Cryptographie Avancée

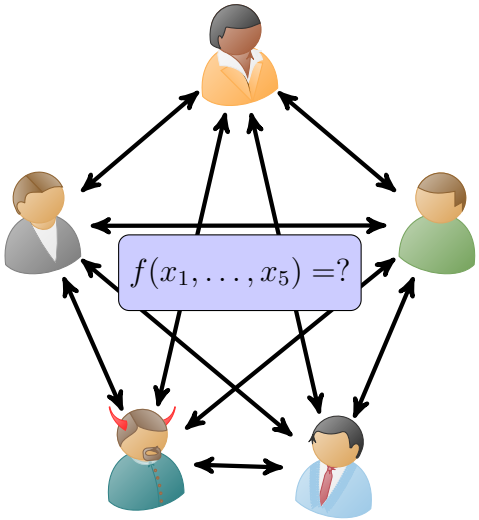
Calcul MultiPartite II - Majorité Malhonnête

Maxime Bombar

Lundi 18 Novembre

Rappel de Vendredi

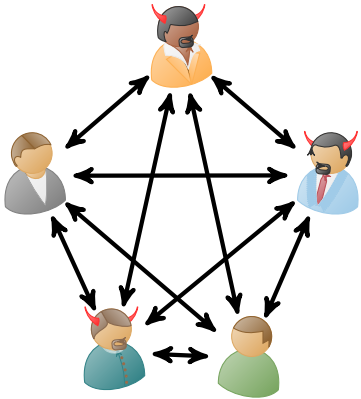
Calcul MultiPartite Sécurisé

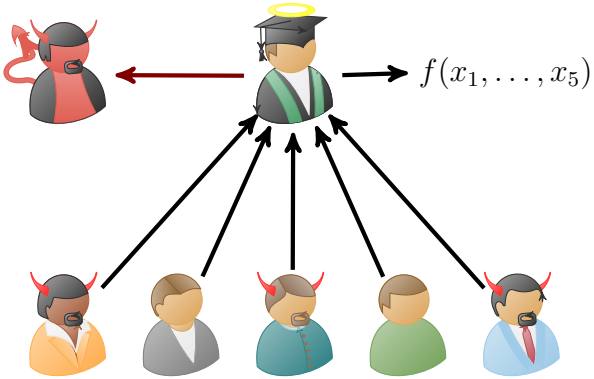


Correctness :	Les joueurs reçoivent à la fin le vrai résultat $f(x_1, \ldots, x_n))$
$t-$**Privacy**	L'adversaire n'apprend rien de plus que le résultat s'il y a au plus t adversaires.

Semi-Honnête	L'adversaire respecte le protocole.
Malicieux	Les parties corrompues peuvent avoir un comportement arbitraire.

Prouver la Sécurité





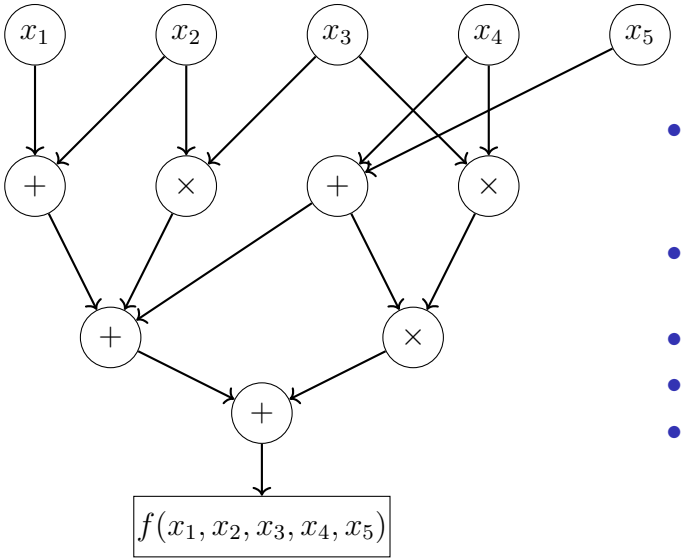
Dans le monde idéal, il n'y a pas d'attaque, par définition. Le protocole réel est sûr si un adversaire ne peut pas distinguer entre le monde réel et le monde idéal uniquement en observant les vues de toutes les parties corrompues (preuve par simulation).

