Un bref historique

- Un premier plan par Craig Gentry ;
- FHE de secondes génération Entrée typique : polynômes de degré 1.
  - somme : la somme des polynômes
  - produit : on obtient un degré 2.
- FHE de troisième génération : BVS.

GSW, premier essai :

Clé secrète : un vecteur $sk \in \mathbb{Z}_q^N$

Clé publique : pk

Chiffrement : Encrypt(pk, μ) $=$ $C \in \mathbb{Z}_q^{N\times N}$ telle que

$$C\vec{sk} = \vec{s}$$

Déchiffrement : Evident

Opérations homomorphes :

Pour $C_i = \text{Encrypt}(\mu_i) \quad (1 \leqslant i \leqslant 2)$ et $\lambda \in \mathbb{Z}_q$

- **Somme :** $C_1 + C_2$

$$(C_1 + C_2)\,\vec{sk} = (\mu_1 + \mu_2)\,\vec{sk}$$

- **Produit :** $C_1 \times C_2$

$$(C_1 \times C_2)\ \vec{sk} = C_1\left(\mu_2\,\vec{sk}\right) = (\mu_1\mu_2)\,\vec{s}$$

- **NAND :** $C_1 * C_2 - \text{Id}$
- **Produit par scalaire :** λC_1

GSW, second essai :

Clé secrète : un vecteur $sk \in \mathbb{Z}_q^N$

Clé publique : pk

Chiffrement : Encrypt$(\mu) = C \in \mathbb{Z}_q^{N\times N}$ telle que

$$C\vec{sk} = \vec{sk} + \vec{e} \qquad \text{avec } \vec{e} \text{ petit}$$

Déchiffrement : on prend un i tel que $\vec{sk}_i$ est grand

$$\begin{aligned}\mathrm{Decrypt}(sk, C) = \left\lfloor \frac{\left(C\vec{sk}\right)_i}{v_i} \right\rceil &= \left\lfloor \frac{\left(\mu\vec{sk}_i + \vec{e}_i\right)_i}{v_i} \right\rceil \\ &= \left\lfloor \mu + \frac{\vec{e}_i}{v_i} \right\rceil \\ &= \mu\end{aligned}$$

GSW, second essai :

Retour sur les opérations homomorphes :

- **Somme :** $C_1 + C_2$

$$(C_1 + C_2)\,\vec{sk} = (\mu_1 + \mu_2)\vec{sk} + \vec{e_1} + \vec{e_2}$$

- **NAND :** $C_1 \times C_2 - \mathrm{Id}$

$$\begin{aligned}(C_1 \times C_2 - \mathrm{Id})\,\vec{sk} &= C_1\left(\mu_2\vec{sk} + \vec{e_2} - \vec{s}\right)\\ &= (\mu_1\mu_2 - 1)\vec{sk} + \mu_2\vec{e_1} + C_1\vec{e_2}\end{aligned}$$

Problème :

Les coefficients de $C_1\vec{e_2}$ peuvent être gros

GSW, version finale :

On utilise une fonction Flatten qui a notamment les propriétés suivantes :

$$C \in \mathbb{Z}_q^{n \times n} \Rightarrow \mathrm{Flatten}(C) \in \{0,1\}^{N \times N}$$
$$\langle Flatten(C), \vec{sk} \rangle = \langle C, \vec{sk} \rangle \quad pourunsecret\vec{sk}bienchoisi$$

Clé secrète : un vecteur $s \in \mathbb{Z}_q^N$ bien choisi
Clé publique : pk
Chiffrement : Encrypt(pk, μ) = Flatten(C) $\in \mathbb{Z}_q^{N \times N}$ pour C telle que

$$C\vec{sk} = \vec{sk} + \color{red}{\vec{e}} \qquad \text{avec } \color{red}{\vec{e}} \text{ petit}$$

Déchiffrement : on prend un i tel que $\vec{s}_i$ est grand et :

$$\mathrm{Decrypt}(\vec{sk}, C) = \left\lfloor \frac{(C\vec{s})_i}{v_i} \right\rceil$$

Opérations homomorphes : on applique Flatten aux précédentes

Le problème DLWE

Paramètres : $n, q \in \mathbb{N}$, une distribution χ sur $\mathbb{Z}_q$

Le problème DLWE(n, q, χ) consiste à distinguer deux distributions :

- La distribution qui crée uniformément $(\vec{a}, b) \in \mathbb{Z}_q^{n+1}$;
- La distribution qui utilise un secret $\vec{s} \in \mathbb{Z}_q^n$ tiré uniformément, et crée des vecteurs $(\vec{a}, b)$ où

$$b_i = \langle \vec{a}_i, \vec{s} \rangle + e_i$$

  e_i étant échantillonné par χ.

