Le calcul sécurisé, deuxième cours

Chiffrement totalement homomorphe: calculer sur des données chiffrées (1re partie)

Xavier Leroy

2025-11-13

Collège de France, chaire de sciences du logiciel
xavier.leroy@college-de-france.fr

Rappels sur le chiffrement

Interface d'un chiffrement asymétrique

Génération de clé :

$$\mathit{Keygen} : \text{clé publique} \times \text{clé privée} \qquad \text{(randomisé)}$$

Chiffrement :

$$\mathcal{E} : \text{clé publique} \times \text{message} \to \text{chiffré} \qquad \text{(randomisé)}$$

Déchiffrement :

$$\mathcal{D} : \text{clé privée} \times \text{chiffré} \to \text{message} \qquad \text{(déterministe)}$$

Correction fonctionnelle

$$\begin{aligned} \mathit{Keygen} &: \text{clé publique} \times \text{clé privée} && \text{(randomisé)} \\ \mathcal{E} &: \text{clé publique} \times \text{message} \to \text{chiffré} && \text{(randomisé)} \\ \mathcal{D} &: \text{clé privée} \times \text{chiffré} \to \text{message} && \text{(déterministe)} \end{aligned}$$

Intuitivement : $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m.$

Plus précisément :

$$\mathcal{D}_{sk}(c) = m \qquad \text{pour tous } (pk, sk) \leftarrow \mathit{Keygen},\ c \leftarrow \mathcal{E}_{pk}(m)$$

En termes plus probabilistes :

$$\Pr\left[(pk, sk) \leftarrow \mathit{Keygen}; \mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m\right] = 1$$

Avec aléa explicite r, r' :

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, r)) = m \qquad \text{si } (pk, sk) = \mathit{Keygen}(r')$$

Sécurité du chiffrement

Sécurité inconditionnelle : (informellement)

Un attaquant qui connaît divers couples (clair, chiffré), et un chiffré c, ne peut déduire aucune information du clair m.

Sécurité du chiffrement

Sécurité inconditionnelle : (informellement)

Un attaquant qui connaît divers couples (clair, chiffré), et un chiffré c, ne peut déduire aucune information du clair m.

Sécurité calculatoire : (informellement)

Un attaquant en temps polynomial en la taille n des clés qui connaît divers couples (clair, chiffré) et un chiffré c a une probabilité $\leq \varepsilon(n)$ de déduire certaines informations du clair m, où ε est une fonction négligeable.

(Négligeable = qui décroît très rapidement avec n, p.ex. 2^{-n})

Modèles d'attaque et objectifs de sécurité

Modèles d'attaque :

- **KPA** (*known plaintext attacks*) : l'attaquant a obtenu des couples (clair, chiffré) sans les choisir.
- **CPA** (*chosen plaintext attacks*) : l'attaquant peut chiffrer tout texte clair de son choix. (Clé publique.)
- **CCA** (*chosen ciphertext attacks*) : CPA + l'attaquant peut faire déchiffrer tout chiffré de son choix sauf celui à attaquer.

Modèles d'attaque et objectifs de sécurité

Modèles d'attaque :

- **KPA** (*known plaintext attacks*) : l'attaquant a obtenu des couples (clair, chiffré) sans les choisir.
- **CPA** (*chosen plaintext attacks*) : l'attaquant peut chiffrer tout texte clair de son choix. (Clé publique.)
- **CCA** (*chosen ciphertext attacks*) : CPA + l'attaquant peut faire déchiffrer tout chiffré de son choix sauf celui à attaquer.

Objectifs de sécurité :

- **IND** (*indistinguishability*) : l'attaquant ne peut pas distinguer un chiffré d'un bruit, ou un chiffré d'un autre chiffré.
- **NM** (*non-malleability*) : à partir d'un chiffré c, l'attaquant ne peut pas produire un chiffré c' tel que $\mathcal{D}(c') = \mathcal{D}(c) \oplus 1$ p.ex.

Le critère IND-CPA (*indistinguishability with chosen plaintexts*)

Présenté comme un jeux à deux joueurs, l'attaquant A et le challenger C :

> C : tire $(pk, sk) \leftarrow \mathit{Keygen}$, envoie pk à A.
>
> A : choisit deux messages m_1, m_2, les envoie à C.
>
> C : choisit $i \in \{1, 2\}$, envoie $c = \mathcal{E}(pk, m_i)$ à A.
>
> A : devine si c est le chiffré de m_1 ou de m_2.

Le chiffrement est IND-CCA si la probabilité que l'attaquant devine correctement est bornée par $1/2 + \varepsilon(n)$.

(Note : un chiffrement déterministe n'est pas IND-CPA.)

La sécurité prouvée

Par réduction à un problème mathématique que l'on pense être algorithmiquement difficile :

Attaque qui réussit
avec probabilité non négligeable

réduction

Algorithme en temps polynomial probabiliste
qui résout un problème algorithmiquement difficile

Exemples de problèmes algorithmiquement difficiles

Pour un ordinateur classique :

- La factorisation : étant donné N, trouver $p, q < N$ tels que $N = pq$.
- Le problème RSA : étant donnés N, e premier avec $\varphi(N)$ et $c < N$, trouver m tel que $m^e = c \bmod N$.
- Le logarithme discret : étant donnés g générateur d'un groupe G et $x \in G$, trouver $a \in \mathbb{N}$ tel que $g^a = x$.
- Le problème Diffie-Hellman : étant donnés g^a et g^b (a, b inconnus), calculer g^{ab}.

Exemples de problèmes algorithmiquement difficiles

Pour un ordinateur classique :

- La factorisation : étant donné N, trouver $p, q < N$ tels que $N = pq$.
- Le problème RSA : étant donnés N, e premier avec $\varphi(N)$ et $c < N$, trouver m tel que $m^e = c \bmod N$.
- Le logarithme discret : étant donnés g générateur d'un groupe G et $x \in G$, trouver $a \in \mathbb{N}$ tel que $g^a = x$.
- Le problème Diffie-Hellman : étant donnés g^a et g^b (a, b inconnus), calculer g^{ab}.

