Le calcul sécurisé, quatrième cours

Calcul multipartite sécurisé : partager des secrets

Xavier Leroy

2025-11-27

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

Le calcul multipartite sécurisé

Calcul multipartite sécurisé (rappel)

(Aussi connu comme *Secure Multi-Party Computation*, MPC.)

Calculer avec des données secrètes provenant de n participants :

- Chaque participant i a un secret x_i.
- Ils coopèrent pour calculer $y = F(x_1, \ldots, x_n)$.
- Le résultat y est révélé à tous.
- Chaque participant i n'apprend rien sur x_j ($j \neq i$) qu'il ne peut déduire de y et de x_i.

Exemple : dépouillement d'un appel d'offre (rappel)

Avec un tiers de confiance :

- Chaque participant i envoie son offre x_i au tiers.
- Le tiers détermine j t.q. $x_j = \min(x_1, \ldots, x_n)$ et l'annonce.
- Les participants apprennent que l'offre de j est minimale.
- Les participants n'apprennent rien d'autre sur les montants des offres des autres participants.

Exemple : dépouillement d'un appel d'offre (rappel)

Avec un tiers de confiance :

- Chaque participant i envoie son offre x_i au tiers.
- Le tiers détermine j t.q. $x_j = \min(x_1, \ldots, x_n)$ et l'annonce.
- Les participants apprennent que l'offre de j est minimale.
- Les participants n'apprennent rien d'autre sur les montants des offres des autres participants.

Peut-on réaliser ce calcul de manière distribuée entre les participants sans recourir à un tiers de confiance et sans révéler les secrets x_i à personne ?

Chiffrement homomorphe *vs.* calcul multipartite sécurisé

	Chiffrement homomorphe	Calcul multipartite
Paradigme	calcul délégué (*cloud*)	calcul distribué
Participants	1 client, 1 calculateur	n participants
Secrets	tous détenus par le client	chaque participant a les siens
Résultats	connus du seul client	connus de tous les participants
Puissance de calcul	calculateur $\gg$ client	$\approx$ la même pour tous
Communications	peu	beaucoup
Protocoles	non-interactifs	interactifs

Critères de correction et de sécurité

Correction : le résultat du calcul est le bon si tous les participants suivent le protocole.

Sécurité passive : une collusion de $\leq A$ participants ne peut rien apprendre des secrets des autres participants.

Sécurité active (détection d'erreur) : si $\leq A$ participants ne suivent pas le protocole, les autres peuvent le détecter et arrêter le calcul.

Sécurité active (correction d'erreur) : si $\leq A$ participants ne suivent pas le protocole, les autres peuvent ignorer leurs actions et continuer le calcul.

(Note : on suppose que les communications entre participants sont chiffrées et authentifiées $\Rightarrow$ les seuls attaquants possibles sont les participants.)

Partage additif de bits : le protocole GMW

Partager un bit

Comment partager un bit secret b entre deux participants ?

- Tirer un bit r au hasard.
- Donner $b_1 = r$ à un participants et $b_2 = b \oplus r$ à l'autre.

Chaque participant n'apprend rien sur le bit b
(principe du masque jetable : $b \oplus r$ est aussi aléatoire que r).

Quand les deux participants se mettent d'accord pour révéler le bit b, ils échangent leurs bits b_1, b_2 et retrouvent b en calculant

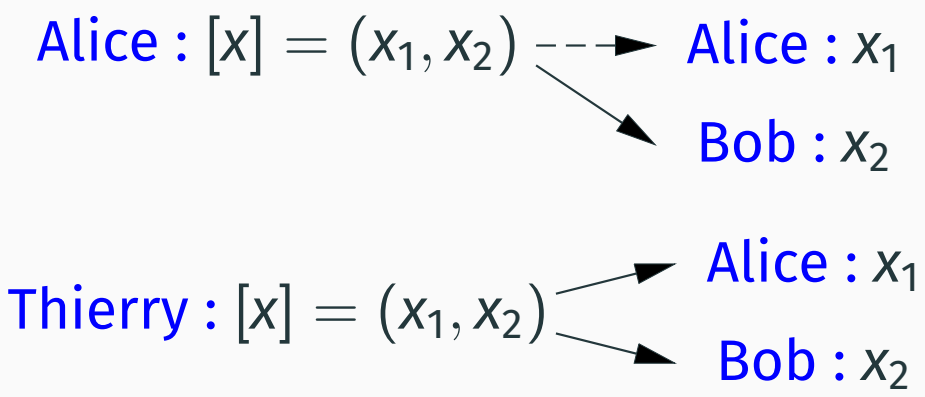
$$b_1 \oplus b_2 = r \oplus b \oplus r = (r \oplus r) \oplus b = b$$

On note $[b]$ un tel partage d'un bit b :

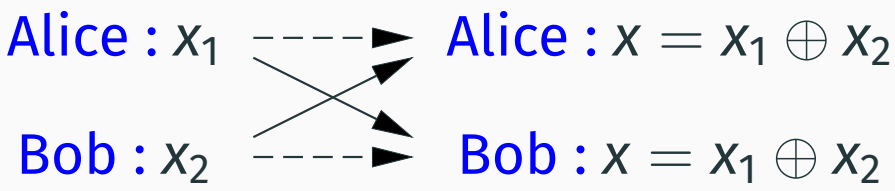
$$[b] = (b_1, b_2) \quad \text{tels que} \quad b = b_1 \oplus b_2$$

