrique

*** Formes**

forme ancienne : codage de **Jules CÉSAR** dans l'*Antiquité*

forme moderne la plus connue : l'algorithme **RSA** (**R**ivest **S**hamir **A**dleman) en **1978**

*** Principe**

- une clé **publique personnelle** **C** de **cryptage**
- la clé **secrète personnelle** de **décryptage**, fonction réciproque $C^{-1}$ de **C**

*** Utilisation**

cryptage **lent à l'exécution** :

l'algorithme plus complexe donne lieu à une plus grande charge de calcul

=> souvent utilisé pour l'**échange de clés secrètes**

# D2. Le contexte d'arithmétique avancée autour de RSA

*** Arithmétique modulaire dans Z/bZ ensemble {0, 1,…, b-1} des restes modulo b**

- division entière (*euclidienne*) de **a** par **b** : **a = b q + r , 0 ≤ r < b**

  **a** = ensemble des entiers de même reste que **a** dans cette division par **b** (: reste modulo **b**)

- **Z/bZ** muni naturellement :

  . d'une **addition** + avec **0** comme élément neutre (**a** + **0** = **a**), du fait de **a + b** = **a** + **b**

  exemple : **Z/2Z = {0 , 1}** = {pair , impair} => **0** + **0** = **0**, **0** + **1** = **1**, **1** + **1** = **0**

  . d'une **multiplication** * avec **1** comme élément neutre (**a** * **1** = **a**), du fait de **a * b** = **a** * **b**

  exemple : **Z/2Z = {0 , 1}** = {pair , impair} => **0** * **0** = **0**, **0** * **1** = **0**, **1** * **1** = **1**

*** Exemples de tables de multiplication dans Z/bZ**

dans **Z/2Z** :

| * | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

dans **Z/4Z** :

| * | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 0 | 2 | 0 | 2 |
| 3 | 0 | 3 | 2 | 1 |

# D2. Le contexte d'arithmétique avancée autour de RSA

*** Entier premier avec un autre entier**

**e** entier est **premier avec b** entier ssi **1** est le seul diviseur commun à **e** et **b**

exemple : **e** = **15**, **b** = **14**

*** L'inverse f de e ≠ 0 n'existe pas toujours dans Z/bZ**

dans **Z/4Z** , table de multiplication des éléments **non nuls** :

| * | 1 | 2 | 3 |
|---|---|---|---|
| 1 | **1** | 2 | 3 |
| 2 | 2 | 0 | 2 |
| 3 | 3 | 2 | **1** |

2 | 2 0 2 => **2** non premier avec **4** n'admet pas d'inverse

3 | 3 2 **1** => **3** premier avec **4** admet un inverse

*** Existence d'un inverse**

- **Théorème** : **e ≠ 0** inversible dans **Z/bZ ssi e premier avec b**

(donc **b** premier => tout **e ≠ 0** est inversible dans **Z/bZ**)

- N.B. **e** premier avec **a** , **e** premier avec **b ≠ a** => **e** premier avec **a * b**

***Détermination de l'inverse f de e modulo b quand e premier avec b**

- **algorithme d'EUCLIDE du PGCD** de **e** et **b**
- calcul associé de **coefficients de BEZOUT**

# D2. Le contexte d'arithmétique avancée autour de RSA

## * Petit théorème de FERMAT (1640) et exponentielle modulaire

- Théorème

**m** premier => si **x** premier avec **m** (i.e. non multiple de **m**), alors : $x^{m-1} = 1$ modulo **m**

- Conséquence

**p** et **q** **nombres premiers** différents, **b = (p-1) (q-1), n = p q** , **k = 1** modulo **b**

=> pour tout entier **x** : $x^k = x$ modulo **n**

## * Application : existence d'une fonction réciproque d'exponentielle modulaire

Théorème

**n = p q**, **p ≠ q** nombres premiers, **e** premier avec **p-1** et **q-1** ( donc avec **b = (p-1) (q-1)** ) =>

- **e** admet un inverse **f** dans **Z / bZ**
- $y = x^e$ modulo **n** => $x = y^f$ modulo **n**

# D3. Fabrication de clé secrète et clé publique personnelles dans RSA

*** Clé secrète d'Alice pour décrypter un message reçu**

**Alice** choisit **p ≠ q** très grands **nombres premiers**

=> clé **secrète** de **décryptage** : $D_A = \{p, q\}$

*** Clé publique d'Alice à consulter par tout émetteur de message pour le crypter**

**n = p q**, **b = (p-1)(q-1)** => très grands nombres

**Alice** détermine **e premier avec b** :

1. algorithme probabiliste => obtention rapide d'un candidat **e** presque sûrement premier avec **b**
2. => algorithme d'**EUCLIDE** du PGCD : PGCD (**e**, **b**) = 1 ? (sinon à nouveau 1.)
   => clé **publique** d'Alice : $C_A = \{n, e\}$

*** Détermination de l'inverse f de e modulo b nécessaire au décryptage**

- **e premier avec b** => l'inverse **f** de **e** (c.a.d. **e f = 1**) **modulo b** existe
- détermination :
  . algorithme d'**EUCLIDE** du **PGCD** de **e** et **b**
  . calcul associé des coefficients de **BEZOUT**