Universal Composability

Un protocole réel réalisant une fonctionnalité $\mathcal{F} = f \circ g$ est dit sûr dans le modèle g-hybride, s'il peut être prouvé sûr en remplaçant tous les calculs de g par une fonctionnalité idéale $\mathcal{G}$ (et donc sans attaque).

Théorème (Canetti 2000) : Soit π un protocole calculant une fonctionnalité $\mathcal{F}$ et prouvé sûr contre t adversaires dans le modèle g-hybride. Soit σ un protocole sûr face à t adversaires, et calculant g. Si π réalise un seul appel à g par tour alors le protocole $\pi[\sigma]$ obtenu en remplaçant les appels à la fonctionnalité idéale $\mathcal{G}$ par le protocole σ calcule la fonctionnalité $\mathcal{F}$ et est sûr contre t-adversaires.

Le Modèle de Calcul : Circuits Arithmétiques



- Toute fonction calculable en temps polynomial peut-être représentée par un circuit arithmétique.
- Exemple : un circuit booléen correspond au cas $\mathbb{F}_2$.
- $x \vee y$ correspond à $x \oplus y + x \cdot y$.
- $x \wedge y$ correspond à $x \cdot y$.
- $\neg x$ correspond à $1 \oplus x$.

Le Protocole BGW

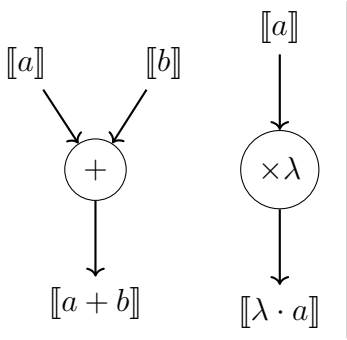
Le protocole de Ben-Or, Goldwasser et Wigderson (1988)

Pour n'importe quelle fonction $f(x_1, \ldots, x_n)$, il existe un protocole de MPC à n joueurs, capable de calculer f avec **Sécurité Parfaite**, en présence d'un adversaire **semi-honnête**, et controlant jusqu'à $\mathbf{t < n/2}$ parties.

Idées : Réaliser un $(t+1, n)$ secret-sharing des x_i et raisonner au niveau de chaque type de portes.

Portes Additives et Multiplications Publiques

Comment passer une porte additive ?

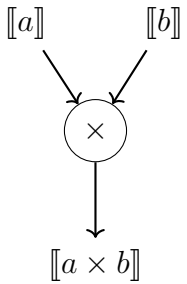


Le partage de Shamir est **linéaire**.

Chaque partie peut **localement** passer les portes.

Portes Multiplicatives

Comment passer une porte multiplicative ?



Le produit de deux parts de Shamir ne donne pas un partage valide du produit :

- Problème de degré sur les polynômes ;
- Problème sur la distribution des parts.

Outils :

- Protocole de rerandomisation et réduction de degré (c'est ici qu'on utilise $t < n/2$).
- Nécessite de communiquer.

Wishful Thinking

BGW nécessite un $(t+1, n)$ secret-sharing de Shamir. En particulier, on a besoin que $n > |\mathbb{F}|$. De plus, $t < n/2$ donc le protocole ne réalise même pas de calcul sécurisé à 2 joueurs.

Le seul moment où on a besoin de $t < n/2$ **c**st pour les multiplications.

Wishful Thinking

BGW nécessite un $(t+1, n)$ secret-sharing de Shamir. En particulier, on a besoin que $n > |\mathbb{F}|$. De plus, $t < n/2$ donc le protocole ne réalise même pas de calcul sécurisé à 2 joueurs.

Le seul moment où on a besoin de $t < n/2$ **c**st pour les multiplications.

BGW est sûr dans le modèle $\mathcal{F}_{\text{mult}}$: il suffit de trouver un autre protocole de multiplication!

Wishful Thinking

BGW nécessite un $(t+1, n)$ secret-sharing de Shamir. En particulier, on a besoin que $n > |\mathbb{F}|$. De plus, $t < n/2$ donc le protocole ne réalise même pas de calcul sécurisé à 2 joueurs.