Pour un ordinateur quantique :

- Le problème MQ : résoudre un système d'équations polynomiales de degré 2 à plusieurs inconnues.

Quantifier la sécurité

Choisir les cryptosystèmes et leurs tailles de clés de sorte que les meilleures attaques connues sont en temps 2^{λ}.

λ est le paramètre de sécurité.

Quelques recommandations du NIST :

Paramètre de sécurité λ	112	128	192	256
Chiffrement symétrique	3DES	AES-128	AES-192	AES-256
Hachage	SHA-224	SHA-256	SHA-284	SHA-512
RSA (factorisation)	2048	3072	7680	15360
Courbes elliptiques	224+	256+	384+	512+
ML-DSA (post-quantique)	—	1312 / 2560	1952 / 4032	2592 / 4896

Chiffrement semi-homomorphe

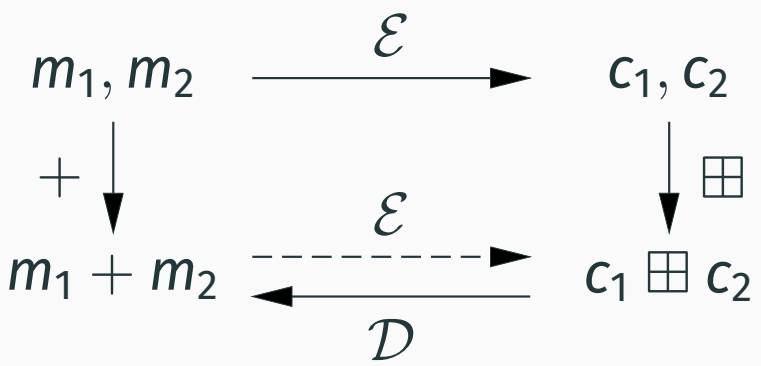
Chiffrement semi-homomorphe

Un chiffrement qui est homomorphe pour une seule opération, typiquement l'addition ou la multiplication des messages en clair, mais pas les deux.

Dans le cas additif, on a une opération $\boxplus$ sur les chiffrés telle que «le chiffré d'une somme est le $\boxplus$ des chiffrés» :

$$c_1 \boxplus c_2 \in \mathcal{E}(m_1 + m_2) \quad \text{pour tous } c_1 \in \mathcal{E}(m_1),\ c_2 \in \mathcal{E}(m_2)$$

En corollaire, $m_1 + m_2 = \mathcal{D}(\mathcal{E}(m_1) \boxplus \mathcal{E}(m_2))$.



RSA (sans *padding*) est homomorphe pour la multiplication

Chiffrement RSA : avec p, q premiers,
— clé publique : $N = pq$ et e premier avec $(p-1)(q-1)$;
— clé secrète : $d = e^{-1} \pmod{(p-1)(q-1)}$.

$$\mathcal{E}(m) \stackrel{def}{=} m^e \bmod N \qquad \mathcal{D}(c) \stackrel{def}{=} c^d \bmod N$$

Si m_1, m_2 sont deux messages en clair,

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e = \mathcal{E}(m_1 \cdot m_2) \pmod{N}$$

La multiplication modulo N a pour opérateur homomorphe la multiplication modulo N.

Inutilisable : RSA sans *padding* n'est pas IND-CPA ; tous les schémas de *padding* aléatoire connus cassent l'homomorphisme.

Le chiffrement d'ElGamal (EG)

Un groupe fini $(G, \cdot)$ d'ordre q engendré par g.

Clé secrète : $x \in \{1, \ldots, q-1\}$. Clé publique : $h \stackrel{def}{=} g^x$.

Chiffrement randomisé (IND-CCA) :

$$\mathcal{E}(m) = (g^r, h^r \cdot m) \quad \text{avec } r \in \{1, \ldots, q-1\} \text{ aléatoire}$$

Note : les messages sont des éléments de G.

Déchiffrement : $\mathcal{D}((a, b)) = (a^x)^{-1} \cdot b$

Homomorphe pour l'opération $\cdot$ du groupe :

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{r_1} \cdot g^{r_2}, (h^{r_1} \cdot m_1) \cdot (h^{r_2} \cdot m_2)) \\ &= (g^r, h^r \cdot (m_1 \cdot m_2)) \qquad (\text{avec } r = r_1 + r_2 \bmod q) \\ &= \text{un chiffré de } m_1 \cdot m_2 \end{aligned}$$

Le chiffrement d'ElGamal exponentiel (EEG)

Les messages en clair sont des entiers $m \in \{0, \ldots, q-1\}$,
que l'on envoie dans G par exponentiation g^m.

Chiffrement :

$$\mathcal{E}(m) = EG.\mathcal{E}(g^m) = (g^r, h^r \cdot g^m) \quad \text{avec } r \text{ aléatoire}$$

Déchiffrement :

$$\mathcal{D}(c) = \log_g(EG.\mathcal{D}(c)) \quad \text{(logarithme discret en base } g\text{)}$$

Praticable pour de petits messages m (votes, sommes d'argent).

Homomorphe pour l'addition des messages (modulo q) :

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= EG.\mathcal{E}(g^{m_1}) \cdot EG.\mathcal{E}(g^{m_2}) \\ &= \text{un chiffré EG de } g^{m_1} \cdot g^{m_2} = g^{m_1+m_2} \\ &= \text{un chiffré EEG de } (m_1 + m_2) \bmod q \end{aligned}$$

Le chiffrement de Paillier

Utilise le fait que certains logarithmes discrets modulo n^2 sont faciles à calculer :

$$(1+n)^x = 1 + xn + \binom{x}{2} n^2 + \ldots = 1 + xn \pmod{n^2}$$

d'où $L((1+n)^x \bmod n^2) = x$ avec $L(u) \stackrel{def}{=} (u-1)/n$.

Génération de clé : p, q nombres premiers de même longueur.