à partir d'un ensemble d'échantillons.

FHE avec bootstrapping

$$C = D + \text{erreur}$$

$$C = D + \text{error} \xrightarrow{\text{Decrypt}(\text{sk}, C)} \mu \xrightarrow{\text{Encrypt}(\text{sk}, \mu)} C^{\text{new}} = D + \text{error}$$

FHE avec bootstrapping

$$C = D + \text{erreur}$$

$$C = D + \text{error} \xrightarrow{\text{Decrypt}(\text{sk}, C)} \mu \xrightarrow{\text{Encrypt}(\text{sk}, \mu)} C^{\text{new}} = D + \text{error}$$

Soit Π le circuit booléen tel que

$$\Pi(\text{binsk}) = \text{Decrypt}\left(\text{sk}, C\right)$$

FHE avec bootstrapping

$$C = D + \text{erreur}$$

$$C = D + \text{error} \xrightarrow{\text{Decrypt}(\text{sk}, C)} \mu \xrightarrow{\text{Encrypt}(\text{sk}, \mu)} C^{\text{new}} = D + \text{error}$$

Soit Π le circuit booléen tel que

$$\Pi(\text{binsk}) = \text{Decrypt}\,(\text{sk}, C)$$

Alors :

$$\begin{aligned} \text{Encrypt}\,(\text{sk},\ \text{Decrypt}\,(\text{sk}, C)) =& \text{Encrypt}\,(\text{sk},\ \Pi(\text{binsk})) \\ =& \text{Eval}\,(\Pi, \text{Encrypt}(\text{sk}, \text{binsk})) \end{aligned}$$

- Si Π contient assez peu de NAND, on peut avoir un FHE.

Découpage de Decrypt

L'algorithme de déchiffrement est le suivant :

1. trouver $1 \leqslant i \leqslant l$ tel que $q/4 \leqslant 2^i < q/2$
2. calculer $a = C_i \cdot \vec{v}$
3. retourner $\lfloor \frac{a}{\vec{v}_i} \rceil$

- On peut ramener le calcul du produit scalaire à une somme de nombres binaire ;
- Diviser par une puissance de 2 est un shift à gauche sur l'écriture binaire ;
- Calculer la valeur valeur absolue implique essentiellement de faire un complément à 2.

Voyons comment sommer deux nombres binaires.

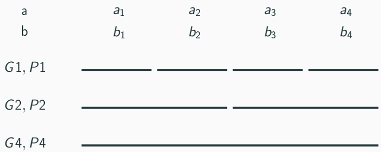
sommer deux listes :

Somme classique entre deux nombres binaires :

$$
\begin{array}{ccccc}
 & a_1 \overset{c_1}{\curvearrowright} & a_2 \overset{c_2}{\curvearrowright} & \cdots \overset{c_{s-1}}{\curvearrowright} & a_s \\
+ & b_1 & b_2 & \cdots & b_s \\
\hline
 & r_1 & r_2 & \cdots & r_s
\end{array}
$$

- a_1 et b_1 présents dans la formule booléenne de r_s.
- profondeur de NAND en $O(s)$

sommer deux listes : carry lookahead adder



sommer deux listes : carry lookahead adder

a	a_1	a_2	a_3	a_4
b	b_1	b_2	b_3	b_4
$G1, P1$				
$G2, P2$				
$G4, P4$				

- G pour génération

$$
\begin{array}{cccccc}
 & a_5 & a_6 & a_7 & a_8 & \\
+ & b_5 & b_6 & b_7 & b_8 & \\
\hline
 & X & X & X & X & 1
\end{array}
$$

