Cryptographie Avancée

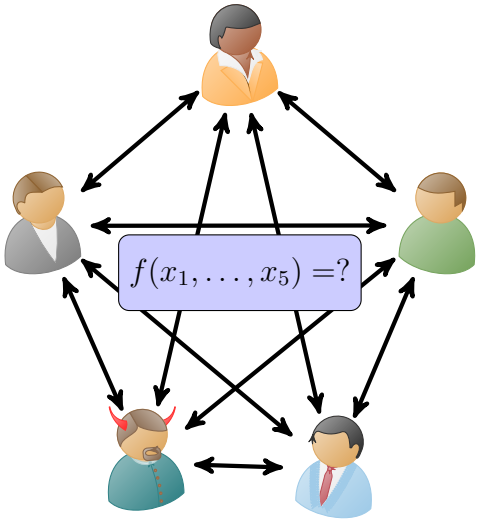
MPC in the Head: Preuves Zero-Knowledge via MPC

Maxime Bombar

Lundi 25 Novembre

Rappels des Derniers Cours

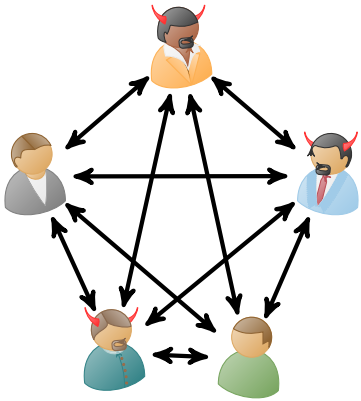
Calcul MultiPartite Sécurisé

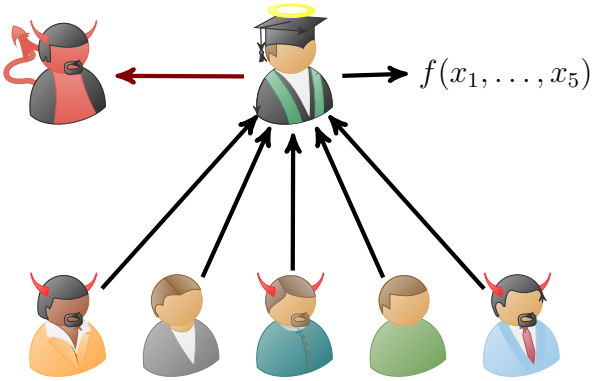


Correctness :	Les joueurs reçoivent à la fin le vrai résultat $f(x_1, \ldots, x_n))$
$t-$**Privacy**	L'adversaire n'apprend rien de plus que le résultat s'il y a au plus t adversaires.

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

Prouver la Sécurité





Un protocole réel est sûr contre t adversaires semi-honnêtes s'il existe un **simulateur** efficace ayant accès aux vues des t parties corrompues, et pouvant founir un transcript indistinguable du protocole réel, alors qu'il n'intéragit qu'avec la fonctionalité idéale.

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 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.

Ne permet pas d'offrir du calcul à 2 joueurs !

Le Protocole BGW

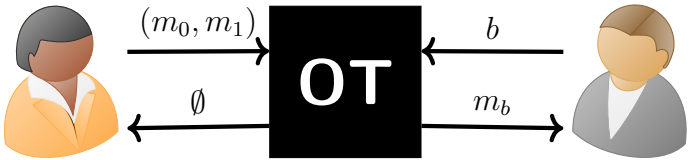
Le protocole de Ben-Or, Goldwasser et Wigderson (1988)

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.

Ne permet pas d'offrir du calcul à 2 joueurs !

Transferts Inconscients



Il existe des protocoles de multiplications sécurisés à 2 joueurs dans le modèle $\mathcal{F}_{\mathrm{OT}}$-hybride.

Il existe des protocoles de transferts inconscients (*e.g.*, Bellare-Micali).

Le protocole GMW

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.

Preuves Zero-Knowledge au Service du MPC

Le protocole GMW peut être étendu en un protocole à la **sécurité malicieuse** de manière standard grâce à un procédé connu sous le nom de GMW compiler.

Il combine deux outils :

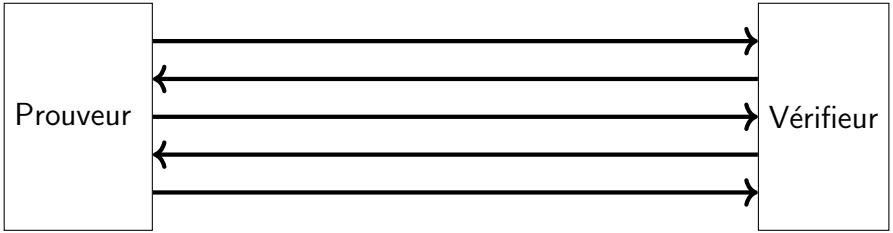
- Mises en gage (*Commitments*)
- Preuves Zero-Knowledge que le protocole est respecté

Aujourd'hui : MPC au Service des Preuves Zéro-Knowledge

- Rappels : Preuves Zero-Knowledge ≪ classiques ≫
- Comment le MPC peut-être exploité pour construire des preuves Zero-Knowledge ?
- Applications : Construction de Signatures Post-Quantiques.