Clé publique : (n, g) avec $n = pq$ et $g = 1 + n$.

Clé secrète : (α, β) avec $\alpha = \varphi(n) = (p-1)(q-1)$
et $\beta = L(g^\alpha \bmod n^2)^{-1} \bmod n$.

Le chiffrement de Paillier

Chiffrement : $\mathcal{E}(m) = g^m \cdot r^n \bmod n^2$
avec $r \in \{1, \ldots, n-1\}$ aléatoire et premier avec n.

Déchiffrement : $\mathcal{D}(c) = L(c^\alpha \bmod n^2) \cdot \beta \bmod n$

Ce chiffrement est homomorphe pour l'addition modulo n et la multiplication par une constante k :

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) \bmod n^2 &= g^{m_1+m_2}(r_1 \cdot r_2)^n \bmod n^2 \\ &= \text{un chiffré de } (m_1 + m_2) \bmod n \\ \mathcal{E}(m)^k \bmod n^2 &= g^{km} r^{kn} \bmod n^2 \\ &= \text{un chiffré de } km \bmod n \end{aligned}$$

Pas de relation entre $\mathcal{E}(m_1 m_2 \bmod n)$ et $\mathcal{E}(m_1), \mathcal{E}(m_2)$.

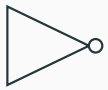
Chiffrement faiblement homomorphe

Chiffrement homomorphe et circuits booléens

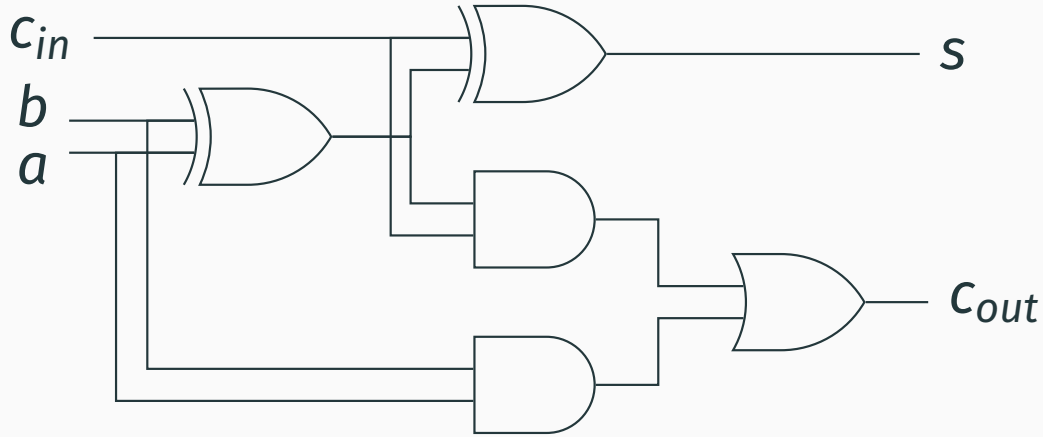
Un chiffrement homomorphe pour l'addition **et** la multiplication (d'entiers modulo p, ou juste modulo 2) est dit totalement homomorphe, car il permet d'évaluer

- des polynômes à plusieurs variables ;
- tous les circuits booléens (combinatoires).

Arithmétisation des circuits :

Porte logique	modulo 2	modulo $p > 2$
NOT 	$x + 1$	$1 - x$
AND 	xy	xy
OR 	$x + y + xy$	$1 - (1 - x)(1 - y)$
XOR 	$x + y$	$x + y - 2xy$