- P pour propagation

$$
\begin{array}{cccccc}
 & a_5 & a_6 & a_7 & a_8 & \\
+ & b_5 & b_6 & b_7 & b_8 & \\
+ & 1 & & & & \\
\hline
 & X & X & X & X & 1
\end{array}
$$

sommer deux listes : carry lookahead adder

a	a_1	a_2	a_3	a_4
b	b_1	b_2	b_3	b_4
$G1, P1$				
$G2, P2$				
$G4, P4$				

- G pour génération $\qquad (G1)_i = a_i \wedge b_i \quad (P1)_i = a_i \vee b_i$

$$\begin{array}{cccccc} & a_5 & a_6 & a_7 & a_8 & \\ + & b_5 & b_6 & b_7 & b_8 & \\ \hline & X & X & X & X & 1 \end{array}$$

- P pour propagation

$$\begin{array}{cccccc} & a_5 & a_6 & a_7 & a_8 & \\ + & b_5 & b_6 & b_7 & b_8 & \\ + & 1 & & & & \\ \hline & X & X & X & X & 1 \end{array}$$

a	a_1	a_2	a_3	a_4
b	b_1	b_2	b_3	b_4
$G1, P1$				
$G2, P2$				
$G4, P4$				

- G pour génération

$$\begin{array}{cccccc} & a_5 & a_6 & a_7 & a_8 & \\ + & b_5 & b_6 & b_7 & b_8 & \\ \hline & X & X & X & X & 1 \end{array}$$

- P pour propagation

$$\begin{array}{cccccc} & a_5 & a_6 & a_7 & a_8 & \\ + & b_5 & b_6 & b_7 & b_8 & \\ + & 1 & & & & \\ \hline & X & X & X & X & 1 \end{array}$$

$$(G1)_i = a_i \wedge b_i \quad (P1)_i = a_i \vee b_i$$

$G2^i, P2^i$ 1

$G2^{i+1}, P2^{i+1}$ 1 2

$$(G2^{i+1})_1 = (G2^i)_2 \vee \left((G2^i)_1 \wedge (P2^i)_2\right)$$

$$(P2^{i+1})_1 = (G2^i)_1 \wedge (P2^i)_2$$

a	0	0	1	1	1	1	0	0
b	0	1	0	1	0	1	0	1
$G1$, $P1$	0 0	0 1	0 1	1 1	0 1	1 1	0 0	0 1
$G2$, $P2$	0 0		1 1		1 1		0 0	
$G4$, $P4$	1 0				0 0			
$G8$, $P8$	0 0							
carry	?	?	?	?	?	?	?	?

Les variables de générations de blocs commençants par 0 calculent des retenues.

sommer deux listes : carry lookahead added

a	0	0	1	1	1	1	0	0
b	0	1	0	1	0	1	0	1
$G1, P1$	0 0	0 1	0 1	1 1	0 1	1 1	0 0	0 1
$G2, P2$	0 0		1 1		1 1		0 0	
$G4, P4$	1 0				0 0			
$G8, P8$	0 0							
carry	0	0	?	1	?	?	?	0

Les variables de générations de blocs commençants par 0 calculent des retenues.

$$c_6 = G2_3 \vee (c_4 \wedge P2_3) = 1 \vee (1 \wedge 1) = 1$$

sommer deux listes : carry lookahead added

a	0	0	1	1	1	1	0	0
b	0	1	0	1	0	1	0	1
$G1, P1$	0 0	0 1	0 1	1 1	0 1	1 1	0 0	0 1
$G2, P2$	0 0		1 1		1 1		0 0	
$G4, P4$	1 0				0 0			
$G8, P8$	0 0							
carry	0	0	?	1	?	1	?	0

Les variables de générations de blocs commençants par 0 calculent des retenues.

$$c_6 = G2_3 \vee (c_4 \wedge P2_3) = 1 \vee (1 \wedge 1) = 1$$

$$c_7 = G1_7 \vee (c_6 \wedge P1_7) = 0 \vee (1 \wedge 0) = 0$$

Profondeur de NAND de l'algorithme de déchiffrement

Paramètres avec bootstrapping obtenus, taille

Paramètres avec bootstrapping obtenus, taille

Des librairies pour faire du FHE

Ce dont nous n'avons pas parlé