Partager puis révéler

Partager un secret (d'un des participants, ou d'un tiers) :



Révéler un secret partagé :



Le protocole GMW (Goldreich, Micali, Wigderson, 1987)

Évaluation bipartite sécurisée de

$$z = F(\mathbf{x}, \mathbf{y})$$

F : un circuit booléen
$\mathbf{x}$: les secrets d'Alice
$\mathbf{y}$: les secrets de Bob

- Entrées : Alice tire des partages $[\mathbf{x}]$, Bob des partages $[\mathbf{y}]$, ils les mettent en commun.
- Calcul bipartite : Alice calcule z_1 et Bob calcule z_2, où (z_1, z_2) est un partage de z.
  (Alice et Bob peuvent communiquer pendant ce calcul.)
- Sortie : Alice et Bob révèlent z_1, z_2 et déterminent $z = z_1 \oplus z_2$.

Additionner deux bits partagés (porte XOR)

On a deux bits partagés, $[x] = (x_1, x_2)$ et $[y] = (y_1, y_2)$.

Alice connaît x_1 et y_1, et calcule $z_1 = x_1 \oplus y_1$.

Bob connaît x_2 et y_2, et calcule $z_2 = x_2 \oplus y_2$.

La paire (z_1, z_2) est bien un partage de $x \oplus y$:

$$z_1 \oplus z_2 = (x_1 \oplus y_1) \oplus (x_2 \oplus y_2) = (x_1 \oplus x_2) \oplus (y_1 \oplus y_2) = x \oplus y$$

Le calcul est local (pas de communication entre participants).

Négation d'un bit partagé (porte NOT)

Si on a un bit partagé $[x] = (x_1, x_2)$:

Alice connaît x_1 et calcule $z_1 = x_1 \oplus 1 = \neg x_1$.

Bob connaît x_2 et prend $z_2 = x_2$.

La paire (z_1, z_2) est bien un partage de $\neg x$:

$$z_1 \oplus z_2 = (x_1 \oplus 1) \oplus x_2 = (x_1 \oplus x_2) \oplus 1 = x \oplus 1 = \neg x$$

Multiplier deux bits partagés (portes AND, OR)

On a deux bits partagés, $[x] = (x_1, x_2)$ et $[y] = (y_1, y_2)$.

On souhaite calculer un partage de $x \wedge y = x \cdot y$
ou de $x \vee y = \neg(\neg x \cdot \neg y)$.

Aucun calcul purement local ne convient. En particulier,
$(x_1 \cdot y_1, x_2 \cdot y_2)$ n'est pas un partage de $x \cdot y$:

$$(x_1 \oplus x_2) \cdot (y_1 \oplus y_2) = x_1 \cdot y_1 \oplus x_1 \cdot y_2 \oplus x_2 \cdot y_1 \oplus x_2 \cdot y_2 \neq x_1 \cdot y_1 \oplus x_2 \cdot y_2$$

Goldreich, Micali, Wigderson (STOC 1987) proposent d'utiliser un
transfert inconscient 1 sur 4.

Le transfert inconscient (*Oblivious Transfer*, OT)

Un protocole entre deux participants :

- Alice connaît n valeurs $v_1, \ldots, v_n$
- Bob choisit $i \in \{1, \ldots, n\}$

À la fin du protocole,

- Bob connaît la valeur v_i.
- Alice ne connaît pas le choix i de Bob.
- Bob n'a rien appris sur les autres valeurs v_j, $j \neq i$.

(Plus de détails dans le 5e cours.)

Multiplication par transfert inconscient

Calcul d'un partage (z_1, z_2) de $x \cdot y$:

Alice choisit z_1 au hasard et tabule la valeur que doit avoir z_2 en fonction des valeurs possibles des inconnues x_2, y_2 :

$$z_2 = z_1 \oplus x \cdot y = z_1 \oplus (x_1 \oplus x_2) \cdot (y_1 \oplus y_2)$$

Sous forme de table :

ligne	x_2	y_2	z_2
0	0	0	$z_1 \oplus (x_1 \cdot y_1)$
1	0	1	$z_1 \oplus (x_1 \cdot \neg y_1)$
2	1	0	$z_1 \oplus (\neg x_1 \cdot y_1)$
3	1	1	$z_1 \oplus (\neg x_1 \cdot \neg y_1)$

Multiplication par transfert inconscient

ligne	x_2	y_2	z_2
0	0	0	$z_1 \oplus (x_1 \cdot y_1)$
1	0	1	$z_1 \oplus (x_1 \cdot \neg y_1)$
2	1	0	$z_1 \oplus (\neg x_1 \cdot y_1)$
3	1	1	$z_1 \oplus (\neg x_1 \cdot \neg y_1)$

Transfert inconscient : Bob demande la ligne numéro $2x_2 + y_2$ correspondant à ses parts x_2, y_2, et reçoit le z_2 correspondant.

On a bien $z_1 \oplus z_2 = x \cdot y$.

Alice ne sait pas quelle ligne Bob a choisi, et n'apprend rien sur x_2, y_2.