Exemple de circuit : additionneur complet 1 bit

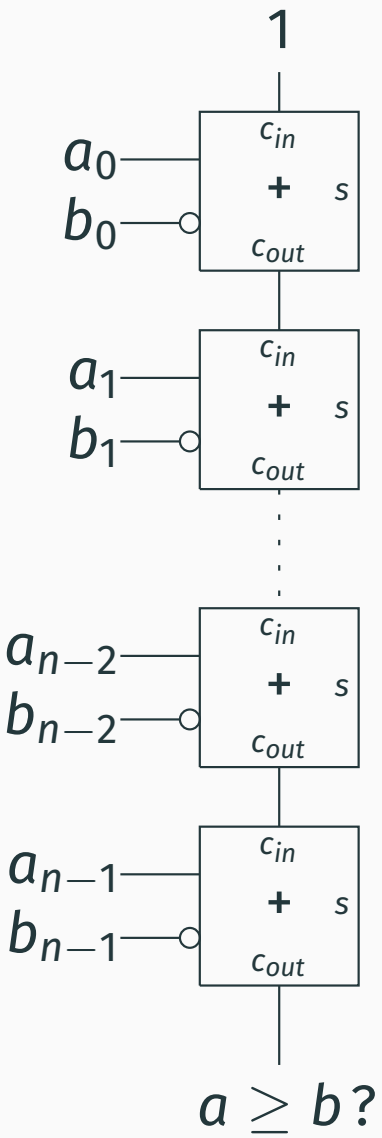


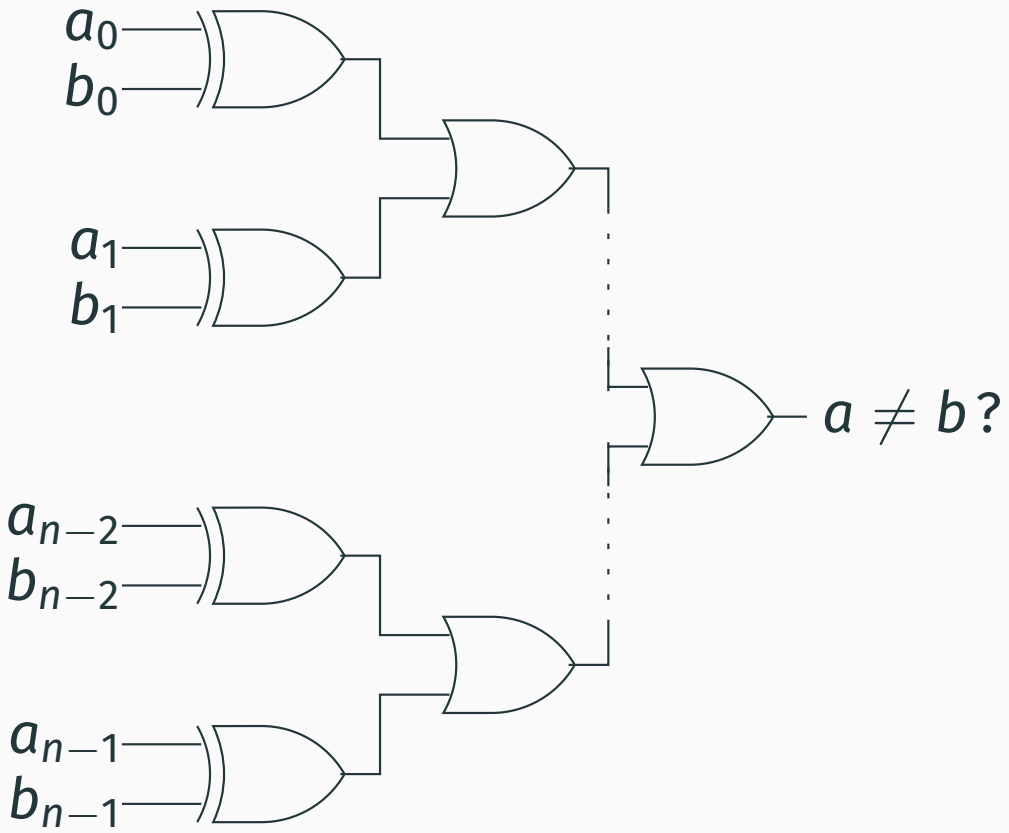
Les polynômes correspondants :

$$s = a + b + c_{in} \pmod 2$$

$$c_{out} = ab + (a + b)c_{in} + ab(a + b)c_{in} = ab + ac_{in} + bc_{in} \pmod 2$$

Exemples de circuit : comparateurs d'entiers n bits





Difficulté du chiffrement totalement homomorphe

On ne connaît pas de chiffrement simple qui soit sûr et totalement homomorphe.

Rabin: May I add a remark? If you then propose to have different p and q for each customer, which is quite difficult and impractical, sometimes a non-innocent by-stander has knowledge of how much money you deposited. The other problem again exists. You must assume at least spotty partial information about the data which is going to be protected.

Rivest: All the systems I've presented, I think, are susceptible to variations of that kind of attack. I do not consider any of them very satisfactory for precisely those kinds of reasons.

(Séance de questions sur l'article fondateur *On data banks and privacy homomorphisms* de Rivest, Adleman, Dertouzos, 1978).

Chiffrement faiblement homomorphe

On sait construire des chiffrements faiblement homomorphes : homomorphes pour un nombre limité d'additions et de multiplications.

Cela permet d'évaluer homomorphiquement des circuits de profondeur bornée.

La procédure de *bootstrap* (introduite par Craig Gentry en 2009) permet d'obtenir un chiffrement totalement homomorphe à partir d'un chiffrement faiblement homomorphe.

Chiffrer en ajoutant du bruit

Une approche commune à plusieurs cryptosystèmes récents (à base de réseaux euclidiens ou du problème LWE, p.ex.).

$$\text{Chiffrer} \quad = \quad \text{ajouter beaucoup de bruit au message.}$$

Le bruit a une structure cachée qui permet de l'éliminer si on connaît la clé privée :

$$\text{Déchiffrer} \quad = \quad \begin{array}{l} \text{réduire le bruit à l'aide de la clé privée ;} \\ \text{retrouver le texte clair le plus proche.} \end{array}$$

Un chiffrement «bruité» avec des nombres entiers

(van Dijk, Gentry, Halevy, Vaikuntanathan, *Fully homomorphic encryption over the integers*, Eurocrypt 2010)

Clé privée : un entier impair p.

Chiffrement symétrique (avec la clé privée) d'un bit b :

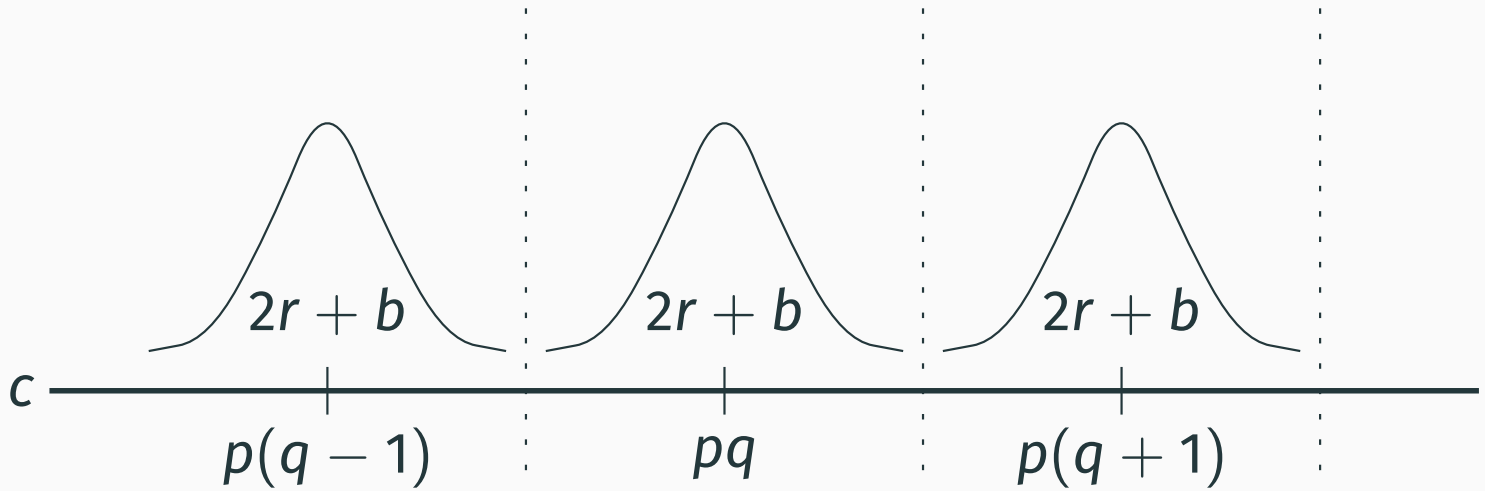
$$\mathcal{E}_p(b) = pq + 2r + b \qquad \begin{array}{l} q \text{ entier naturel aléatoire} \gg p \\ r \text{ entier relatif aléatoire, } |r| < p/4 \end{array}$$