# D4. Fonctionnement de RSA sans authentification

*** Cryptage par Alice d'un message à envoyer à Bob**

- **Alice** consulte la clé **publique** de **Bob** : $K_B = \{n, e\}$
  (**p ≠ q** très grands **nombres premiers**, **n = p q**, **b = (p-1)(q-1)**, **e** premier avec **b**)
- message codé en séquence d'éléments **m** de l'ensemble {**0**, **1**,…, **n-1**} des **restes modulo n**
- algorithme de **cryptage** par **Alice** : $C_{K_B}(m) = m' = m^e$ **modulo n**, envoyé à **Bob**

*** Décryptage par Bob du message envoyé par Alice**

- **Bob** détermine l'**inverse f** de **e** (**e * f = 1**) **modulo b** , d'où sa clé **secrète {n, f}**
- algorithme de **décryptage** par **Bob** : $D_{K_B}(m') = m'^f = (m^e)^f = m^{ef} = m$ modulo **n**

*** Sécurité du cryptage RSA**

basée sur le temps infiniment grand (**exponentiel, dépassant celui de la vie humaine** )

pour décomposer un très grand nombre en facteurs premiers

*** Authentification du message reçu**

***rien ne garantit à Bob que c'est Alice le bon expéditeur !***

# D4. Fonctionnement de RSA sans authentification : exemple rudimentaire

## * Prétraitement du message

par exemple, alphabet décimal à **2** chiffres :

**A** = "**01**", **B** = "**02**",.., **Z** = "**26** ; **0** = "**30**", **1** = "**31**",.., **9** = "**39**" ; □ = "**40**"

message à envoyer par **Alice** à **Bob** : numéro de plan à exécuter = **9** => **m = 39**

## * Codage par Alice

$C_B$ = {**n = 187, e = 7**}, **n = pq**, **p = 17** et **q = 11** premiers, **e** premier avec **16 = (p-1)** et **10 = (q-1)**

**m' = $C_B$ (m) = $m^e$** modulo **n**, reçu par **Bob** : **m' = $m^7$** modulo **187 = $39^7$** modulo **187 = 129**

## * Détails du calcul de m'

**7 = $2^2 + 2 + 1$** => **$39^7 = 39 . 39^2 . 39^{2.2}$**

**$39^2 = 1521$** = 8 . 187 + 25 = **25** modulo **187**

=> **$39^{2.2} = 25^2$** modulo **187** ; **$25^2$** = 625 = 3 . 187 + 64 = **64** modulo **187**

=> **$39^7$ = 39 .25. 64** modulo **187 = 975 . 64** modulo **187 = 40 . 64** modulo **187** (975 = 5.187 + 40)

= **2560** modulo **187 = 129** (2560 = 13 . 187 + 129)

# D5. Fonctionnement de RSA avec signature électronique

*** Cryptage par Alice d'un message m à envoyer à Bob**

- **Alice** consulte sa **clé secrète** $D_A$
- elle code **m** en $D_A(m)$ (***signature électronique***)
- elle consulte la **clé publique** $C_B$ de **Bob**
- elle code $D_A(m)$ en $m' = C_B\,(D_A(m))$, reçu par **Bob**

*** Décryptage par Bob du message envoyé par Alice**

- **Bob** consulte sa **clé secrète** $D_B$
- il l'applique au message reçu $m' = C_B\,(D_A(m))\,D_B \Rightarrow D_B\,(C_B\,(D_A(m))) = D_A(m)$
  ($D_B \circ C_B$ = **identité** : fonction réciproque)
- il applique ensuite la **clé publique** $C_A$ d'**Alice** => $C_A\,(D_A(m)) = m$
  ($C_A \circ D_A$ = **identité** : fonction réciproque)
- ***cette fois, il est ainsi garanti à Bob que c'est Alice le bon expéditeur :***
  ***elle est la seule à avoir pu utiliser sa clé secrète !***

# E - Cryptage homomorphe

**E1 – Motivation initiale du cryptage homomorphe :**
**confidentialité de données traitées par un fournisseur**

**E2 - Cryptage homomorphe : définitions**

**E3 - Réalisation de tout algorithme avec XOR et AND et de tout calcul avec l'addition et la multiplication**

**E4 - Cryptage homomorphe : historique**

**E5 - Cryptage homomorphe et vote électronique**

# E1 - Motivation initiale du cryptage homomorphe : confidentialité de données traitées par un fournisseur

**Faire opérer des traitements par un service extérieur sur des données cryptées ?**

- confier des données pour conservation et traitements à la demande à un service de sauvegarde extérieur (cas du "***cloud computing***" : "*informatique en nuage*")

- un **problème essentiel** : préservation de la **confidentialité** entre client et fournisseur **?**

**réponse** : **cryptage** des données en clair numérisées, s'y appliquent **additions** et **multiplications**

- et si le client veut faire des recherches sur les données confiées (*moyennes, nombre de données dépassant un seuil, tri de données, fréquences…*) **?**

**réponse** : impossible directement sur les données cryptées, sauf si les algorithmes de traitement de données passent à travers la couche de cryptage (**homomorphie**) => le service pourra traiter des informations illisibles pour lui, et transmettre les résultats cryptés au client, le seul à pouvoir les **décrypter** avec sa **clé secrète,** qui doit être quasiment **impossible à découvrir**

# E1 - Motivation initiale du cryptage homomorphe : confidentialité de données traitées par un fournisseur

*** Progrès en informatique mathématique : cryptage homomorphe**

**- homomorphie : crypté du composé par une opération sur des données en clair numérisées égal au composé par une opération analogue des cryptés de ces données**

- méthode de ***cryptage asymétrique*** permettant de

. masquer toute l'information

. autoriser **additions** et **multiplications** (***homomorphie complète*** : cf plus loin)