Le seul moment où on a besoin de $t < n/2$ **c**st pour les multiplications.

BGW est sûr dans le modèle $\mathcal{F}_{\text{mult}}$: il suffit de trouver un autre protocole de multiplication!

Remplacer la sécurité statistique par calculatoire ?

Aujourd'hui

- Transferts Inconscients : Une clé de voute du MPC
- Le protocole GMW
- Optimisations : Modèle du Précalcul

Protocoles de Multiplication à 2 Joueurs

Multiplication Sécurisée

x_a, x_b, y_a, y_b aléatoires, conditionnés à $x_a + x_b = x$ et $y_a + y_b = y$.

Produit sécurisé

(x_a, y_a)

(x_b, y_b)

z_a

z_b

Objectif :

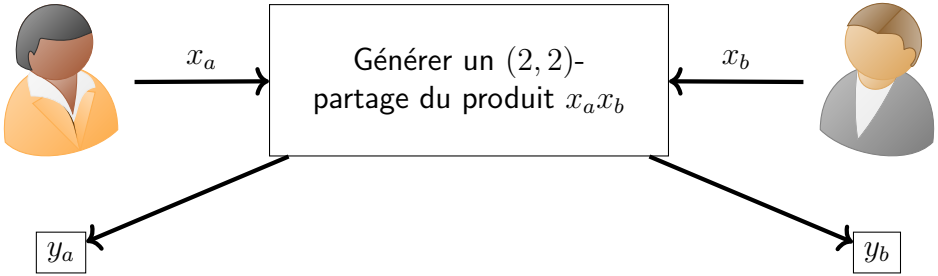
Distribuer un $(2, 2)$-sharing (z_a, z_b) de $z \stackrel{\text{def}}{=} x \cdot y$.

Outil Clé - Transfert Inconscient

Objectif :

- Bob apprend m_b.
- Alice n'apprend rien sur b.
- Bob n'apprend rien sur m_{1-b}.

Étape I - Multiplier 2 éléments



Où (y_a, y_b) doit être uniformément aléatoire, modulo la relation $y_a + y_b = x_a x_b$.

Solution

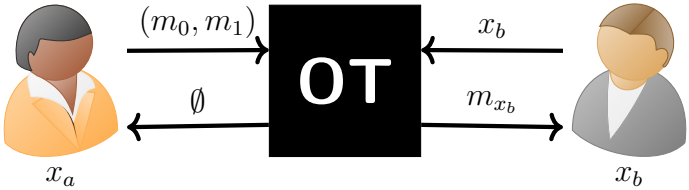
Solution

x_a

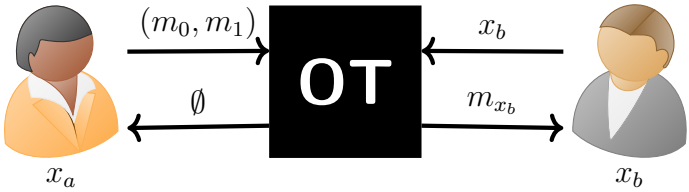
Idée : Écrire cette multiplication comme un OT.

x_b

Solution

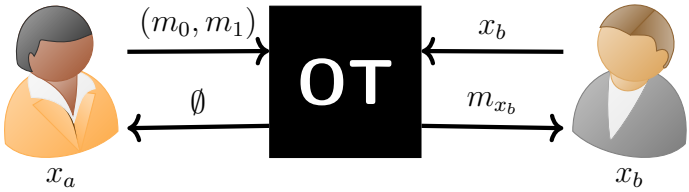


Solution



- On réalise un OT où le bit de sélection est l'entrée de Bob : x_b.
  Développons l'équation

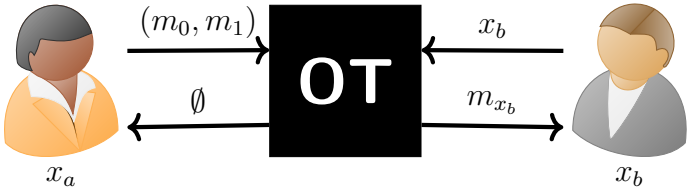
Solution



- On réalise un OT où le bit de sélection est l'entrée de Bob : x_b.