Bob n'apprend rien sur les autres lignes, et ne peut donc rien déduire sur x_1, y_1.

Multiplication avec triplets de Beaver

(D. Beaver, *Efficient Multiparty Protocols Using Circuit Randomization*, CRYPTO 1991.)

On prépare à l'avance des triplets de Beaver :
des bits partagés $[a], [b], [c]$ aléatoires tels que $c = a \cdot b$.

Alice a les parties a_1, b_1, c_1 de ces triplets.
Bob a les parties a_2, b_2, c_2.

On peut produire à l'avance de tels triplets, par transfert inconscient entre les deux participants, ou via un tiers de confiance, ou par d'autres techniques cryptographiques.

(Communications ≪offline≫ avant le début du calcul et non plus ≪online≫ comme avec le protocole OT utilisé par GMW.)

Multiplication avec triplets de Beaver

Calcul d'un partage (z_1, z_2) de $x \cdot y$:

Alice et Bob prennent le prochain triplet a, b, c sur la liste.

Alice envoie à Bob $a_1 \oplus x_1$ et $b_1 \oplus y_1$
(ses parts de x et y masquées par a et b)

Bob envoie à Alice $a_2 \oplus x_2$ et $b_2 \oplus y_2$ (même masquage)

Alice et Bob connaissent maintenant $d = a \oplus x$ et $e = b \oplus y$.

Alice calcule z_1 et Bob calcule z_2 comme suit :

$$z_i = d \cdot y_i \oplus a_i \cdot e \oplus c_i$$

(z_1, z_2) est un partage de $x \cdot y$, car

$$\begin{aligned} z_1 \oplus z_2 &= a \cdot y \oplus x \cdot y \oplus a \cdot b \oplus a \cdot y \oplus c \\ &= x \cdot y \oplus (a \cdot b \oplus c) = x \cdot y \quad \text{puisque } c = a \cdot b \end{aligned}$$

Extension à n $>$ 2 participants

On peut partager un bit b entre $n > 2$ participants :

$$[b] = (b_1, \ldots, b_n) \quad \text{avec} \quad b = b_1 \oplus \cdots b_n$$

Si le participant 1 souhaite partager le secret x, il tire $b_2, \ldots, b_n$ au hasard, envoie b_i au participant i, et conserve $b_1 = x \oplus b_2 \oplus \cdots \oplus b_n$.

Pour révéler le partage $[b] = (b_1, \ldots, b_n)$, chaque participant i envoie sa part b_i aux $n - 1$ autres participants.
Tous les participants connaissent alors $b = b_1 \oplus \cdots \oplus b_n$.

Sécurité du protocole GMW

En supposant la sécurité du protocole OT utilisé.

- Sécurité passive : le seul moyen de retrouver un bit partagé b est que les n participants divulguent leurs parts $b_1, \ldots, b_n$. Une collusion de $A < n$ participants n'apprend rien sur b.
- Sécurité active : aucune. Si un participant produit un b_i faux, le résultat du calcul est faux, et c'est indétectable.
- Robustesse : aucune. Si un participant tombe en panne, le résultat du calcul est perdu.

Partages k parmi n

Partage répliqué 2 parmi 3

Un exemple de partage redondant entre 3 participants.
Chaque participant détient 2 parts sur les 3 du secret.

$$b = b_1 \oplus b_2 \oplus b_3$$

Alice détient b_1 et b_2
Bob détient b_2 et b_3
Charlie détient b_3 et b_1

Il suffit que deux participants mettent en commun leurs parts pour retrouver le secret.

Robustesse : résiste à la panne d'un participant sur les 3.

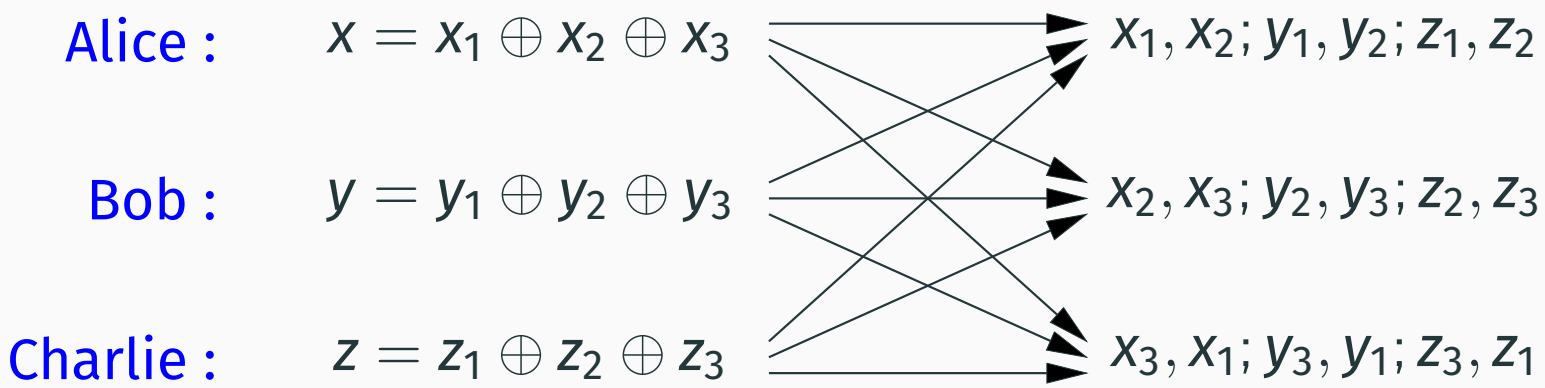
Sécurité passive : un seul participant n'apprend rien sur le secret.