Déchiffrement :

$$\mathcal{D}_p(c) = (c - p\lfloor c/p \rceil) \bmod 2 = LSB(\lfloor c/p \rceil) \oplus LSB(c)$$

où $\lfloor \cdot \rceil$ est l'arrondi à l'entier le plus proche, et $\oplus$ le ou exclusif.

Illustration du chiffrement et du déchiffrement



Connaissant le chiffré c et la clé p, on retrouve $q = \lfloor c/p \rceil$, et

$$(c - pq) \bmod 2 = (2r + b) \bmod 2 = b$$

Comme p est impair, pq est impair ssi q est impair, d'où

$$b = LSB(c - pq) = LSB(c) \oplus LSB(pq) = LSB(c) \oplus LSB(q)$$

(Note : q suffit pour masquer b ; r est là pour empêcher l'attaque par PGCD.)

Le problème du PGCD approché

Étant donnés des nombres x_i de la forme $pq_i + e_i$ avec $|e_i| \leq E$, peut-on retrouver p ?

Si $E = 0$: il suffit de calculer $\gcd(x_i, x_j)$ pour quelques $i \neq j$.

Si $E = 1$: pareil, en considérant aussi les $x_i + 1$ et les $xi - 1$.

Si E, p, q_i suffisamment grands : les algorithmes connus sont exponentiels en E.

Tailles recommandées par van Dijk *et al* :

$$e_i : \lambda \text{ bits} \qquad p : \lambda^2 \text{ bits} \qquad q_i : \lambda^5 \text{ bits}$$

où λ est le paramètre de sécurité.

Chiffrement à clé publique

Clé publique pk = un ensemble de chiffrements de zéro $pq_i + 2r_i$ (avec q_i aléatoire et r_i aléatoire, $|r_i| \ll p$).

Chiffrement à clé publique :

$$\mathcal{E}_{pk}(b) = \sum_{x \in S} x + 2r + b$$

S sous-ensemble aléatoire de pk

r entier relatif aléatoire, $|r| < p/4$

van Dijk *et al* recommandent d'utiliser λ^5 chiffrés de zéro, ce qui donne une clé publique de taille λ^{10} bits (!!)

Chiffrement faiblement homomorphe

Pour l'addition :

$$\begin{aligned}
&(pq_1 + 2r_1 + b_1) + (pq_2 + 2r_2 + b_2) \\
&\quad = \quad p(q_1 + q_2) + 2(r_1 + r_2 + (b_1 + b_2)/2) + (b_1 + b_2) \bmod 2 \\
&\quad = \quad \text{un chiffré de } b_1 \oplus b_2 \text{ si } r_1 + r_2 \text{ suffisamment petit}
\end{aligned}$$

Pour la multiplication :

$$\begin{aligned}
&(pq_1 + 2r_1 + b_1) \cdot (pq_2 + 2r_2 + b_2) \\
&\quad = \quad p(\cdots) + 2(2r_1r_2 + r_1b_2 + r_2b_1) + b_1b_2 \\
&\quad = \quad \text{un chiffré de } b_1 \wedge b_2 \text{ si } r_1r_2 \text{ suffisamment petit}
\end{aligned}$$

Gestion du bruit

Le bruit $N(c)$ d'un chiffré est

$$N(c) = \text{nombre de bits de } r \quad \text{si } c = pq + 2r + b$$

Pour que c se déchiffre correctement en b, il faut

$$N(c) < \text{nombre de bits de } p - 2$$

Le bruit augmente lentement par addition homomorphe :
$r_1, r_2 \mapsto r_1 + r_2 + (b_1 + b_2)/2$, d'où

$$N(c_1 + c_2) \leq \max(N(c_1), N(c_2)) + 1$$

Le bruit augmente rapidement par multiplication homomorphe :
$r_1, r_2 \mapsto 2(2r_1r_2 + r_1b_2 + r_2b_1)$, d'où

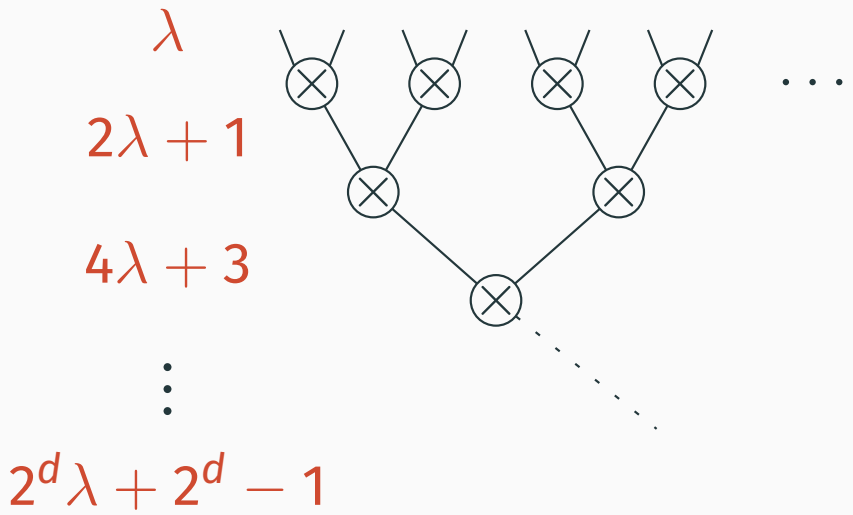
$$N(c_1 \cdot c_2) \leq N(c_1) + N(c_2) + 1$$