$$\begin{aligned} m_{x_b} &= x_b \cdot m_1 + (1 + x_b) \cdot m_0 \\ &= x_b \cdot (m_0 + m_1) + m_0 \\ m_0 + m_{x_b} &= (m_0 + m_1) \cdot x_b \end{aligned}$$

Solution

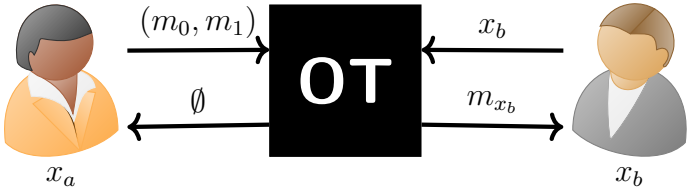


- On réalise un OT où le bit de sélection est l'entrée de Bob : x_b.

$$\begin{aligned} y_a + y_b &= x_a \cdot x_b \\ m_0 + m_{x_b} &= (m_0 + m_1) \cdot x_b \end{aligned}$$

On doit donc avoir $m_0 + m_1 = x_a$ pour que ça fonctionne.

Solution



- On réalise un OT où le bit de sélection est l'entrée de Bob : x_b.
- Alice partage son entrée x_a en (m_0, m_1) et choisit $y_a = m_0$.
- Bob choisit s_{x_2} comme sa part y_b du produit.

Étape II - Multiplication Sécurisée

x_a, x_b, y_a, y_b aléatoires, conditionnés à $x_a + x_b = x$ et $y_a + y_b = y$.

Produit sécurisé

(x_a, y_a)

(x_b, y_b)

z_a

z_b

Objectif :

Distribuer un $(2, 2)$-sharing (z_a, z_b) de $z \stackrel{\text{def}}{=} x \cdot y$.

Étape II - Multiplication Sécurisée

(x_a, y_a)

(x_b, y_b)

$$x \cdot y \quad = (x_a + x_b) \cdot (y_a + y_b)$$

Étape II - Multiplication Sécurisée

(x_a, y_a)

(x_b, y_b)

$$\begin{aligned} x \cdot y &= (x_a + x_b) \cdot (y_a + y_b) \\ &= (x_a \cdot y_a + x_a \cdot y_b) + (x_a \cdot y_b + x_b \cdot y_b)\,. \end{aligned}$$

Étape II - Multiplication Sécurisée

(x_a, y_a)

(x_b, y_b)

$$\begin{aligned} x \cdot y &= (x_a + x_b) \cdot (y_a + y_b) \\ &= \Big(\boxed{x_a \cdot y_a} + \boxed{x_a \cdot y_b} \Big) + \Big(\boxed{x_b \cdot y_a} + \boxed{x_b \cdot y_b} \Big) . \end{aligned}$$

Étape II - Multiplication Sécurisée

(x_a, y_a)

(x_b, y_b)

$$\begin{aligned} x \cdot y &= (x_a + x_b) \cdot (y_a + y_b) \\ &= \Big(\boxed{x_a \cdot y_a} + \boxed{x_a \cdot y_b} \Big) + \Big(\boxed{x_b \cdot y_a} + \boxed{x_b \cdot y_b} \Big) . \end{aligned}$$

Un produit sur des parts additives, se réalise à l'aide de 2 OT et de calculs locaux!

Étape II - Multiplication Sécurisée

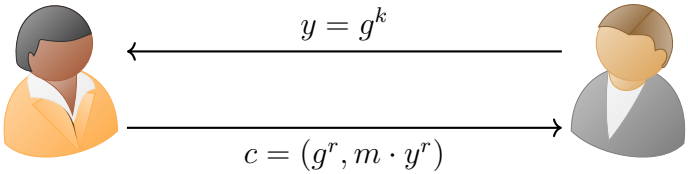
(x_a, y_a)

(x_b, y_b)

$$\begin{aligned} x \cdot y &= (x_a + x_b) \cdot (y_a + y_b) \\ &= \Big(\boxed{x_a \cdot y_a} + \boxed{x_a \cdot y_b} \Big) + \Big(\boxed{x_b \cdot y_a} + \boxed{x_b \cdot y_b} \Big) . \end{aligned}$$

Un produit sur des parts additives, se réalise à l'aide de 2 OT et de calculs locaux!