Rappels : Preuves Zero-Knowledge

Preuve Interactive



Un système de preuve intéractif pour un langage $\mathcal{L}$ est la donnée d'un couple d'algorithmes (P, V) et d'un protocole π tels que

- **Efficacité :** V est PPT et renvoie un bit d'information Accept (1) ou Reject (0).
- **Complétude :** (*Completeness*) Si $x \in \mathcal{L}$ alors $P(x) \leftrightarrow V$ accepte.
- **Robustesse :** (*Soundness*) Si $x \notin \mathcal{L}$ alors pour tout prouveur $P^\star$, $\mathbb{P}(P^\star(x) \leftrightarrow V \ \mathrm{Accept})$ est négligeable.

Pour la Culture

IP = PSPACE

FOCS (symposium on Foundations Of Computer Science), 1990.

ADI SHAMIR

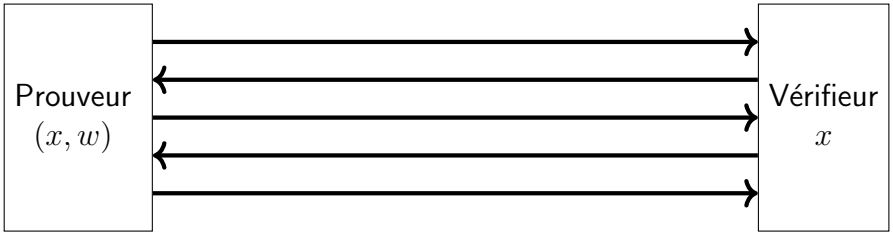
En cryptographie, on demande en plus que le prouveur soit aussi PPT.

Systèmes de preuves pour P ou pour NP ?

Preuve pour **P** : Notion de **calcul vérifiable**.

Preuves pour **NP** : Le prouveur possède un **témoin** supplémentaire.
Dans le cas des preuves Zero-Knowledge, ce témoin est **gardé secret**.

Preuve Zero-Knowledge



(P, V) est **zero-knowledge** par rapport au témoin w si V ne peut rien apprendre d'autre sur w que ce qu'il peut déduire de $x \in \mathcal{L}$ (ou $x \notin \mathcal{L}$).

Remarque : Une preuve zero-knowledge est un **protocole de MPC**! Le caractère zero-knowledge se prouve à l'aide d'un simulateur S, en montrant que la vue de V intéragissant avec S est indistinguable de la vue de V dans le véritable protocole contre n'importe quel prouveur $P^\star$.

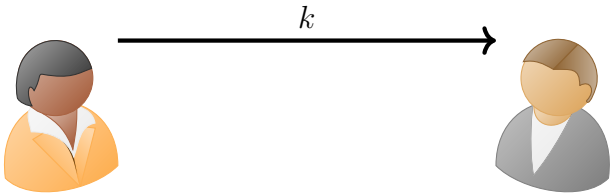
Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.

$r \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$
$k \leftarrow r^e$

Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

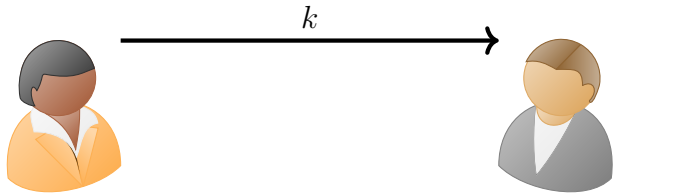
- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



$r \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$
$k \leftarrow r^e$

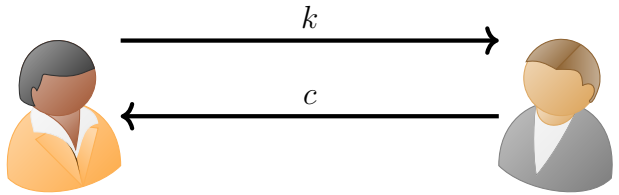
Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



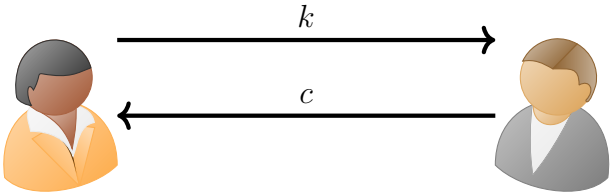
Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



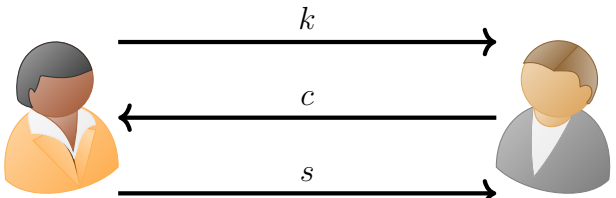
Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** (N, e), $y = x^e \mod N$.
- **Objectif :** Prouver qu'on connaît x.