Profondeur multiplicative d'un circuit

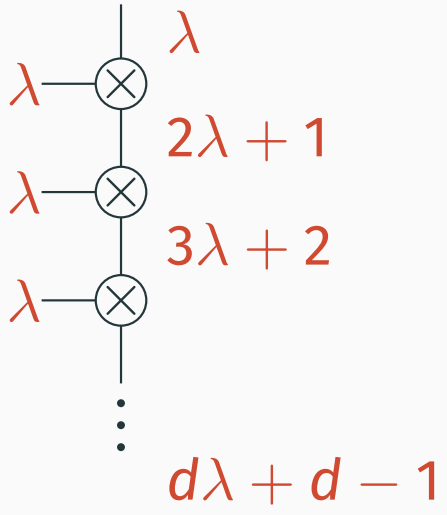
Profondeur multiplicative = le nombre max de portes «et»/«ou» entre une entrée et une sortie.

Si le bruit initial est λ bits, et si p a λ^2 bits, la profondeur multiplicative est limitée au pire à $\approx \log_2 \lambda$ et au mieux à $\approx \lambda$.

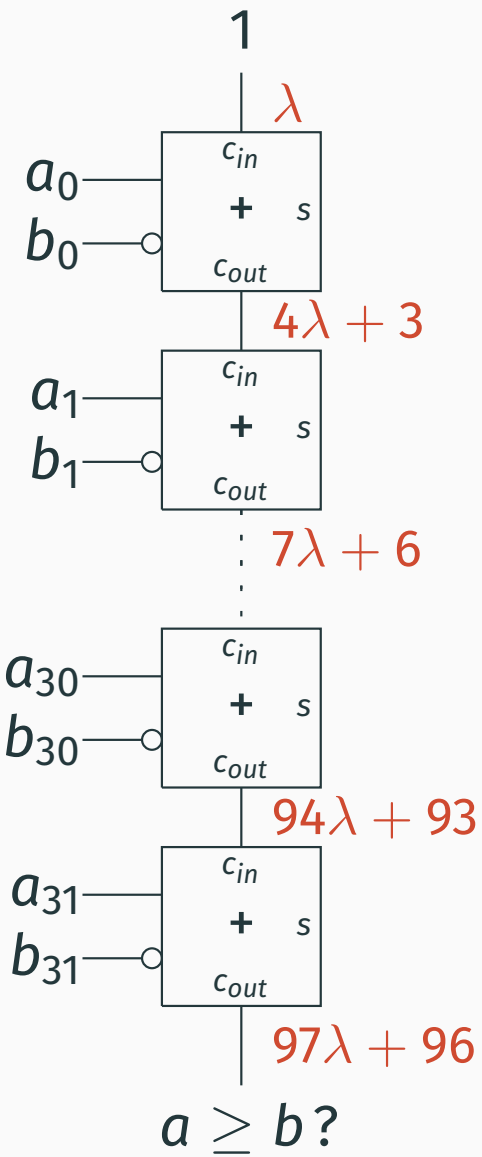
Croissance exponentielle du bruit :

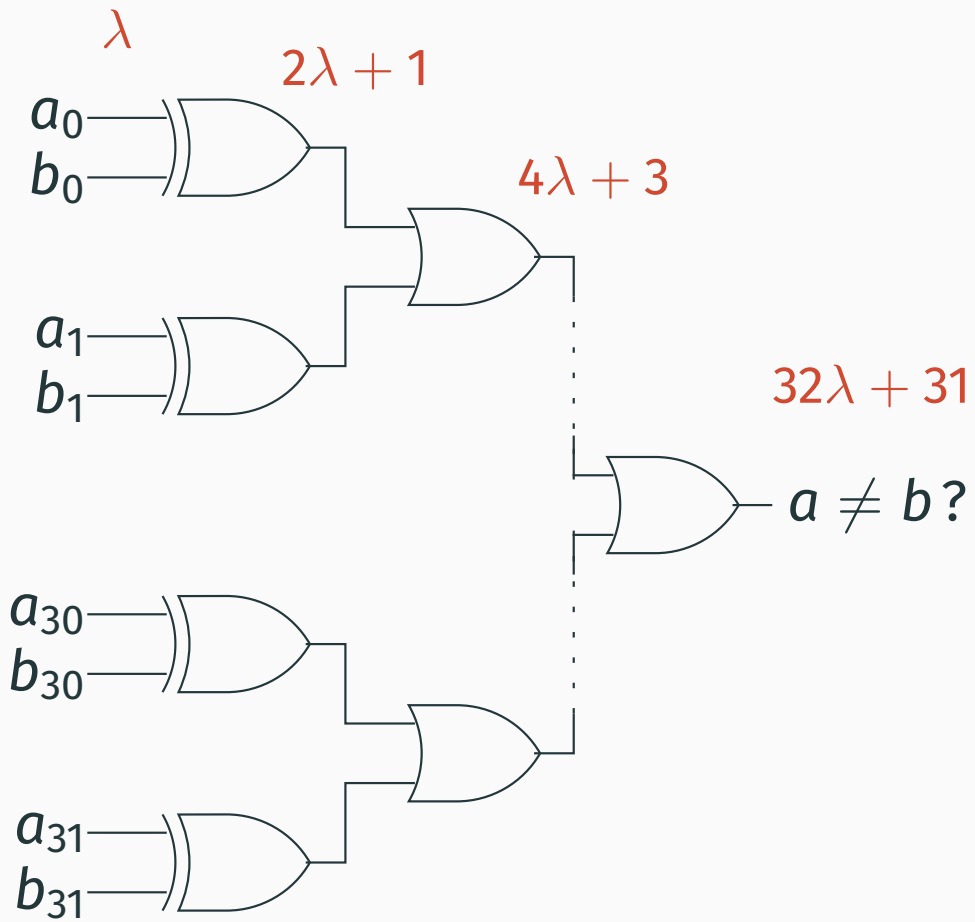


Croissance linéaire :