Sécurité active : on peut détecter si un participant produit des résultats faux, mais pas si deux le font.

Partage de secret

Pour partager son secret x, Alice tire x_2, x_3 au hasard, pose $x_1 = x \oplus x_2 \oplus x_3$, et envoie les bons x_i à Bob et Charlie.

Bob peut faire de même avec y et Charlie avec z.



Addition

Les participants additionnent leurs parts point à point :

$$\begin{array}{llll} \text{Alice} & x_1, x_2 & y_1, y_2 & \to & x_1 \oplus y_1, x_2 \oplus y_2 \\ \text{Bob} & x_2, x_3 & y_2, y_3 & \to & x_2 \oplus y_2, x_3 \oplus y_3 \\ \text{Charlie} & x_3, x_1 & y_3, y_1 & \to & x_3 \oplus y_3, x_1 \oplus y_1 \end{array}$$

Si $x = x_1 \oplus x_2 \oplus x_3$ et $y = y_1 \oplus y_2 \oplus y_3$, le résultat est bien un partage redondant de $x \oplus y$.

Multiplication

Les participants combinent leurs parts comme suit :

$$\begin{array}{llll} \text{Alice} & x_1, x_2 & y_1, y_2 & \to \quad p = x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \\ \text{Bob} & x_2, x_3 & y_2, y_3 & \to \quad q = x_2y_2 \oplus x_2y_3 \oplus x_3y_2 \\ \text{Charlie} & x_3, x_1 & y_3, y_1 & \to \quad r = x_3y_3 \oplus x_3y_1 \oplus x_1y_3 \end{array}$$

Alice tire un partage $[p]$ de p, le partage avec Bob et Charlie.
Bob tire un partage $[q]$ de q, le partage avec Alice et Charlie.
Charlie tire un partage $[r]$ de r, le partage avec Alice et Bob.

Les 3 participants calculent un partage de $p \oplus q \oplus r$ par addition locale. C'est un partage de xy, puisque

$$\begin{aligned} xy &= (x_1 \oplus x_2 \oplus x_3)(y_1 \oplus y_2 \oplus y_3) \\ &= x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2 \oplus x_2y_3 \oplus x_3y_2 \oplus x_3y_3 \oplus x_3y_1 \oplus x_1y_3 \\ &= p \oplus q \oplus r \end{aligned}$$

Partage de Shamir

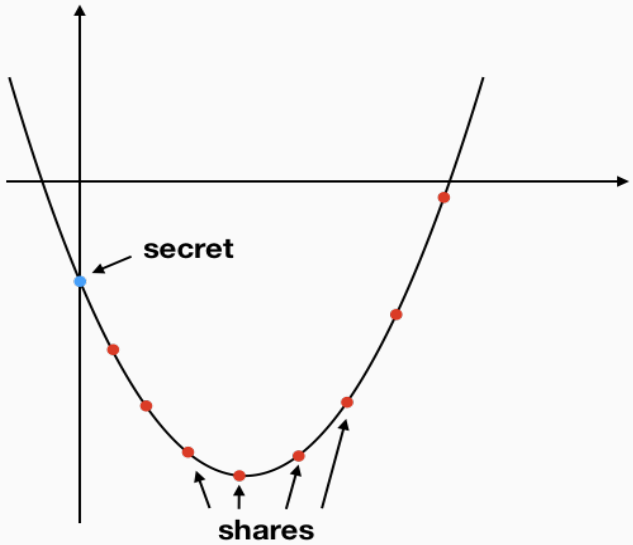
(A. Shamir, *How to share a secret*, CACM 22(11), 1979.)

Les secrets et les parts sont des éléments d'un corps fini $\mathbb{F}_q$ d'ordre $q > n$ (nombre de participants).

Partager un secret x :

Choisir un polynôme P de degré $t < n$ avec le coefficient constant égal à x et les autres coefficients aléatoires dans $\mathbb{F}_q$.

Les parts sont $x_i = P(i)$ pour $i = 1, \ldots, n$.



(Cf. les codes de Reed-Solomon.)

Partage de Shamir

$$[x] = (P(1), \ldots, P(n)) \quad \text{avec } \deg(P) = t \text{ et } P(0) = x$$

Retrouver le secret x à partir de $t+1$ parts :

Connaître $t+1$ parts, c'est connaître $t+1$ points $(x_0, y_0), \ldots, (x_t, y_t)$ sur la courbe de P.

P étant de degré t, ces $t+1$ points déterminent P entièrement.

Le secret x est $P(0)$.