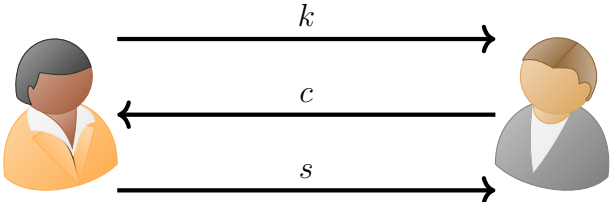
Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



Soundness ?

$r \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$
$k \leftarrow r^e$

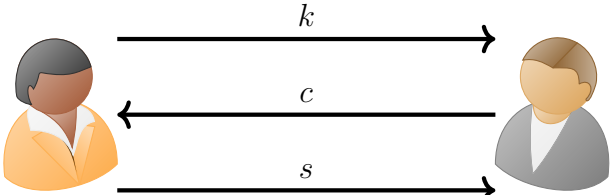
$c \leftarrow \{0, \ldots, e-1\}$

$s \leftarrow rx^c \mod N$

$s^e \equiv ky^c \mod N$?

Zero-Knowledge pour RSA : Guillou-Quisquater (1988)

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



Soundness ?
$\to 1/e.$

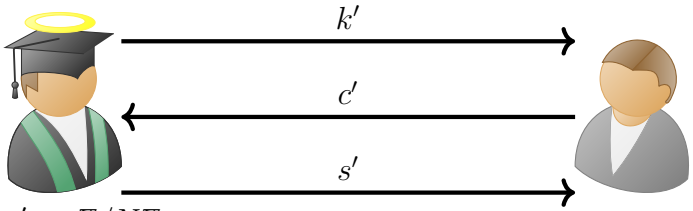
Zero-Knowledge ?

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.

Exercice : Montrez, à l'aide d'une preuve par simulation, que ce protocole est bien zéro-knowledge.

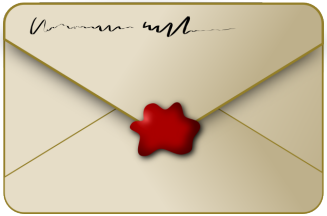
Zero-Knowledge ?

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



(1) Génère $s' \leftarrow \mathbb{Z}/N\mathbb{Z}$.
(2) Génère $c' \leftarrow \{0, \ldots, e-1\}$.
(3) Calcule $k' \stackrel{\text{def}}{=} s^e y^{-c} \mod N$
Retourne (k', c', s').

Commitment Schemes

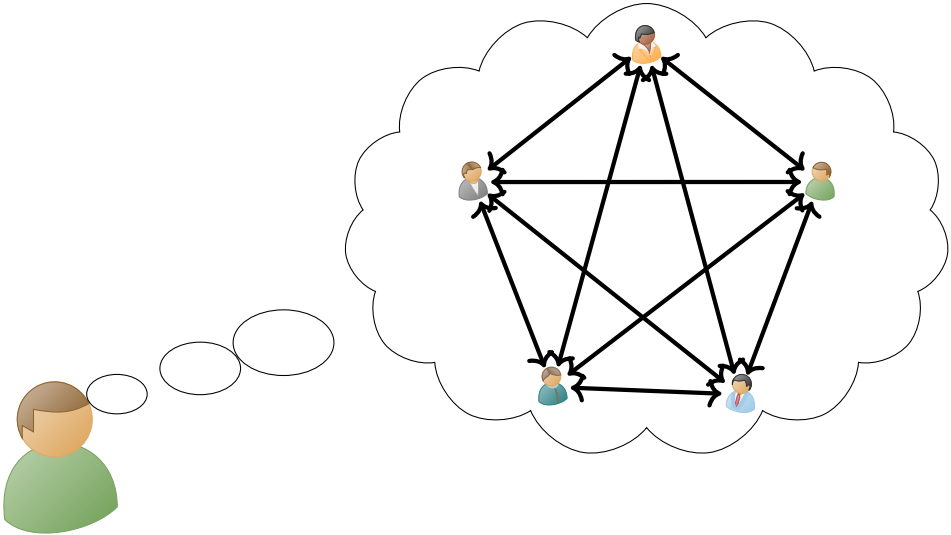


$$\text{Commit}(x; r) = (c, s)$$
$$\text{Open}(c, s) = x \text{ ou } \perp.$$

Hiding : c ne doit rien révéler sur x.
Binding : Il doit être impossible d'ouvrir un commitment sur 2 valeurs distinctes.

MPC dans la Tête

Framework



MPC-in-the-Head

Ishai, Kushilevitz, Ostrovsky, Sahai (STOC, 2007)
Zero-Knowledge from Secure Multiparty Computation

Objectif : Preuve zero-knowledge pour un langage $\mathcal{L} \in \mathrm{NP}$,
i.e., $x \in \mathcal{L} \iff \exists w\ \mathcal{R}(x, w) = 1$.