Bruit dans des circuits : comparateurs d'entiers 32 bits

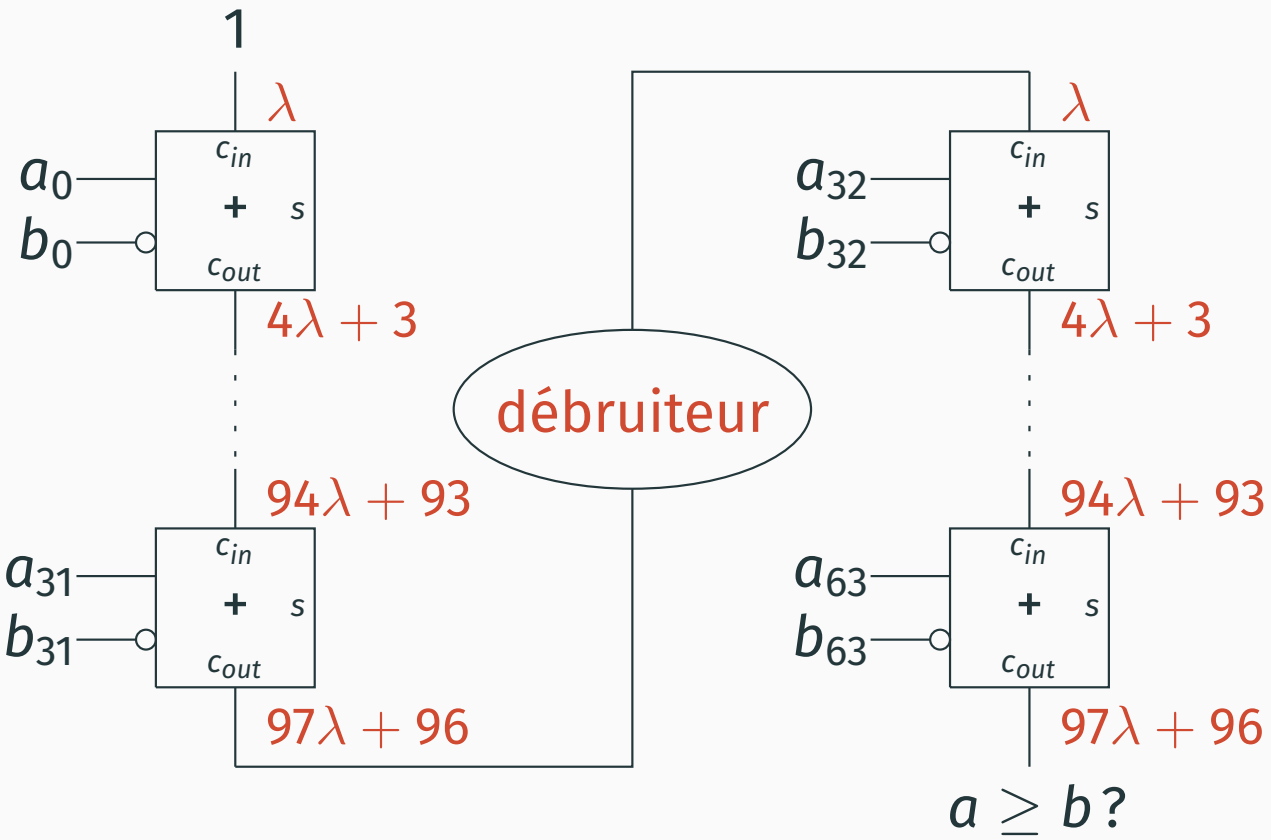




Chiffrement totalement homomorphe par *bootstrap*

Réduire le bruit à certains points du circuit

Pour évaluer homomorphiquement des circuits arbitraires, il faut pouvoir réduire le bruit aux points du circuit où il risque de dépasser le seuil.



Réduire le bruit par déchiffrement et rechiffrement

Comment réduire le bruit ?

Il suffit de déchiffrer puis rechiffrer !

$$pq + 2r + b \xrightarrow{\mathcal{D}_p} b \xrightarrow{\mathcal{E}_{pk}} pq' + 2r' + b$$

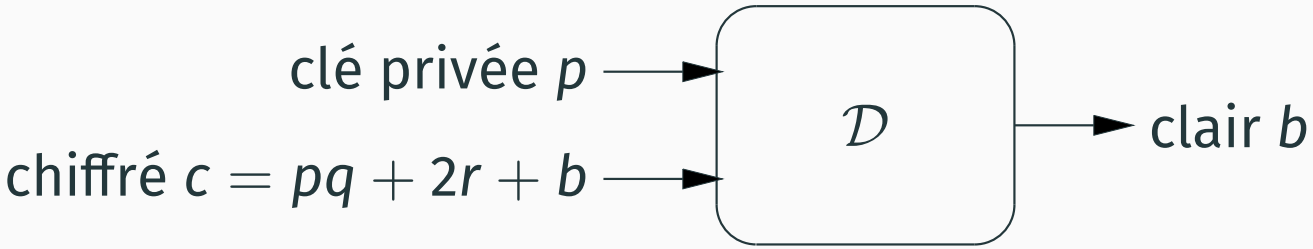
Chiffré très bruité
$(r \approx p/4)$

Chiffré peu bruité
$(r' \ll p/4)$

Problème : le calculateur n'a pas la clé de déchiffrement p …

Réduire le bruit par déchiffrement homomorphe

(Craig Gentry, *A fully homomorphic encryption scheme*, PhD, Stanford, 2009.)