Formule d'interpolation de Lagrange :

$$x = P(0) = \sum_{j=0}^{t} y_j \lambda_j \quad \text{avec} \quad \lambda_j = \prod_{k=0, k\neq j}^{t} \frac{x_k}{x_k - x_j}$$

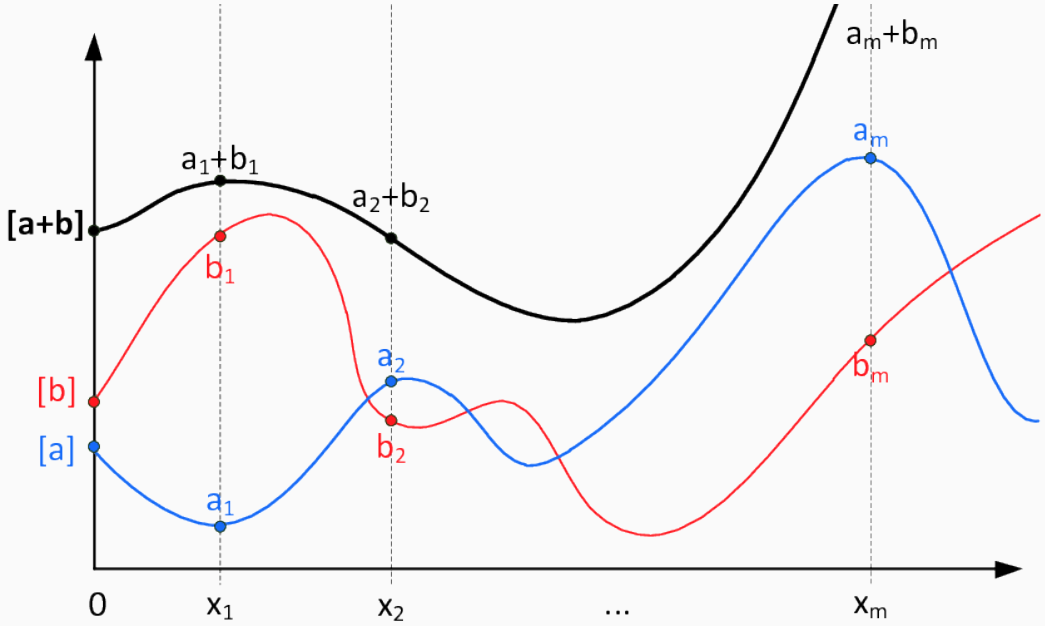
Partage de Shamir

Robustesse : $t + 1$ parts parmi n suffisent pour reconstruire le secret.

Sécurité passive : une collusion d'au plus t participants n'apprend rien sur le secret.
(Si $P(i)$ est connu pour t points $i \neq 0$, $P(0)$ peut avoir n'importe quelle valeur.)

Sécurité active : comme avec les codes de Reed-Solomon, on peut détecter jusqu'à $n - t - 1$ erreurs et corriger jusqu'à $(n - t - 1)/2$ erreurs.

Addition de deux partages de Shamir



Soient $[a] = (a_1, \ldots, a_n)$ et $[b] = (b_1, \ldots, b_n)$ des partages de Shamir pour les secrets a et b.

Alors $(a_1 + b_1, \ldots, a_n + b_n)$ est un partage pour $a + b$.

Il peut être calculé localement par chacun des n participants.

Addition et multiplication par une constante

Si $[a] = (a_1, \ldots, a_n)$ est un partage de a (polynôme P) :

- $(a_1 + k, \ldots, a_n + k)$ est un partage de $a + k$ (polynôme $P + k$).
- $(ka_1, \ldots, ka_n)$ est un partage de ka (polynôme kP).

(Calcul local.)

Multiplication de deux partages de Shamir

Soient $[a] = (a_1, \ldots, a_n)$ et $[b] = (b_1, \ldots, b_n)$ des partages de Shamir pour les secrets a et b :

$$a = P(0) \quad a_i = P(i) \quad b = Q(0) \quad b_i = Q(i)$$

où P et Q sont des polynômes de degré t.

Les points $(i, a_i b_i)$ sont sur la courbe du polynôme PQ.

Mais PQ est de degré $2t$, donc t points ne suffisent pas à déterminer $PQ(0) = ab$.

Et donc $(a_1 b_1, \ldots, a_n b_n)$ n'est pas un partage de ab.

Multiplication de deux partages de Shamir

Supposons $t < n/2$. Chacun des $2t$ premiers participants prépare un partage $[a_i b_i]$ du produit de ses deux coefficients a_i et b_i, et le transmet aux autres participants.

On a donc des polynômes aléatoires $R_1, \ldots, R_{2t}$ de degré t avec

$$R_i(0) = a_i b_i \qquad \text{le participant } j \text{ connaît } R_i(j)$$

Les n participants reconstruisent (localement) un partage $(c_1, \ldots, c_n)$ par la formule d'interpolation de Lagrange :

$$c_j = \sum_{i=1}^{2t} R_i(j)\lambda_i \quad \text{avec} \quad \lambda_i = \prod_{k=1, k\neq i}^{2t} \frac{k}{k-i}$$

Multiplication de deux partages de Shamir

$$R_i(0) = a_i b_i \qquad \deg(R_i) = t$$

$$c_j = \sum_{i=1}^{2t} R_i(j)\lambda_i \quad \text{avec} \quad \lambda_i = \prod_{k=1, k\neq i}^{2t} \frac{k}{k-i}$$

Soit $R = \sum_{i=1}^{2t} R_i \lambda_i$. On a

$$\deg R = t \qquad c_j = R(j)$$

$$R(0) = \sum_{i=1}^{2t} a_i b_i \lambda_i = \sum_{i=1}^{2t} PQ(i)\lambda_i = PQ(0) = ab$$

Donc, $(c_1, \ldots, c_n)$ est un partage de ab via le polynôme R.