Méthode : Définir une fonctionalité à n-parties

$$g(x; w_1, \ldots, w_n) = \mathcal{R}(x, w_1 \oplus \cdots \oplus w_n).$$

Simuler (dans sa tête) un protocole π sécurisé à n parties pour calculer g tel que

- π est sécurisé dans le modèle semi-honnête
- Autorise 2 adversaires.

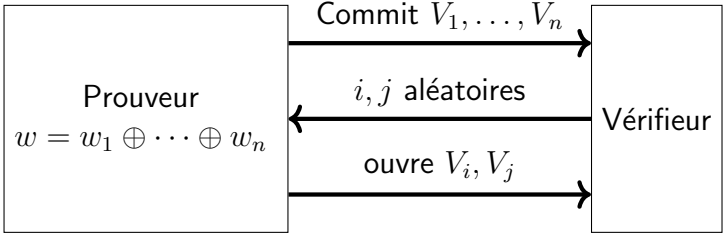
Consistance

- On note V_i la vue de P_i.
  - Son entrée w_i
  - Son aléa r_i
  - Tous les messages **reçus** par P_i.
  - Le résultat $g(x; w_1, \ldots, w_n)$

- Deux vues V_i et V_j sont dites **consistantes** si les messages émis $P_i \to P_j$ sont identiques aux messages reçus $P_j \leftarrow P_i$.

Remarque : Toutes les paires (V_i, V_j) sont consistantes ssi il existe une exécution honnête du protocole de sorte que la vue de P_i soit exactement V_i.

Preuve Zero-Knowledge

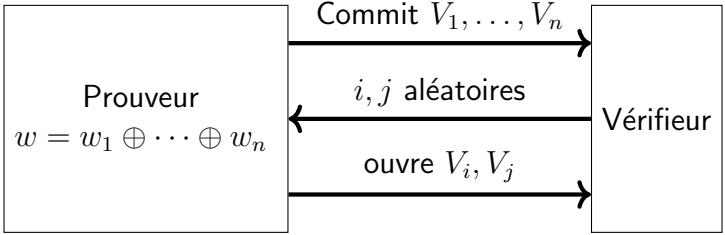


V accepte si et seulement si

- $g(x, w) = 1$
- V_i, V_j sont consistantes

- **Completeness :**
- **Zero-Knowledge :**

Preuve Zero-Knowledge

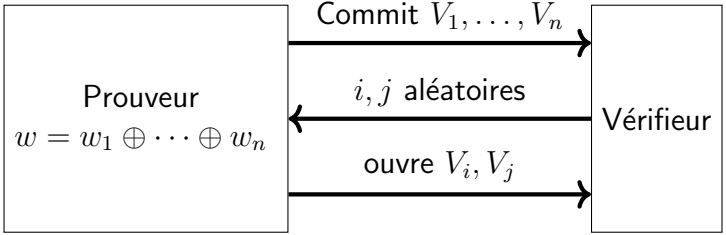


V accepte si et seulement si

- $g(x, w) = 1$
- V_i, V_j sont consistantes

- **Completeness :** Correction du protocole de MPC.
- **Zero-Knowledge :**

Preuve Zero-Knowledge

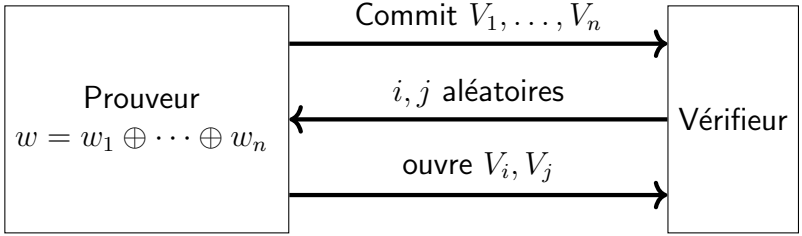


V accepte si et seulement si

- $g(x, w) = 1$
- V_i, V_j sont consistantes

- **Completeness :** Correction du protocole de MPC.
- **Zero-Knowledge :** Puisque le protocole est sûr face à 2 adversaires, V n'apprend pas w (Formellement : simulateurs)

Soundness

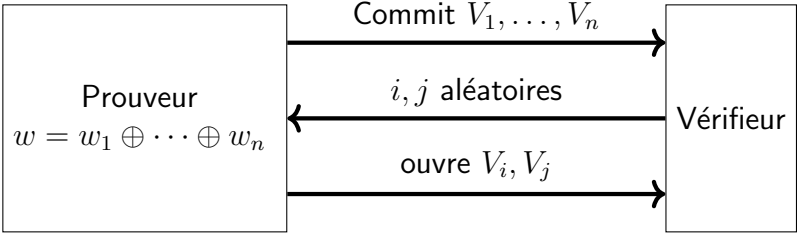


Soundness : Si $x \notin \mathcal{L}$ alors

(1) Soit $g(x, w_1, \ldots, w_n) = 0$

(2) Soit il existe (V_{i_0}, V_{j_0}) inconsistantes. Quelle est la probabilité de détection ?