*** Remarque : homomorphismes de fonctions en maths**

- élévation à une puissance (**x** réel⟶ $\mathbf{x^p}$) : $\mathbf{x^{p+q} = x^p . X^q}$

- exponentiation à base **a** (**x** réel ⟶ $\mathbf{a^x}$, noté $\mathbf{exp_a(x)}$) : $\mathbf{exp_a(x+y) = exp_a(x) . exp_a(y)}$

- logarithme à base **a** (**x** réel positif ⟶ $\mathbf{log_a(x)}$) :

. fonction réciproque de l'exponentiation : $\mathbf{exp_a\ (log_a(x)) = x}$

. $\mathbf{log_a\ (x.y) = log_a(x)\ +\ log_a(y)}$

# E2 - Cryptage homomorphe : définitions

## * Homomorphies additive et multiplicative

- crypté du composé par opération (addition, multiplication) sur des données en clair numérisées
  = composé par une opération analogue * des cryptés de ces données
  => l'algorithme de **cryptage** $C_K$ de clé **K** permet les opérations
  aussi bien sur les données en clair que sur les données cryptées

- **x** et **y** données en clair numérisées :

***homomorphie additive*** **(ha)** : $C_K(x + y) = C_K(x) * C_K(y)$

***homomorphie multiplicative*** **(hm)** : $C_K(x \cdot y) = C_K(x) * C_K(y)$

## * Cryptage complètement homomorphe

cryptage ***complètement homomorphe*** **(FHE** : **F**ully **H**omomorphic **E**ncryption) :
propriétés **(ha)** et **(hm)** vérifiées simultanément

# E2 - Exercice : exemple élémentaire de non homomorphie

*** Cryptage de m par ⊕ bit à bit avec clé c de même longueur que m**
**(*masque jetable de VERNAM*)**

Rappels :

. cryptage non asymétrique : non candidat au cryptage homomorphe

. **cryptage** : **m** de bits consécutifs $\mathbf{m_i} \longrightarrow \mathbf{C}\ \mathbf{(m)}$ de bits consécutifs $\mathbf{m'_i = m_i \oplus c_i}$

*** Non homomorphie additive**

. écriture binaire : **c = 11** , **m = 10**, **m' = 01**

. **cryptage du composé** : $\mathbf{m \oplus m' = C(m \oplus m') = C\ (11) = 11 \oplus c = 11 \oplus 11 = 00}$

. tandis que pour le **composé des cryptés** :

$\mathbf{C\ (m) = 10 \oplus 11 = 01}$ , $\mathbf{C\ (m') = 01 \oplus 11 = 10 \Rightarrow C\ (m) \oplus C\ (m') = 11 \neq 00}$

# E3 - Réalisation de tout algorithme avec XOR et AND et de tout calcul avec l'addition et la multiplication

**1. Une fonction logique s'écrit toujours avec OR, AND, NOT**

**2. Réalisation de tout algorithme avec XOR et AND**

**3. Réalisation de tout calcul avec l'addition et la multiplication**

# E3

## 1. Une fonction logique s'écrit toujours avec OR, AND, NOT

**Toute fonction logique - en jeu dans tout circuit électronique (ou "*booléen*" ) – peut s'exprimer avec OR, AND, et NOT**

- **F = FAUX , V = VRAI**

- **NOT** (**NON x**, noté ¬ **x**, nommé **négation**)

. table de vérité :

| x | F | V |
|---|---|---|
| ¬ x | V | F |

- **OR** (= **OU**, noté ∨ , nommé **disjonction**)

. **x, y** ⟶ **x ∨ y** : **VRAI** si et seulement au moins l'un des deux est **VRAI**

. table de vérité :

| ∨ | F | V |
|---|---|---|
| F | F | V |
| V | V | V |

- **AND** (= **ET, noté** ∧, nommé **conjonction**)

**x, y** ⟶ **x ∧ y** : **VRAI** si et seulement si (**x** est **VRAI**) et (**y** est **VRAI**) :

| ∧ | F | V |
|---|---|---|
| F | F | F |
| V | F | V |

# E3
# 2. Réalisation de tout algorithme avec XOR et AND

*** XOR (= OU exclusif, noté ⊕, nommé disjonction exclusive)**

. **x, y** ⟶ **x ⊕ y** : **VRAI** si et seulement si valeurs différentes de **x** et **y**

. table de vérité :

| ⊕ | F | V |
|---|---|---|
| F | F | V |
| V | V | F |

. on savait que : **x ⊕ F = x**, **x ⊕ x = F** - **on découvre** : **x ⊕ V = ¬ x** et **x ⊕ ¬ x = V**

*** Une fonction s'écrit toujours avec OR, AND, NOT et aussi avec XOR et AND**

en effet :

. **x ⊕ V = ¬ x** => **NOT x = x XOR V**

. **x OR y = (x XOR y) XOR (x AND y)**

(preuve technique de plus de 15 lignes de calcul propositionnel ! )

# E3

## 3. Réalisation de tout calcul avec l'addition et la multiplication

**XOR correspond à l'addition modulo 2, AND à la multiplication modulo 2 :**

*** (*Rappel*) Table de XOR = Table de ⊕**

| XOR | F | V |
|---|---|---|
| F | F | V |
| V | V | F |

| ⊕ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

*** Table de AND = Table de x**

| AND | F | V |
|---|---|---|
| F | F | F |
| V | F | V |

| x | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

# E4 - Cryptage homomorphe : historique

*** Cryptages additivement homomorphes ou multiplicativement homomorphes**

- **1978** : **RIVEST** et **ADLEMAN** => existence d'un cryptage **complètement homomorphe ???**