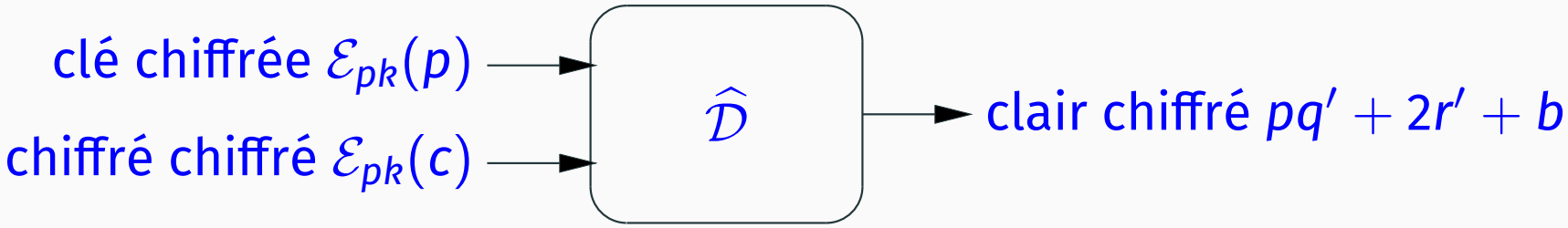
L'algorithme de déchiffrement $\mathcal{D}$ peut être réalisé par un circuit.

$$\mathcal{D}_p(c) = LSB(\lfloor c/p \rceil) \oplus LSB(c)$$

Supposons que la profondeur multiplicative de ce circuit est suffisamment faible pour qu'on puisse l'évaluer avec notre chiffrement faiblement homomorphe.

Réduire le bruit par déchiffrement homomorphe

(Craig Gentry, *A fully homomorphic encryption scheme*, PhD, Stanford, 2009.)



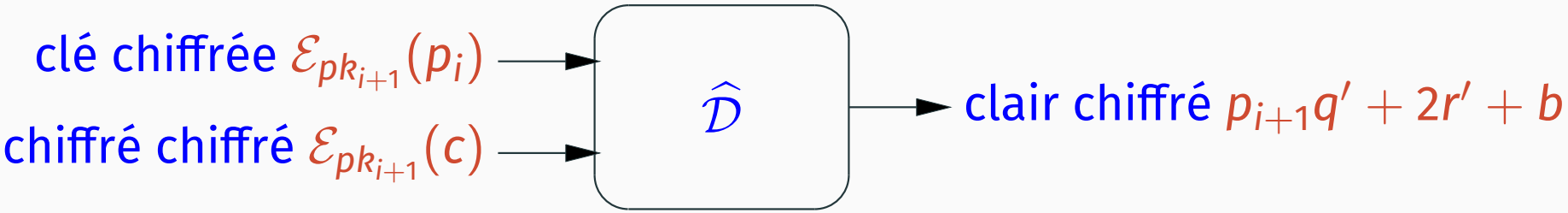
Le circuit $\widehat{\mathcal{D}}$ homomorphe à $\mathcal{D}$ prend la clé privée chiffrée et le chiffré c sur-chiffré, et produit un chiffré du texte clair.

Le résultat est un chiffré équivalent à c, mais dont le bruit dépend uniquement de la profondeur multiplicative du circuit $\widehat{\mathcal{D}}$.

On a donc réduit le bruit de c à un niveau connu.

Réduire le bruit par déchiffrement homomorphe

(Craig Gentry, *A fully homomorphic encryption scheme*, PhD, Stanford, 2009.)



Pour éviter la ≪sécurité circulaire≫ (chiffrer une clé avec elle-même), on peut changer de clés au moment du bootstrap.

On a une suite de clés $(p_1, pk_1), \ldots, (p_i, pk_i), \ldots$

On donne au calculateur les clés publiques pk_i et les clés privées chiffrées $\mathcal{E}_{pk_2}(p_1), \ldots, \mathcal{E}_{pk_{i+1}}(p_i), \ldots$.

Profondeur multiplicative du circuit de déchiffrement

$$\mathcal{D}_p(c) = LSB(\lfloor c/p \rceil) \oplus LSB(c)$$

On effectue la division par une multiplication par un quasi-inverse de p :

$$\mathcal{D}_p(c) = LSB(\lfloor cp'/2^N \rceil) \oplus LSB(c) \qquad \text{avec} \quad \frac{p'}{2^N} \approx \frac{1}{p}$$

Malgré cela, le circuit de multiplication cp' est un polynôme de degré élevé ($\geq$ nombre de bits de p) et ne peut pas être évalué homomorphiquement.

Simplifier le déchiffrement

Gentry propose de changer le schéma de chiffrement pour que le déchiffrement soit juste un produit scalaire, de faible profondeur multiplicative.

Clé publique : comme avant + des nombres en virgule fixe $y_1, \ldots, y_N$.

Clé privée : un mot $\vec{s} = (s_1, \ldots, s_n)$ de N bits avec $\alpha \ll N$ bits à 1 tel que $1/p \approx \sum_i s_i \cdot y_i$

Un chiffré de b : un entier $c = pq + 2r + b$ comme avant et des nombres en virgule fixe $z_1, \ldots, z_N$ tels que $c/p \approx \sum_i s_i \cdot z_i$

Le déchiffrement $LSB(\sum s_i \cdot z_i) \oplus LSB(c)$ peut être réalisé par un circuit avec un bruit maximal de $N \cdot \alpha \cdot \text{poly}(\log \alpha)$.

Point d’étape

Point d'étape

Vu aujourd'hui :

- Des chiffrements semi-homomorphes (1 opération) efficaces.
- Un chiffrement faiblement homomorphe ($+$ et $\times$, en nombre limité par le bruit), inefficace.
- La procédure de *bootstrap* pour obtenir (à grand peine) un chiffrement totalement homomorphe à partir d'un chiffrement faiblement homomorphe.

La semaine prochaine : comment rendre tout cela plus efficace et plus utilisable.

Bibliographie

Bibliographie

Background sur le chiffrement :

- Jean-Philippe Aumasson, *La cryptographie déchiffrée : une introduction pratique au chiffrement moderne*, Dunod, 2024. Chapitres 1, 3, 9.

Introduction au chiffrement homomorphe par bootstrap :

- Craig Gentry, *Computing Arbitrary Functions of Encrypted Data*, CACM 53(3), 2010. https://cacm.acm.org/research/computing-arbitrary-functions-of-encrypted-data/