Soundness



Soundness : Si $x \notin \mathcal{L}$ alors

(1) Soit $g(x, w_1, \ldots, w_n) = 0$

(2) Soit il existe (V_{i_0}, V_{j_0}) inconsistantes. $\mathbb{P}_{\text{Détection}} \geqslant \binom{n}{2}^{-1} = \dfrac{2}{n(n-1)}$.

MPC-in-the-Head avec BGW

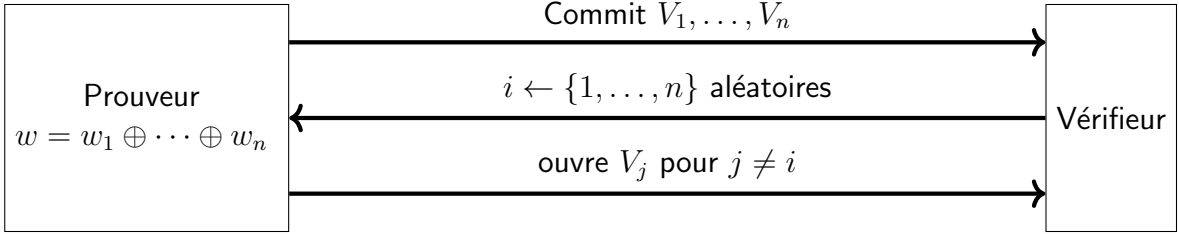
Quelques remarques :

- Sécurité : **théorie de l'information**!
- La probabilité de tricher augmente avec n.
- La complexité augmente aussi avec n.
- $\Rightarrow$ choisir n minimal qui autorise 2 adversaires !

Avec BGW, on a besoin d'au moins $n \geqslant 5$ parties :

- P commit 5 vues (et en révèle 2).
- S'il triche, V le détecte avec probabilité $\geqslant 1/10$.
- **Soundess error :** Après k exécutions indépendantes, la probabilité d'erreur est $\leqslant (9/10)^k \leqslant 2^{-128}$ pour $k \geqslant 842$.

MPC-in-the-Head avec GMW



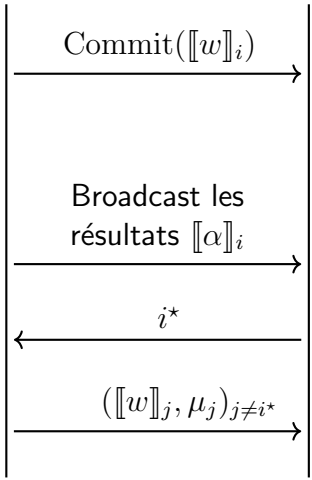
- On choisit un protocole sûr jusqu'à $N - 1$ corruptions.
- Au moins une des vues doit être fausse.
- P triche sans détection si la vue fausse est $V_i \Rightarrow$ **soundness error** $= \dfrac{1}{n}$.
- Après k répétition, la probabilité d'erreur est n^{-k}.
- Pour $n = 5$, il faut répéter 56 fois pour avoir une erreur $\leqslant 2^{-128}$.

La Transformation MPCitH

(1) Générer et commiter les parts $[\![w]\!] \stackrel{\text{def}}{=} ([\![w]\!]_1, \ldots, [\![w]\!]_N)$

(2) Émuler le protocole de MPC dans la tête.

(4) Ouvrir les vues $\{1, \ldots, N\} \setminus \{i\}$



(3) Choisir un joueur aléatoire $i^\star \leftarrow \{1, \ldots, N\}$

Exemple : RSA-in-the-Head

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.

Déterminer un protocole MPC-in-the-Head pour prouver la connaissance de x.

Exemple : RSA-in-the-Head

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.

$x_1, \ldots, x_{n-1} \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$

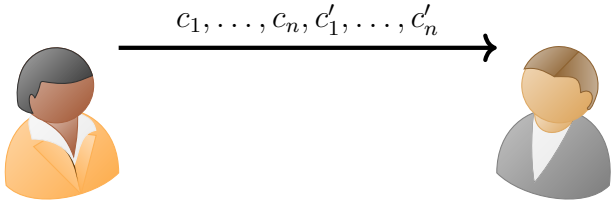
$x_n \stackrel{\text{def}}{=} x / \prod x_i$

$y_i \stackrel{\text{def}}{=} x_i^e$

$c_i = \text{Commit}(x_i), c'_i = \text{Commit}(y_i)$

Exemple : RSA-in-the-Head

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



$x_1, \ldots, x_{n-1} \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$

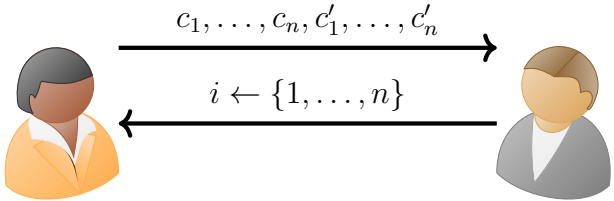
$x_n \stackrel{\text{def}}{=} x / \prod x_i$