- avant **2009**, cryptages asymétriques ***additivement homomorphes*** :
  . **PAILLIER (1999)** : met en jeu des ***produits de très grands nombres premiers***

- avant **2009**, cryptages asymétriques ***multiplicativement homomorphes*** :
  . **RSA (1978)** : met en jeu des ***produits de très grands nombres premiers***
  . **EL GAMAL (1984)** : met en jeu des éléments d'arithmétique encore plus avancés (**logarithme discret**) que ceux de **RSA**

*** Recherches et progrès en *cryptographie complètement homomorphe* depuis 2009**

- **2009** : thèse de **Craig GENTRY** (Université de Stanford - USA)
  => obtention **théorique** du **premier cryptage complètement homomorphe** !
- nombreux travaux actuels d'*optimisation* pour arriver à sa **mise en pratique effective**

# E5 - Cryptage homomorphe et vote électronique

**1. Modélisation du vote : codage en *bits*, clés, cryptage, transmissions, décryptage**

**2. Cryptages homomorphes**

**3. Logiciel BELENIOS de vote électronique**

**4. Compléments : preuve de cryptage correct, signature, vérifiabilité**

# E5
# 1. Modélisation du vote : codage en *bits*, clés

## * Codage en bits des votes

- chaque votant $\mathbf{V_i}$ (**i** = **1** à **n**) exprime un choix de réponse **OUI** ou **NON** :
  . à un **référendum**
  . ou concernant un candidat $\mathbf{C_j}$ (**j** = **1** à **p**) à une **élection**
- son choix se traduit par un **bit**, **1** si **OUI**, **0** si **NON** :
  . noté $\mathbf{X_i}$ en cas de **référendum**
  . noté $\mathbf{X_{ij}}$ s'il concerne le candidat $\mathbf{C_j}$ à une **élection**

## * Clé publique C et clé secrète D

- **clé publique C** : à utiliser par chaque votant pour **crypter** son vote
- **clé secrète D** (**D ° C = identité** : **D** fonction réciproque de **C**) :
  couple **{C , D}** choisi par l'***opérateur de dépouillement***, qui communique **C** à chaque votant $\mathbf{V_i}$

# E5
# 1. Modélisation du vote : cryptage

## * Cryptage C additivement homomorphe

**- crypté du composé** par l'addition des données en clair

= **composé des cryptés** par une opération analogue – par ex. la multiplication *

**- traduction analytique :**

(***référendum***) $C(X_1 + \ldots + X_i + \ldots + X_n) = C(X_1) * \ldots * C(X_i) * \ldots * C(X_n) = P$

(***élection***) $C(X_{1j} + \ldots + X_{ij} + \ldots + X_{nj}) = C(X_{1j}) * \ldots * C(X_{ij}) * \ldots * C(X_{nj}) = P_j$

## * Cryptage des votants

$V_i$ **crypte** le bit de chacun de ses choix avec **C**

transmet chaque nombre crypté $C(X_i)$ ou $C(X_{ij})$ à l'***opérateur de calcul***

(qui ne connaît pas **D** et donc **ne peut découvrir les votes individuels**)

# E5
# 1. Modélisation du vote : transmissions, décryptage

*** L'opérateur de calcul**

- calcule le produit **P** ou les produits $\mathbf{P_j}$ des nombres cryptés reçus
- le(s) transmet à l'***opérateur de dépouillement***

*** L'opérateur de dépouillement**

- décrypte **P** ou les $\mathbf{P_j}$ grâce à la **clé secrète D**
  mais ne connaît pas les nombres cryptés transmis individuellement
  => **ne peut connaître les votes individuels** malgré sa possession de la clé de décryptage **D**
- **référendum** :

$$\mathbf{C(X_1 + \ldots + X_i + \ldots + X_n) = C(X_1) * \ldots * C(X_i) * \ldots * C(X_n)}$$

$$\mathbf{D\ (P) = D(C(X_1) * \ldots * C(X_i) * \ldots * C(X_n)) = D(C(X_1 + \ldots + X_i + \ldots + X_n))}$$

$$\mathbf{D \circ C = identité \Rightarrow D\ (P) = X_1 + \ldots + X_i + \ldots + X_n = \text{nombre de 1 (OUI)}}$$

- **élection** :

$$\mathbf{C(X_{1j} + \ldots + X_{ij} + \ldots + X_{nj}) = C(X_{1j}) * \ldots * C(X_{ij}) * \ldots * C(X_{nj})}$$

$$\mathbf{D\ (P_j) = D(C(X_{1j}) * \ldots * C(X_{ij}) * \ldots * C(X_{nj})) = D(C(X_{1j} + \ldots + X_{ij} + \ldots + X_{nj}))}$$

$$\mathbf{D \circ C = identité \Rightarrow D\ (P) = X_{1j} + \ldots + X_{ij} + \ldots + X_{nj} = \text{nombre de 1 (OUI)}} \text{ pour } \mathbf{C_j}$$

- diffuse le résultat du **référendum**, ou le nom du vainqueur avec un maximum de **OUI**