Rappel : Chiffrement ElGamal



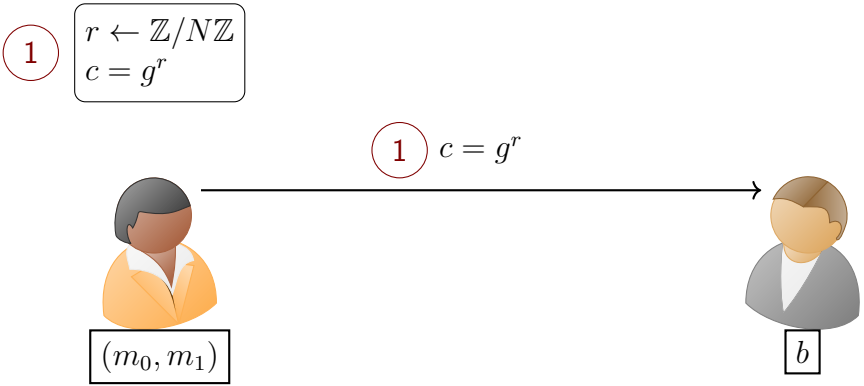
$m \in G$
$r \leftarrow \mathbb{Z}/N\mathbb{Z}$

$k \leftarrow \mathbb{Z}/N\mathbb{Z}$
$pk = g^k$
$sk = k$

Paramètres :
$G \stackrel{\text{def}}{=} \langle g \rangle$ d'ordre N.

Sécurité : ElGamal est IND-CPA si DDH est difficile dans G.

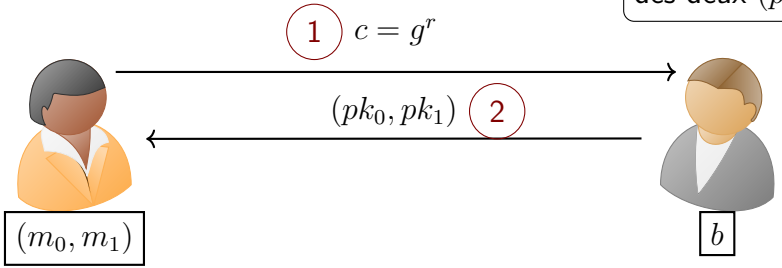
Le Protocole de Bellare-Micali (1989)



Le Protocole de Bellare-Micali (1989)

1 $r \leftarrow \mathbb{Z}/N\mathbb{Z}$
$c = g^r$

Ne connaît la clé secrète que pour l'une des deux (pk_0, pk_1). $(\star)$

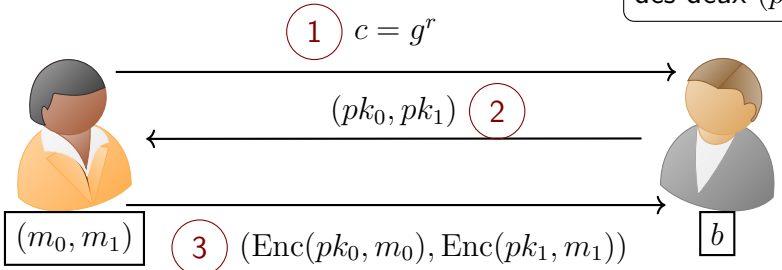


2 $z \leftarrow \mathbb{Z}/N\mathbb{Z}$
$pk_b \stackrel{\text{def}}{=} g^z$
$pk_{1-b} = c \cdot g^{-z}$.

Le Protocole de Bellare-Micali (1989)

1 $r \leftarrow \mathbb{Z}/N\mathbb{Z}$
$c = g^r$

Ne connaît la clé secrète que pour l'une des deux (pk_0, pk_1). $(\star)$



3 Vérifie $pk_0 \cdot pk_1 == c$
Sinon Abort.
Prouve $(\star)$.

2 $z \leftarrow \mathbb{Z}/N\mathbb{Z}$
$pk_b \stackrel{\text{def}}{=} g^z$
$pk_{1-b} = c \cdot g^{-z}$.