$y_i \stackrel{\text{def}}{=} x_i^e$

$c_i = \text{Commit}(x_i), c'_i = \text{Commit}(y_i)$

Exemple : RSA-in-the-Head

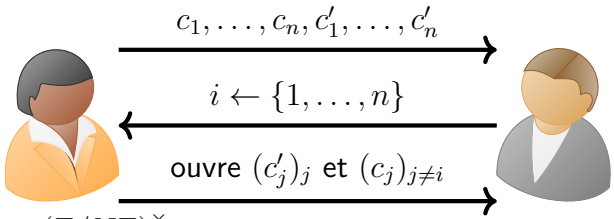
- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



$x_1, \ldots, x_{n-1} \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$
$x_n \stackrel{\text{def}}{=} x / \prod x_i$
$y_i \stackrel{\text{def}}{=} x_i^e$
$c_i = \text{Commit}(x_i), c'_i = \text{Commit}(y_i)$

Exemple : RSA-in-the-Head

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



$x_1, \ldots, x_{n-1} \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$

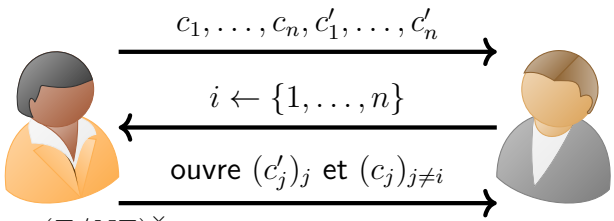
$x_n \stackrel{\text{def}}{=} x / \prod x_i$

$y_i \stackrel{\text{def}}{=} x_i^e$

$c_i = \text{Commit}(x_i), c'_i = \text{Commit}(y_i)$

Exemple : RSA-in-the-Head

- **Données :** (N, e), $y = x^e \mod N$.
- **Objectif :** Prouver qu'on connaît x.



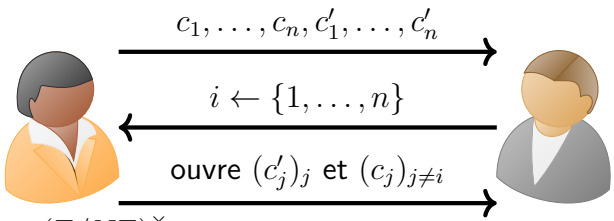
$x_1, \ldots, x_{n-1} \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$
$x_n \stackrel{\text{def}}{=} x / \prod x_i$
$y_i \stackrel{\text{def}}{=} x_i^e$
$c_i = \text{Commit}(x_i), c'_i = \text{Commit}(y_i)$

Accepte si et seulement si :

- Les commits sont valides.
- $y_j = x_j^e \mod N$ pour $j \neq i$
- si $y_1 \cdots y_n = y \mod N$.

Exemple : RSA-in-the-Head

- **Données :** $(N, e), \quad y = x^e \mod N.$
- **Objectif :** Prouver qu'on connaît x.



$x_1, \ldots, x_{n-1} \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$
$x_n \stackrel{\text{def}}{=} x / \prod x_i$
$y_i \stackrel{\text{def}}{=} x_i^e$
$c_i = \text{Commit}(x_i), c'_i = \text{Commit}(y_i)$

Accepte si et seulement si :

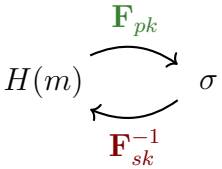
- Les commits sont valides.
- $y_j = x_j^e \mod N$ pour $j \neq i$
- si $y_1 \cdots y_n = y \mod N$.

Soundness : $1/n$.

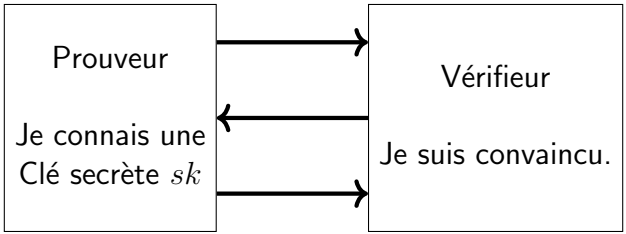
Applications : Signatures (Post-Quantiques)

Comment Construire une Signature ?

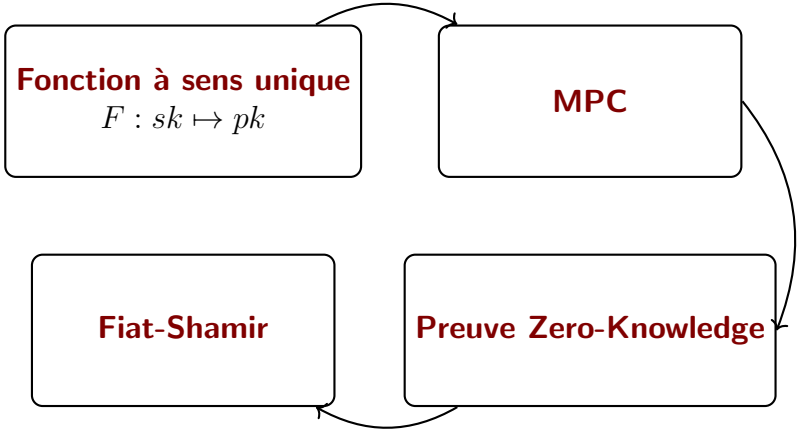
Hash and Sign



Preuve ZK et Fiat-Shamir



Signature MPC-in-the-Head



Plus de flexibilité sur le choix de F!