# E5

# 2. Cryptage multiplicativement homomorphe RSA (1978)

*** Cryptage par Alice d'un message à envoyer à Bob**

- **Alice** consulte la clé **publique** de **Bob** : $K_B = \{n, e\}$
  ($p \neq q$ très grands **nombres premiers**, $n = p\,q$, $b = (p-1)(q-1)$, **e** premier avec **b**)
- message codé en séquence d'éléments **m** de l'ensemble {**0**, **1**,…, **n-1**} des **restes modulo n**
- algorithme de **cryptage** par **Alice** : $C_{K_B}(m) = m' = m^e$ **modulo n**, envoyé à **Bob**

*** Décryptage par Bob du message envoyé par Alice**

- **Bob** détermine l'**inverse f** de **e** ($e * f = 1$) **modulo b** , d'où sa clé **secrète {n, f}**
- algorithme de **décryptage** par **Bob** : $D_{K_B}(m') = m'^f = (m^e)^f = m^{ef} = m$ modulo **n**

*** RSA algorithme de cryptage multiplicativement homomorphe**

$C_{K_B}(m_1) * C_{K_B}(m_2) = m_1^e * m_2^e = (m_1 * m_2)^e \pmod n = C_{K_B}(m_1 * m_2)$

N.B. : cette propriété est aussi une faiblesse de RSA lorsque ce système est mal utilisé

# E5

# 2. Cryptage multiplicativement homomorphe de EL GAMAL (1984)

## * Logarithme discret

. **a** nombre réel => **logarithme réel** = fonction réciproque de l'**exponentiation** : **x** réel $\longrightarrow$ $\mathbf{a^x}$

**a** élément de l'ensemble {0, 1,…, n-1} des restes modulo **n**

=> $\mathbf{G_a} = \{\mathbf{a^0}, \mathbf{a^1}, \ldots, \mathbf{a^{n-1}}\}$ groupe **cyclique** fini engendré par **a**

($\mathbf{a^n = a^0 = 1}$, $\mathbf{a^{n+1} = a}$, etc… => **cyclique**)

. **logarithme discret** : **b** dans $\mathbf{G_a}$ $\longrightarrow$ le plus petit entier **k** tel que $\mathbf{a^k = b}$

(= fonction réciproque de l'**exponentiation k** entier $\longrightarrow$ $\mathbf{a^k}$)

## * Cryptage C additivement homomorphe de EL GAMAL

- => arithmétique plus avancée que celle de **RSA**, basée sur le problème du **logarithme discret**
- utilisé par le logiciel **BELENIOS**

# E5
# 3. Logiciel BELENIOS de vote électronique

*** Développement de logiciels pour le vote électronique en cas de grandes élections**

- bon fonctionnement du vote électronique en cas de grandes élections politiques

=> réaliser un **logiciel très avancé** assurant :

. **confidentialité** via la **cryptographie homomorphe**

. **vérifiabilité**

- logiciel **HELIOS** : Université de Harvard aux USA, Université de Louvain en Belgique

*** Le logiciel BELENIOS**

- variante améliorant **HELIOS**, développé sous licence libre en France, à l'**INRIA** (**I**nstitut **N**ational de la **R**echerche en **I**nformatique et **A**utomatique) par divers chercheurs dont **Véronique CORTIER** (auteur de nombreux articles grand public sur le sujet)

- plateforme de vote : permet la **mise en place simple et gratuite d'une élection**

- protocole en **2 phases** comme pour le vote papier classique : **vote** et **dépouillement**

- utilise le cryptage additivement homomorphe **EL GAMAL**

# E5

## 4. Compléments : preuve de cryptage correct, signature, vérifiabilité

*** Preuve de cryptage correct**

- pour s'assurer que l'électeur a bien crypté **0** ou **1** et non une autre valeur :
technique de **preuve à divulgation nulle de connaissance** =>
**sûr** que le contenu d'un crypté vérifie une certaine propriété, sans rien indiquer sur le contenu

- => impossible pour un votant malhonnête de construire un bulletin valide
cryptant par exemple la valeur **100**, ce qui aurait permis d'ajouter **99** votes **"OUI"** !

*** Signature**

- Le vote crypté, accompagné de la preuve de validité, forme le **bulletin**
- l'électeur **signe** l'ensemble à l'aide d'un **jeton de vote** reçu par courrier ou mail :
=> impossibilité pour un électeur d'ajouter des bulletins sous d'autres noms !

*** Vérifiabilité**

- le bulletin est envoyé à une **urne**
- l'urne affiche sur une page Web publique tous les bulletins reçus :
chaque électeur peut **vérifier** que son bulletin est présent dans l'urne

# E5

## 4. Compléments : preuve à divulgation nulle de connaissance

### * Preuve à divulgation nulle de connaissance ?

protocole (spécification de plusieurs règles relatives à 1 type de communication) sécurisé :
. élément **fournisseur de preuve** mathématique d'1 proposition vraie
(exemple : *le contenu d'une information vérifie une certaine propriété*)
. ↦ élément **vérificateur** sans lui révéler d'autres informations
(dans l'exemple : *sans rien indiquer sur le contenu*)

### * Exemple

- 2 billes identiques de couleurs différentes : **rouge** et **vert**, indiscernables pour 1 **aveugle**

objectif : le convaincre que les couleurs sont différentes sans jamais qu'il sache laquelle est verte

- protocole :

. l'aveugle place les 2 billes dans son dos, en prend une et la montre
il la remet dans son dos et montre une des billes : la même (**Proba** : **1/2**), ou l'autre (**Proba** : **1/2**)
il demande : "*ai-je changé de bille* ? " => on lui répond d'après les 2 couleurs successives