Preuve de Correction : $\mathrm{OT}((m_0, m_1); b) = (\emptyset; m_b)$

Bob :

- Connaît sk_b, donc peut déchiffrer m_b.
- Par construction, $pk_{1-b} = c \cdot g^{-z} = g^{r-z}$. La connaissance de $sk_{1-b} = r - z$ implique celle de r, qui était un challenge DLOG aléatoire. Donc il ne peut pas apprendre m_{1-b} si le chiffrement est sûr.

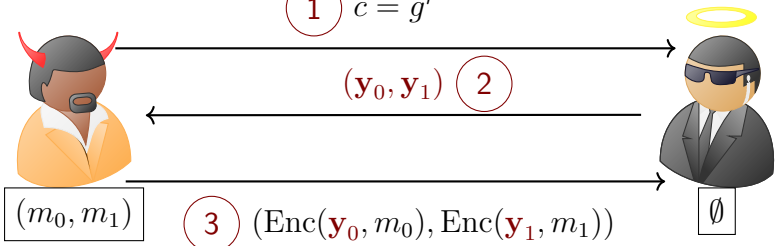
Alice :

- pk_b et pk_{1-b} sont indistinguables donc Alice ne peut pas savoir quelle clé secrète connaît Bob. Elle n'apprend donc rien sur b.

Sécurité contre Alice : Simulation de Bob

1 $r' \leftarrow \mathbb{Z}/N\mathbb{Z}$
$c = g^{r'}$

2 $z \leftarrow \mathbb{Z}/N\mathbb{Z}$
$\mathbf{y}_0 \stackrel{\text{def}}{=} g^z, \quad \mathbf{y}_1 = c \cdot g^{-z}.$

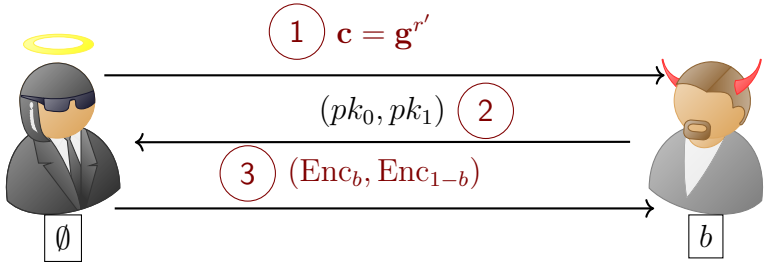


$$\begin{aligned} \text{Sim}(m_0, m_1) &= \{r', g^{r'}, m_0, m_1, \mathbf{y}_0, \mathbf{y}_1, \text{Enc}(\mathbf{y}_0, m_0), \text{Enc}(\mathbf{y}_1, m_1)\} \\ &\approx \\ \text{view}_{A,\text{reel}} &= \{r, g^r, m_0, m_1, pk_0, pk_1, \text{Enc}(pk_0, m_0), \text{Enc}(pk_1, m_1)\} \end{aligned}$$

Sécurité contre Bob : Simulation d'Alice

1 $r' \leftarrow \mathbb{Z}/N\mathbb{Z}$
$c = g^{r'}$

2 $z \leftarrow \mathbb{Z}/N\mathbb{Z}$
$pk_b \stackrel{\text{def}}{=} g^z, pk_{1-b} = c \cdot g^{-z}.$



$$\begin{aligned}
\text{Sim}(b, m_b) &= \{b, \mathbf{g}^{r'}, pk'_0, pk'_1, sk'_b, \text{Enc}(pk'_b, m_b), \text{Enc}(pk'_{1-b}, \mathbf{g}^{r''}), m_b\} \\
&\approx_c \\
\text{view}_{B,\text{reel}} &= \{b, g^r, pk_0, pk_1, sk_b, \text{Enc}(pk_0, m_0), \text{Enc}(pk_1, m_1), m_b\}
\end{aligned}$$

Remarque : Oblivious Sampleable Public-Keys

La sécurité contre un Bob corrompu repose sur **deux** propriétés importantes d'ElGamal :

- IND-CPA (reposant elle-même sur DDH).
- Oblivious Sampleable Public Keys ($\mathrm{pk\text{-}OS}$), qui signifie qu'on peut générer des clés publiques aléatoires sans connaître la clé secrète associée.