Signature MPC-in-the-Head

On choisit une fonction F à sens unique.

La clé secrète s_k est donnée par un élément aléatoire du domaine de F.

La clé publique est donnée par l'image $p_k = F(s_k)$.

On construit un protocole de MPC efficace pour la connaissance d'un antécédent de p_k par F.

Choix de la fonction F

Le paradigme MPCitH offre une grande flexibilité sur le choix de la fonction à sens unique F. Il suffit en effet de définir un protocole de MPC efficace!

- Réseaux Euclidiens (SIS, LWE)
- Codes : Problème de Décodage (métrique de Hamming, métrique rang)
- Multivarié : Systèmes Polynomiaux Quadratiques
- Symmétrique : Inverser un chiffrement par blocs.
- ...

Quelques Éléments de Taille

Clé secrète : La clé secrète est simplement représentée par une graine.
Clé publique : La clé publique dépend de la représentation de l'image de F.

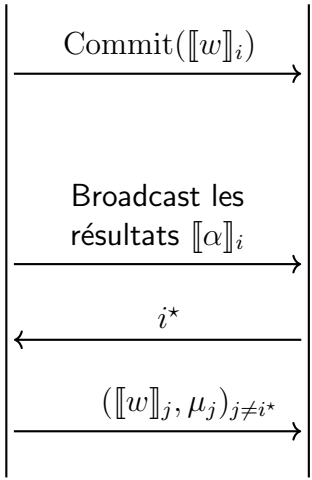
Signature : La taille de la signature augmente avec la taille des commits, l'erreur de *soundness* (puisqu'il faut faire plus de répétition si celle-ci est petite).

Template Général MPCitH

(1) Générer et commiter les parts $[\![w]\!] \stackrel{\text{def}}{=} ([\![w]\!]_1, \ldots, [\![w]\!]_N)$

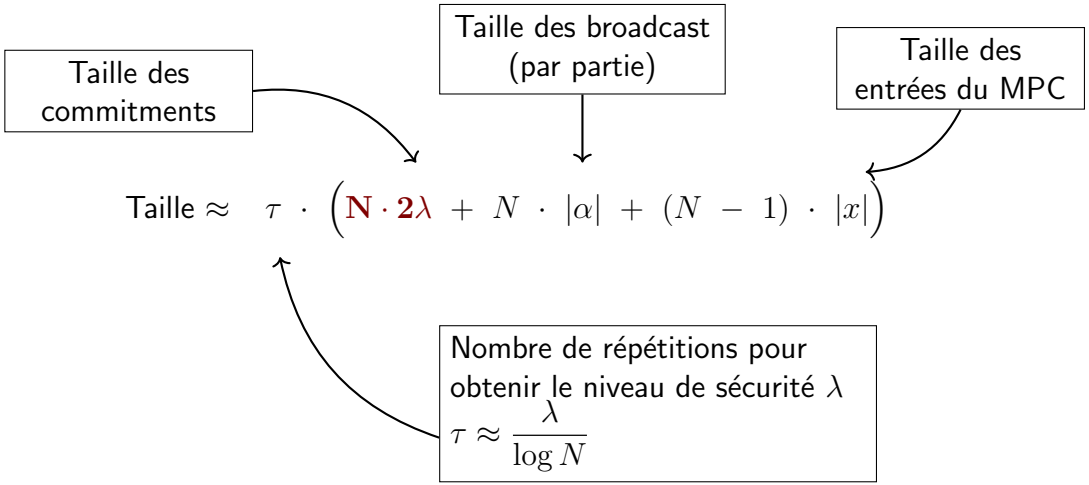
(2) Émuler le protocole de MPC dans la tête.