- répétition du protocole autant de fois que nécessaire (jusqu'à un **changement de couleur**)
**Proba de ce changement** = *série géométrique* = **1/2 + 1/4 + 1/8 +… 1/2$^{n}$ +… = 1 / (1 - 1/2) = 1**
( = somme des probas d'événements favorables, ainsi **1/4 =** Proba {*inchangé*} . Proba {*changé*})

# F - Discussion autour du vote électronique et du logiciel BELENIOS

## F1. Besoins spécifiques du vote électronique

## F2. Avantages de BELENIOS

## F3. Inconvénients de BELENIOS

# F1. Besoins spécifiques du vote en ligne

## - Immunité -

**- contre la destruction, qui peut fausser un possible recompte des voix :**

. les données rassemblées sur plusieurs ordinateurs peuvent être facilement détruites

. l'outillage nécessaire au stockage peut subir des dommages temporaires ou définitifs

. le logiciel peut dysfonctionner du fait de ***bugs***, ou d'attaque de **virus** (***malware***)

**- contre les logiciels malveillants**

. ils peuvent enregistrer, changer, divulguer des votes brisant l'anonymat

(**keylogger** - *enregistreur de frappe* : en dissimulant son activité, espionne un ordinateur avec enregistrement de touches saisies, captures d'écran, listage d'actions de l'utilisateur…)

**- contre la coercition**

. un espion informatique pourrait influencer un électeur ou l'obliger à voter autrement

. un tiers ne doit pas savoir l'intention d'abstention ou vote d'1 électeur, sinon influence possible

(**vote obligatoire** - avec parfois **vote blanc** - dans certains pays comme :
Belgique, Danemark, Grèce, Turquie, Egypte, Liban, Australie, pays d'Amérique Latine…)

# F1. Besoins spécifiques du vote en ligne

## * Vérifiabilité

**- dépouillement tenant compte de tout vote personnel**

tout votant doit pouvoir **vérifier** que son vote a bien été pris en compte :

. il sait juste vérifier que son vote est dans le tableau final, sans pouvoir en vérifier le contenu réel

. il sait en plus vérifier le contenu de son vote

**- résultat final vérifiable par chaque votant**

tout votant doit pouvoir **vérifier** que le résultat publié est bien la somme de tous les votes émis

**- validation de chaque vote par les autorités**

les autorités doivent **vérifier** que seuls les ayants droit ont voté, 1 seule fois, de manière correcte

**- vérifiabilité => le système doit fournir des preuves mathématiques (compréhension réservée aux experts) de la correction du résultat**

- système entièrement vérifiable comme **BELENIOS** => avancée pour des élections à distance

## * Vérifiabilité et résistance à la coercition = propriétés antagonistes

- on doit : montrer qu'un vote a été inclus dans le résultat, ne pas montrer son vote à un tiers
- satisfaire ces deux propriétés => recours important à la **cryptographie**

(logiciel délicat **CIVITAS** : seul à assurer **vérifiabilité** et **résistance à la coercition**)

# F1. Besoins spécifiques du vote en ligne

*** Contrôle de l'authentification de l'électeur**

- **vote à l'urne** : l'électeur s'identifie avec carte d'électeur ou pièce d'identité

- identifiants de **vote électronique** reçus par mail, poste ou SMS => **risques** :
  . vote par un tiers à la place de l'électeur
  => contrôle des envois : fabricants d'identifiants, opérateurs téléphoniques ou de messagerie
  . **vente des identifiants** par un électeur indélicat !

*** Impossibilité de manipulation du résultat du vote par des autorités**

les organisations gouvernementales, organisations de sécurité de l'Etat (CIA, NSA…)
ne doivent pas pouvoir influencer ou altérer le résultat du vote populaire

# F2. Avantages de BELENIOS

## * Qualités caractéristiques

- système de vote relativement simple, facile à mettre en œuvre
- logiciel libre aux sources disponibles => permet toutes analyses de sécurité !
- assure la confidentialité des votes
- **entièrement vérifiable par tous** :
  . tout électeur peut suivre son bulletin dans l'urne
  . tout électeur peut calculer le résultat de l'élection sous une forme chiffrée
  . tout électeur peut vérifier les calculs effectués par les autorités de décryptage
- système **ouvert et vérifiable** comme **BELENIOS**

=> . convient très bien pour des **consultations** (**sondage**, **référendum**)
. progrès à encore améliorer pour des **élections** à distance

## * Différence fondamentale

**les solutions commerciales actuelles :**

- ont un système et un code source secrets, au fonctionnement réservé à des experts habilités
- ne permettent pas à chacun de vérifier que le résultat proclamé est conforme
- ont une sécurité reposant sur l'intégrité du serveur de vote :
  un attaquant sur le serveur peut détruire des bulletins sans que les électeurs le voient
  (alors que **BELENIOS** détecte de telles attaques)

# F3. Inconvénients de BELENIOS

## BELENIOS non totalement adapté à des élections à forts enjeux

* élections politiques à forts enjeux : présidentielles, législatives…

  => nécessitent une **précision absolue**

* **BELENIOS** ne garantit pas la **résistance à la coercition**

* un **ordinateur piraté** par un ***hacker*** pourrait :

  . compromettre le **secret du vote** en transmettant la valeur du vote d'un électeur à un tiers