Définition : Un schéma de chiffrement $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ vérifie la propriété $\mathrm{pk\text{-}OS}$ si

- Il existe un algorithme PPT Samp tel que

$$\{pk \mid pk \leftarrow \mathrm{Samp}\} \approx \{pk \mid (pk, sk) \leftarrow \mathrm{Gen}\}.$$

- Il existe un algorithme PPT pkSim tel que

$$\{(pk, r) \mid pk \leftarrow \mathrm{Samp}, r \leftarrow \$\} \approx_c \{(pk, r) \mid (pk, sk) \leftarrow \mathrm{Gen}, r \leftarrow \mathrm{pkSim}(pk)\}$$

Généralisation : Multiplication Sécurisée à n-joueurs

Le protocole de multiplication présenté ici se généralise aisément à n joueurs en réalisant $n \cdot (n-1)$ instances du protocole de multiplication à 2 joueurs.

Le Protocole GMW

MPC avec sécurité passive totale

Le protocole de Goldreich, Micali, Wigderson (1987)

Pour n'importe quelle fonction $f(x_1, \ldots, x_n)$, il existe un protocole de MPC à n joueurs, capable de calculer f avec **Sécurité Calculatoire**, en présence d'un adversaire **semi-honnête**, et pouvant contrôler jusqu'à $n-1$ parties.

Vers une Sécurité Malicieuse

Le protocole GMW peut être transformé en un protocole résistant aux **adversaires actifs** de manière standard grâce à un procédé connu sous le nom de $\mathrm{GMW\ compiler}$.
Il combine deux outils :

- Mises en gage (*Commitments*)
- Preuves Zéro-Knowledge que le protocole est respecté

Quelques Ordres de Grandeur

Communication : Les instantiations modernes de Bellare-Micali pour réaliser des OT utilisent des groupes de courbes elliptiques. Les implémentations efficaces nécessitent de l'ordre de 750 bits de communication par OT, ou encore 1.5kb par porte multiplicative dans le circuit. Pour réaliser une distance d'édition entre deux chaînes de 4095 bits, on estime qu'il faut alors plus de 1TO de communication.

Temps de Calcul : La cryptographie à clé publique n'est pas rapide. Si on instanciait BGW avec le protocole de Bellare-Micali pour les OT, cela nécessiterait plus de 700 heures sur un serveur de calcul de taille raisonnable pour calculer la distance d'édition précédente.

Optimiser la Communication

Aléa Corrélé et Produit Sécurisé

On suppose que chaque joueur possède un partage additif d'un **produit aléatoire** :

$$(\mathbf{u}, \mathbf{v}, \mathbf{w} \stackrel{\text{def}}{=} \mathbf{u} \cdot \mathbf{v})$$

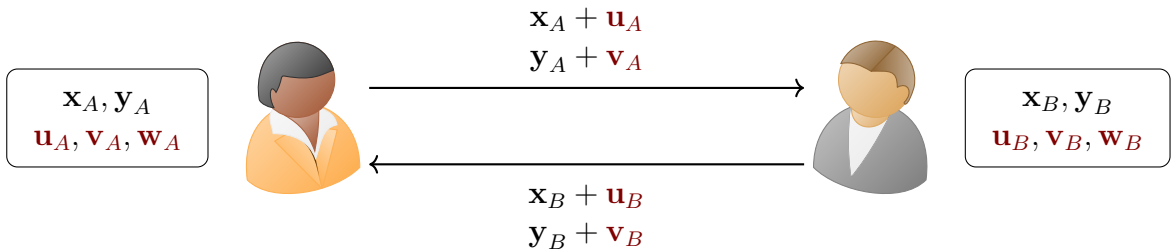
$\mathbf{x}_A, \mathbf{y}_A$
$\mathbf{u}_A, \mathbf{v}_A, \mathbf{w}_A$

$\mathbf{x}_B, \mathbf{y}_B$
$\mathbf{u}_B, \mathbf{v}_B, \mathbf{w}_B$

Aléa Corrélé et Produit Sécurisé

On suppose que chaque joueur possède un partage additif d'un **produit aléatoire** :

$$(\mathbf{u}, \mathbf{v}, \mathbf{w} \overset{\text{def}}{=} \mathbf{u} \cdot \mathbf{v})$$