Alternative : utilisation de triplets de Beaver

On suppose que les participants ont reçu à l'avance un partage $[u], [v], [w]$ avec u, v aléatoires et $w = uv$.

Pour calculer le produit ab :

Les participants calculent localement $[a + u]$ et $[b + v]$,
et révèlent ces partages.

Tous les participant connaissent alors $\alpha = a + u$ et $\beta = b + v$
(les arguments secrets a, b masqués par u, v aléatoires).

Les participants calculent localement

$$[c] = [w] + \alpha[b] - \beta[u]$$

C'est un partage du produit ab, car

$$c = uv + ab + ub - bu - vu = ab$$

Généralisation : partages linéaires de secrets

Un partage linéaire de secrets (*Linear Secret Sharing Scheme*, LSSS)

Défini par

- une matrice $\mathbf{M}$ de dimensions $m \times d$
- un vecteur $\mathbf{v}$ de dimension d
- une fonction surjective $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ (colonne $\mapsto$ participant).

Pour partager le secret s, on tire un vecteur $\mathbf{k}$ aléatoire tel que

$$s = \langle \mathbf{v}, \mathbf{k} \rangle$$

puis on calcule m parts $s_1, \ldots, s_m$ en appliquant la matrice $\mathbf{M}$

$$\mathbf{M} \cdot \mathbf{k} = (s_1, \ldots, s_m)^\mathsf{T}$$

et on donne la part s_i au participant numéro $\varphi(i)$.

Partage additif complet vu comme un LSSS trivial

Dimensions $m = d = n$ (nombre de participants).

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \qquad \mathbf{v} = (1, 1, \ldots, 1)$$

$$\varphi(i) = i$$

Pour partager s, on choisit $\mathbf{k} = (k_1, \ldots, k_n)$ tel que

$$s = \langle \mathbf{v}, \mathbf{k} \rangle = k_1 + \cdots + k_n$$

On distribue la part $s_i = k_i$ au participant numéro $\varphi(i) = i$.

Partage de Shamir vu comme un LSSS

Dimensions $m = t + 1$ et $d = n$

(t degré des polynômes, n nombre de participants).

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & \cdots & 1^t \\ 1 & 2 & \cdots & 2^t \\ \vdots & \vdots & & \vdots \\ 1 & n & \cdots & n^t \end{pmatrix} \qquad \mathbf{v} = (1, 0, \ldots, 0)$$

$$\varphi(i) = i$$

Pour partager s, on choisit $\mathbf{k} = (s, k_1, \ldots, k_t)$ avec k_i aléatoire.

$\mathbf{k}$ sont les coefficients d'un polynôme P. On a

$$s = \langle \mathbf{v}, \mathbf{k} \rangle \qquad \mathbf{M} \cdot \mathbf{k} = (P(1), \ldots, P(n))^\mathsf{T}$$

On distribue la i-ème part $P(i)$ au participant numéro $\varphi(i) = i$.

Partage répliqué 2 parmi 3 vu comme un LSSS

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad \mathbf{v} = (1, 1, 1)$$

$$\varphi(i) = \lceil i/2 \rceil$$

Pour partager s, on écrit $s = \langle \mathbf{v}, \mathbf{k} \rangle = k_1 + k_2 + k_3$ avec k_i aléatoires.

Les parts sont $\mathbf{M} \cdot \mathbf{k} = (k_2, k_3, k_1, k_3, k_1, k_2)^\mathsf{T}$.

On donne les deux premières à P1, les deux suivantes à P2, les deux dernières à P3.

Révéler le secret à partir d'un partage LSSS

$$s = \langle \mathbf{v}, \mathbf{k} \rangle = \mathbf{v}^\intercal \cdot \mathbf{k} \qquad \mathbf{M} \cdot \mathbf{k} = (s_1, \ldots, s_m)^\intercal$$

On peut retrouver le secret s à partir des parts s_i s'il existe une combinaison linéaire des lignes de $\mathbf{M}$ qui est égale à $\mathbf{v}$, c.à.d. un vecteur $\mathbf{x}$ de dimension d tel que

$$\mathbf{M}^\intercal \cdot \mathbf{x} = \mathbf{v}$$

Alors,

$$\langle \mathbf{x}, s_1, \ldots, s_m \rangle = \mathbf{x}^\intercal \cdot \mathbf{M} \cdot \mathbf{k} = (\mathbf{M}^\intercal \cdot \mathbf{x})^\intercal \cdot \mathbf{k} = \mathbf{v}^\intercal \cdot \mathbf{k} = s$$

Révéler le secret à partir d'un partage LSSS

$$\mathbf{M}^\mathsf{T} \cdot \mathbf{x} = \mathbf{v}$$

Si le codage est redondant, plusieurs vecteurs $\mathbf{x}$ sont possibles, et les zéros dans $\mathbf{x}$ correspondent à des parts qui ne sont pas nécessaires pour retrouver s.