  . voter contrairement au vœu de l'électeur

# Un petit "plus" en cryptographie si affinités ?

## * Film : "*Imitation game*" (2014)

sur **Alan TURING** artisan de la victoire décodant la **machine nazie** ***Enigma***

- grand mathématicien logicien britannique (**1912** - **1954**)
- sportif de haut niveau (marathon en 2h46 : temps alors olympique!)
- travaux scientifiques successifs :
  . fondements scientifiques de l'informatique :
  machine de **TURING** (*enseignée en master 2 d'informatique*)
  => résolution du problème arithmétique de la décidabilité
  concepts de programmation et fonction calculable
  . 1er projet détaillé d'un ordinateur, suivant les travaux de **VON NEUMANN** sur sa structure
  . exploration du problème de l'intelligence artificielle (test de **TURING**)
  . biologie : modélisation de la morphogenèse du vivant (structures de **TURING**)

- **décryptage** en **1941** **d'***Enigma* crue **inviolable** par les **nazis**
  . services secrets polonais (mathématicien : **REJEWSKI**)
  . services secrets anglais : équipe de **TURING** à **Bletchley Park**

## * Livre : "*Histoire des codes secrets*" , par **Simon SINGH**

# THE END ?

**(cf annexe ? : aperçu sur le cryptage *complètement homomorphe* de GENTRY)**

***Merci de votre attention…***

# G - Problème de la confidentialité client fournisseur : aperçu sur le cryptage complètement homomorphe de GENTRY

**G1. Clé secrète et multiple "approximatif" introduisant du "bruit"**

**G2. Cryptage et décryptage**

**G3. Exemple rudimentaire**

**G4 - Cryptage additivement (XOR) (et complètement) homomorphe**

**G5 - Cryptage multiplicativement (AND) (donc complètement) homomorphe**

**G6 - "Bruit" croissant avec les opérations**

# G1 - Clé secrète et multiple "approximatif" introduisant du "bruit"

*** Différence avec les envois précédents de messages cryptés**

le **client** *Alice* crypte le message et l'envoie au **fournisseur** *Bob*, qui le traite crypté suivant les opérations demandées par *Alice*, et renvoie les résultats cryptés à *Alice* qui les décrypte

*** Clé secrète p**

très grand entier **p impair** par ex. de **500** chiffres binaires

*** Multiple "approximatif" de p introduisant du "bruit" 2r**

. entier de la forme **p q + 2r**

. **q** entier aléatoire par ex. de **plusieurs millions** de chiffres binaires

. **r** entier aléatoire par ex. de **20** chiffres binaires => 2 **r** pair introduit une sorte de **"bruit"**

. avantage : **espion** cherchant **p** par essais de multiples "approximatifs" => calculs irréalisables

# G2 - Cryptage et décryptage

*** Cryptage C d'un bit b**

**C(b) = b + pq + 2r = pq + (b+2r)**, avec **b+2r < p** (*à cause du nombre de chiffres*)

=> présentation de **C(b)** comme une **division euclidienne** par **p** (quotient **q**, reste **b+2r**)

*** Décryptage D du crypté d'un bit b**

**c = C(b)** : division euclidienne par **p**, avec quotient **q** et reste **b + 2r** => **b + 2r = c mod p**

**b** = reste dans la division par **2** de **b + 2r** (**b < r** *à cause du nombre de chiffres*)

=> **b = b + 2r mod 2 = (c mod p) mod 2 = D (c)**

# G3 - Exemple rudimentaire

*** Cryptage**

$p = 521$, $r_1 = 6$, $r_2 = 4$, $q_1 = 737$, $q_2 = 369$, $b_1 = 1$, $b_2 = 0$

$c_1 = C(b_1) = p\,q_1 + 2r_1 + b_1 = 737 \cdot 521 + 2 \cdot 6 + 1 = 383990$

$c_2 = C(b_2) = p\,q_2 + 2r_2 + b_2 = 369 \cdot 521 + 2 \cdot 4 + 0 = 192257$

*** Décryptage**

$c = C(b_1) + C(b_2) = c_1 + c_2 = (383990 + 192257) = 576247$

$c$ est le crypté d'un bit $b = D(c) = (c \bmod p) \bmod 2 = (576247 \bmod 521) \bmod 2 = 21 \bmod 2$

$21 \bmod 2 = 1 = 1 \oplus 0 \Rightarrow C(b_1) + C(b_2) = C(b_1 \oplus b_2)$

=> **cryptage additivement (XOR) homomorphe**

*** Rappel de la table d'addition $\oplus$ modulo 2 (XOR logique)**

| $\oplus$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| XOR | F | V |
|---|---|---|
| F | F | V |
| V | V | F |

# G4 - Cryptage additivement (XOR) homomorphe

*Composé par **XOR** (⊕) des données en clair = **addition** des **cryptés** ?*

*** Les 2 bits cryptés**

$\mathbf{b_1}$ et $\mathbf{b_2}$ 2 bits => $\mathbf{C(b_1) = c_1 = p\ q_1 + 2r_1 + b_1}$ , $\mathbf{C(b_2) = c_2 = p\ q_2 + 2r_2 + b_2}$

$\mathbf{r_1}$ et $\mathbf{r_2}$ : **20** chiffres binaires, **p** : **500** chiffres binaires, $\mathbf{q_1}$ et $\mathbf{q_2}$ : **millions** de chiffres binaires

*** Addition des cryptés des 2 bits**

$\mathbf{c = C(b_1) + C(b_2) = c_1 + c_2 = (q_1 + q_2)\ p + 2\ (r_1 + r_2) + (b_1 + b_2)}$