Aléa Corrélé et Produit Sécurisé

On suppose que chaque joueur possède un partage additif d'un **produit aléatoire** :

$$(\mathbf{u}, \mathbf{v}, \mathbf{w} \stackrel{\text{def}}{=} \mathbf{u} \cdot \mathbf{v})$$

$\mathbf{x}_A, \mathbf{y}_A$
$\mathbf{u}_A, \mathbf{v}_A, \mathbf{w}_A$
α, β

$\alpha = \mathbf{x} + \mathbf{u}$
$\beta = \mathbf{y} + \mathbf{v}$

$\mathbf{x}_B, \mathbf{y}_B$
$\mathbf{u}_B, \mathbf{v}_B, \mathbf{w}_B$
α, β

Aléa Corrélé et Produit Sécurisé

On suppose que chaque joueur possède un partage additif d'un **produit aléatoire** :

$$(\mathbf{u}, \mathbf{v}, \mathbf{w} \stackrel{\text{def}}{=} \mathbf{u} \cdot \mathbf{v})$$

$\mathbf{x}_A, \mathbf{y}_A$
$\mathbf{u}_A, \mathbf{v}_A, \mathbf{w}_A$
α, β

$\alpha = \mathbf{x} + \mathbf{u}$
$\beta = \mathbf{y} + \mathbf{v}$

$\mathbf{x}_B, \mathbf{y}_B$
$\mathbf{u}_B, \mathbf{v}_B, \mathbf{w}_B$
α, β

$$\begin{aligned} \mathbf{x} \cdot \mathbf{y} &= \Big((\mathbf{x} + \mathbf{u}) - \mathbf{u}\Big) \cdot \Big((\mathbf{y} + \mathbf{v}) - \mathbf{v}\Big) \\ &= (\alpha - \mathbf{u}) \cdot (\beta - \mathbf{v}) \\ &= \alpha\beta - \alpha\mathbf{v} - \beta\mathbf{u} + \mathbf{w} \end{aligned}$$

Combinaison affine **publique** du triplet!

Triplet Multiplicatif de Beaver

Définition : Un triplet de Beaver, ou triplet multiplicatif, est un triplet $(a, b, c) \in \mathbb{F}^3$ uniformément distribué, conditionné à $c = a \cdot b$.

Il est possible de réaliser un protocole de multiplication sécurisée à 2 joueurs en consommant un tel triplet et en ne communiquant que 2 éléments de $\mathbb{F}$ par joueur. Ces triplets sont **indépendants** des entrées des deux joueurs, et peuvent être précalculés.

Remarque : Un triplet de Beaver aléatoire peut-être généré à partir de 2 transferts inconscients aléatoires.

Le modèle de l'Aléa Corrélé

Précalcul

$(\mathbf{a}_i, \mathbf{b}_i)_{i \leqslant N} \leftarrow (\mathbb{F} \times \mathbb{F})^N$

$(\llbracket \mathbf{a}_i \rrbracket, \llbracket \mathbf{b}_i \rrbracket, \llbracket \mathbf{a}_i \cdot \mathbf{b}_i \rrbracket)$

Phase « en ligne »

Protocole GMW

Précalculer les OT

Précalcul

$(\mathbf{a}_i, \mathbf{b}_i)_{i \leqslant N} \leftarrow (\mathbb{F} \times \mathbb{F})^N$

$(\llbracket \mathbf{a}_i \rrbracket, \llbracket \mathbf{b}_i \rrbracket, \llbracket \mathbf{a}_i \cdot \mathbf{b}_i \rrbracket)$

Phase « en ligne »

Protocole GMW

L'Histoire Continue

Comment générer efficacement des triplets de Beaver / des OT aléatoires ?

- Protocoles d'extension d'OT : À partir d'un petit nombre d'OT, on en génère un nombre arbitraire (Beaver, 1996), protocole IKNP (Ishai, Kilian, Nissim, Petrank, 2003)
- Paradigme récent : Précalcul Silencieux, *Pseudorandom Correlation Generators* (PCG).