Exemple : pour le partage de Shamir avec $t = 2$ et $n = 4$, on a quatre $\mathbf{x}$ possibles qui ont un coefficient nul :

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}$$

$$\mathbf{v} = (1, 0, 0)$$

$$\mathbf{x}_1 = (0, 6, -8, 3)$$
$$\mathbf{x}_2 = (2, 0, -2, 1)$$
$$\mathbf{x}_3 = (8/3, -2, 0, 1/3)$$
$$\mathbf{x}_4 = (3, -3, 1, 0)$$

Arithmétique sur les partages LSSS

Addition :

- addition locale de chaque part.

Multiplication par une constante :

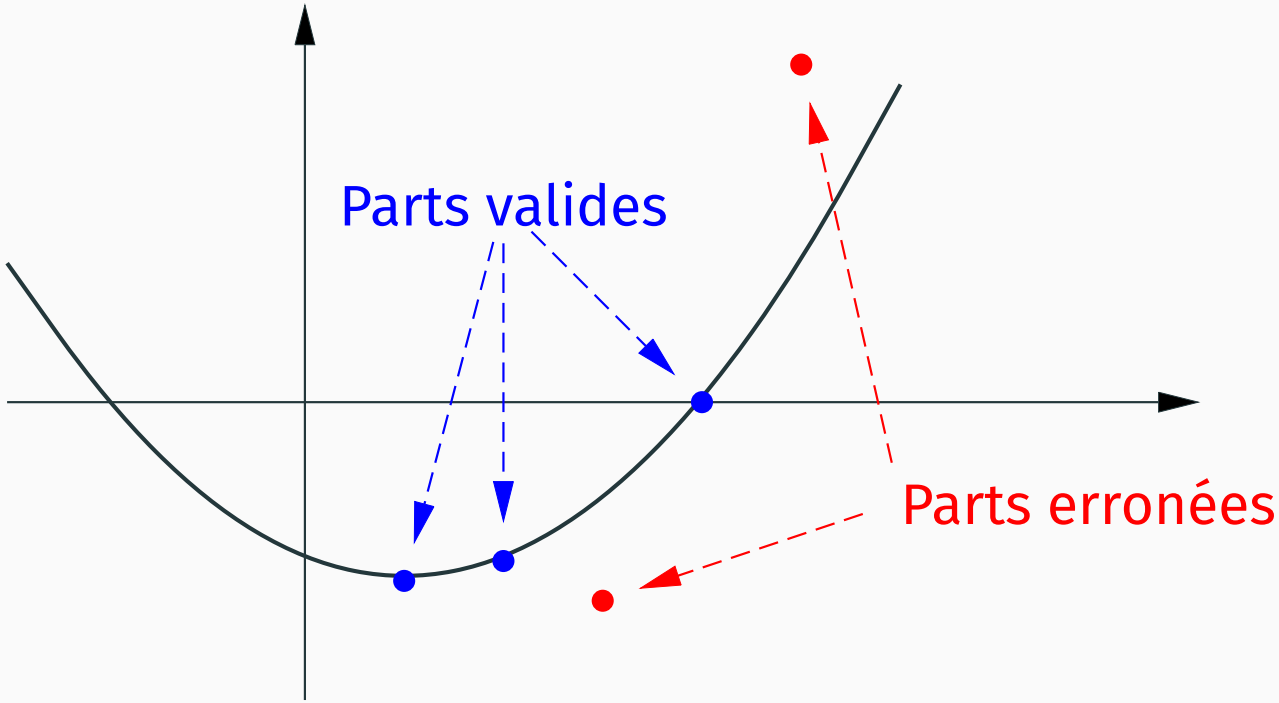
- multiplication locale de chaque part.

Multiplication générale :

- Produit de Schur (combinaison linéaire de produits locaux) si le LSSS s'y prête (Shamir avec $2t < n$; réplication 2-3).
- Triplets de Beaver.
- Produit de Damgård-Nielsen (non traité ici).

Résistance aux attaques actives

Détection d'erreur dans les partages de Shamir



S'il n'existe pas de polynôme P de degré t qui passe par les points $(1, a_1), \ldots, (n, a_n)$, le partage $(a_1, \ldots, a_n)$ est erroné.

Une ou plusieurs parts a_i ont été corrompues, peut-être par des participants malveillants (attaque active).

Détection et correction d'erreur

Un code (n, t) de Reed-Solomon / un partage (n, t) de Shamir peut :

- Détecter jusqu'à $n - t - 1$ erreurs.

  Juste en appliquant la matrice de parité H au partage :

  $$H \cdot (a_1, \ldots, a_n)^\mathsf{T} \neq \mathbf{0} \implies \text{le partage } (a_1, \ldots, a_n) \text{ est erroné}$$

- Corriger jusqu'à $(n - t - 1)/2$ erreurs.

  Naïvement : en trouvant un sous-ensemble de $n - (n - t - 1)/2$ parts qui ne contient pas d'erreurs.

  Efficacement : algorithme de Berlekamp-Welch.

Sécurité active

A participants malveillants, pas assez nombreux pour révéler les secrets ($A \leq t$), mais qui ne suivent pas le protocole.