=> **c** se présente comme une **division euclidienne** par **p**,

avec quotient $\mathbf{q_1 + q_2}$ et reste $\mathbf{2\ (r_1 + r_2) + (b_1 + b_2)}$ (*à cause du nombre de chiffres*)

*** Composé par XOR (⊕) des données en clair**

**c** est le crypté d'un bit $\mathbf{b = D(c) = (c \bmod p) \bmod 2 = [2(r_1+r_2)+(b_1+b_2)] \bmod 2 = (b_1+b_2) \bmod 2}$

$\mathbf{b = (b_1+b_2) \bmod 2}$ (addition modulo 2) = $\mathbf{b_1 \oplus b_2}$ (**XOR**) :

$$\mathbf{c = C(b_1) + C(b_2) = C(b_1 \oplus b_2)}$$

# G5 - Cryptage multiplicativement (AND) (donc complètement) homomorphe

*Composé par multiplication (**AND**) des données en clair =**multiplication** des **cryptés** ?*

*** Les 2 bits cryptés**

$\mathbf{b_1}$ et $\mathbf{b_2}$ 2 bits => $\mathbf{C(b_1) = c_1 = p\ q_1 + 2r_1 + b_1}$ , $\mathbf{C(b_2) = c_2 = p\ q_2 + 2r_2 + b_2}$

*** Multiplications des cryptés des 2 bits**

$\mathbf{c = C(b_1)C(b_2) = c_1\ c_2 = [q_1\ q_2 + q_1\ (2r_2 + b_2) + q_2\ (2r_1 + b_1)]p + 2\ (2r_1\ r_2 + b_1\ r_2 + b_2\ r_1) + b_1\ b_2}$

=> **c** se présente comme une **division euclidienne** par **p**,

avec quotient = $\mathbf{[q_1\ q_2 + q_1\ (2r_2 + b_2) + q_2\ (2r_1 + b_1)]}$, reste = $\mathbf{2(\ 2r_1\ r_2 + \ b_1\ r_2 + \ b_2\ r_1) + b_1\ b_2}$

(*à cause du nombre de chiffres*)

*** Composé par multiplication (AND) des données en clair**

**c** est le crypté d'un bit $\mathbf{b = D(c)}$ = (**c** mod **p**) mod **2**

= $\mathbf{2(\ 2r_1\ r_2 + \ b_1\ r_2 + \ b_2\ r_1) + b_1\ b_2}$ mod **2** = $\mathbf{b_1\ b_2}$ mod **2**

$\mathbf{b = b_1\ b_2}$ mod **2** (multiplication modulo 2) = $\mathbf{b_1\ b_2}$ = $\mathbf{b_1 \wedge b_2}$ (**AND**) :

$\mathbf{c = C(b_1)\ C(b_2) = C(b_1\ b_2)}$

*** Cryptage complètement homomorphe**

additivement homomorphe + multiplicativement homomorphe => complètement homomorphe

# G6 - Le "bruit"

*** Rappel : bruit d'un crypté**

crypté **c** de **b** = **C(b)** = **b** + **pq** + **2 r** : **"bruit"** = **2 r**

*** Addition => addition du bruit**

**"*bruit*"** sur $C(b_1) = 2\,r_1$ , **"*bruit*"** sur $C(b_2) = 2\,r_2$

$c = C(b_1) + C(b_2) = c_1 + c_2 = (q_1 + q_2)\,p + (b_1 + b_2) + 2\,(r_1 + r_2)$

=> **"*bruit*"** sur $C(b_1) + C(b_2) = 2\,(r_1+r_2) = 2\,r_1 + 2\,r_2$

*** Multiplication => multiplication (approximativement) du bruit**

**"*bruit*"** sur $C(b_1) = 2\,r_1$ , **"*bruit*"** sur $C(b_2) = 2\,r_2$

$c = C(b_1)C(b_2) = c_1c_2 = [q_1\,q_2 + q_1\,(2r_2 + b_2) + q_2\,(2r_1 + b_1)]\,p + 2(2r_1\,r_2 + b_1\,r_2 + b_2\,r_1) + b_1b_2$

=> **"*bruit*"** sur $C(b_1)\,C(b_2) = 2(2r_1\,r_2 + b_1\,r_2 + b_2\,r_1) = 2\,r_1\,.\,2\,r_2 + 2(b_1\,r_2 + b_2\,r_1)$

et le 2e terme est négligeable par rapport à $2\,r_1\,.\,2\,r_2$

# G6 - Le "bruit"

*** Croissance du bruit avec le nombre d'opérations**

- le bruit grossit à chaque nouvelle opération :

doit rester < **p** pour que l'on puisse déchiffrer (crypté => correspondre à une division par **p**)

(bruit devenant trop grand => la procédure de décryptage retourne un message erroné

*** Réduction du bruit par la technique de "*bootstrapping*"**

- croissance du bruit => réduire le nombre d'opérations
- => utiliser du "***bootstrapping***" = *rafraîchissement, réamorçage…*

(*faculté miraculeuse du **baron de Münchhausen** de s'élever vers le ciel en tirant sur ses bottes*)

= évaluation régulière homomorphe du décryptage réduisant le bruit à un niveau acceptable

(c = crypté **c** de **m** de bruit **k** => retour d'un crypté $\mathbf{c_0}$ de **m** de bruit $\mathbf{k_0}$ tel que $||\mathbf{k_0}|| < ||\mathbf{k}||$ )