Hypothèse : ces participants malveillants ne peuvent modifier que leurs parts des partages, mais pas celles des participants honnêtes.

Détection de l'attaque :

Si $A \leq n - t - 1$, l'attaque peut être détectée au moment où les n participants révèlent leurs parts $(a_1, \ldots, a_n)$.

Les participants honnêtes calculent $H \cdot (a_1, \ldots, a_n)^\intercal \neq \mathbf{0}$.

Peut toujours être réalisé tant que $A < n/2$ (en prenant $t \approx n/2$).

Sécurité active

A participants malveillants, pas assez nombreux pour révéler les secrets ($A \leq t$), mais qui ne suivent pas le protocole.

Hypothèse : ces participants malveillants ne peuvent modifier que leurs parts des partages, mais pas celles des participants honnêtes.

Neutralisation de l'attaque :

Si $A \leq (n - t - 1)/2$, même détection de l'attaque. De plus, les participants honnêtes peuvent identifier les attaquants et continuer le calcul en les ignorant.

Peut toujours être réalisé tant que $A < n/3$ (en prenant $t \approx n/3$).

Sécurité active des opérations sur les partages

L'hypothèse

> *Les participants malveillants ne peuvent modifier que leurs parts des partages, mais pas celles des participants honnêtes*

est vraie pour les opérations locales (addition, multiplication par une constante) où chaque participant modifie uniquement sa part.

Elle n'est pas vraie lorsqu'un participant prépare et diffuse un partage $[x]$ d'une valeur x qu'il choisit.

Sécurité active de la multiplication

Exemple : la multiplication de partages de Shamir par produits de Schur.

> *Chacun des 2t premiers participants prépare un partage $[a_i b_i]$ du produit de ses deux coefficients a_i et b_i et le transmet aux autres participants.*

Si un de ces participants ment sur sa valeur de $a_i b_i$, le résultat de la multiplication est faux, mais sans erreurs de codage.

Sécurité active de la multiplication de Beaver

La multiplication de Beaver ne fait intervenir que des opérations «vérifiables» (addition, multiplication par une constante, révélation) :

> Calculer localement $[a + u]$ et $[b + v]$
> Révéler ces partages, donnant $\alpha = a + u$ et $\beta = b + v$ à tous
> Calculer localement $[c] = [w] + \alpha[b] - \beta[u]$

Cependant, les attaquants ont pu fausser le triplet de Beaver : les partages $[a], [b], [c]$ sont bien formés mais $c \neq ab$.

Cela peut être le cas si les triplets ont été générés au préalable par produits de Schur entre les participants.

Vérifier un triplet de Beaver en ≪sacrifiant≫ un autre triplet

On a deux triplets de Beaver $([u], [v], [w])$ et $([x], [y], [z])$.

On veut confirmer que $w = uv$.

Soit t un entier aléatoire connu de tous les participants (mais contrôlé par aucun).

> Calculer localement $[\rho] = t[u] - [x]$ et $[\sigma] = [v] - [y]$
> Révéler ρ et σ
> Calculer localement $[e] = t[w] - [z] - \sigma[x] - \rho[y] - \sigma\rho$
> Révéler e
> Si $e \neq 0$, rejeter le triplet $([u], [v], [w])$.

Échoue avec probabilité élevée si $w \neq uv$ ou $z \neq xy$.
On peut tester plusieurs $([x], [y], [z])$ pour augmenter la confiance.

SPDZ : authentifier les partages à l'aide d'un MAC

Une autre manière d'authentifier les partages qui s'applique à tous les schémas linéaire, y compris au schéma additif complet.

Un partage authentifié $\langle s \rangle$ est un double partage $([s], [\alpha \cdot s])$:

$$\langle s \rangle = (s_1, \ldots, s_n, m_1, \ldots, m_n) \text{ avec } s = \sum s_i \text{ et } \alpha \cdot s = \sum m_i$$

α est un secret global partagé.
Le participant i connaît s_i, m_i, et α_i uniquement.

On sait vérifier $\alpha \sum s_i = \sum m_i$ lorsqu'on révèle $\langle s \rangle$,
et calculer de manière homomorphe sur les partages $\langle s \rangle$.

Sécurité active même en présence de $n - 1$ attaquants.

(Voir la section 6.6 de Evans et al, *A pragmatic introduction to secure multi-party computation*, 2018.)

Point d’étape

Le partage linéaire de secrets

Une manière de calculer à plusieurs sur des données privées en contrôlant quels résultats du calcul sont révélés à tous.

De nombreux avantages :

- Adapté à de nombreuses situations pratiques (du type ≪vivre ensemble sans confiance mutuelle≫).
- En plus de la confidentialité des données privées, peut garantir la résistance aux pannes et aux attaques actives.
- Pas de cryptographie compliquée !

Deux principales limitations :

- Les protocoles sont nécessairement interactifs.
- Les coûts des communications entre participants limite beaucoup la vitesse des calculs.

Bibliographie

Bibliographie

Principale source pour ce cours :

- *Cryptography made simple*, Nigel P. Smart, Springer, 2016. Chapitre 19 et section 22.3.

Pour aller plus loin :

- *A pragmatic introduction to secure multi-party computation*, David Evans, Vladimir Kolsnikov, Mike Rosulek, NOW Publishers, 2018.