(4) Ouvrir les vues $\{1, \ldots, N\} \setminus \{i\}$

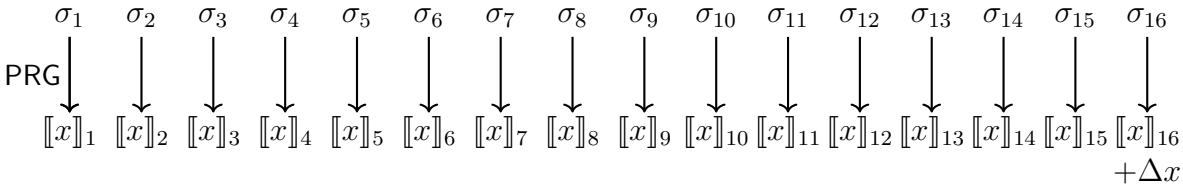


(3) Choisir un joueur aléatoire $i^\star \leftarrow \{1, \ldots, N\}$

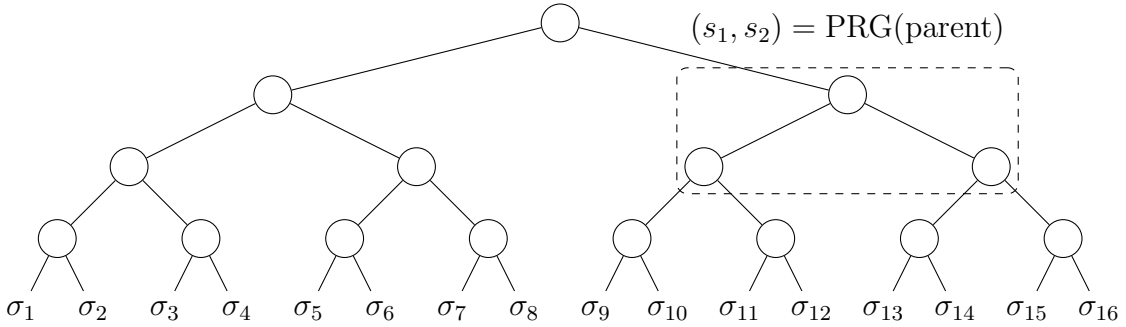
Taille des Signatures



Compresser les Preuves : Arbre GGM

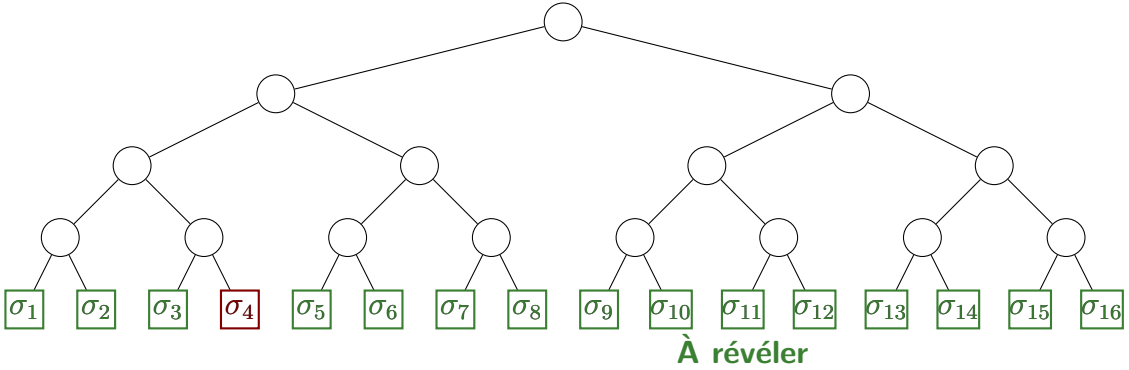


Compresser les Preuves : Arbre GGM



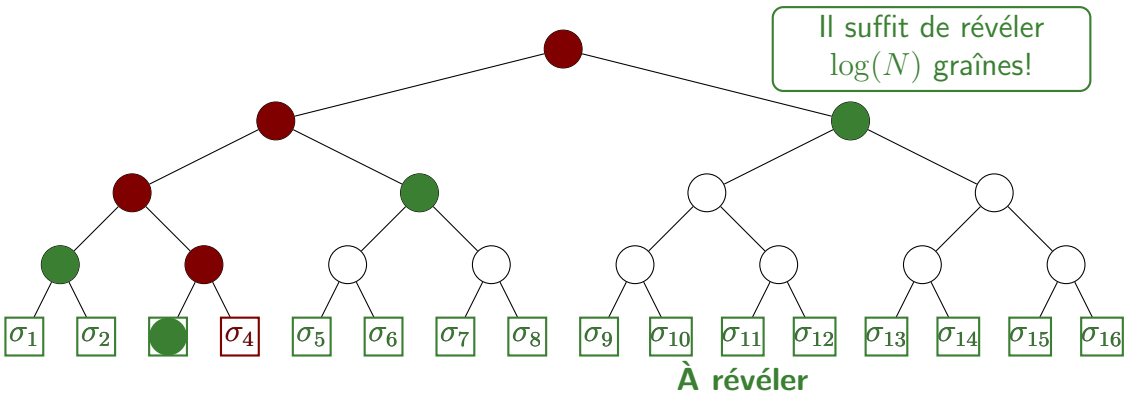
$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_3$ $[\![x]\!]_4$ $[\![x]\!]_5$ $[\![x]\!]_6$ $[\![x]\!]_7$ $[\![x]\!]_8$ $[\![x]\!]_9$ $[\![x]\!]_{10}$ $[\![x]\!]_{11}$ $[\![x]\!]_{12}$ $[\![x]\!]_{13}$ $[\![x]\!]_{14}$ $[\![x]\!]_{15}$ $[\![x]\!]_{16}$

Compresser les Preuves : Arbre GGM



$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_3$ $[\![x]\!]_4$ $[\![x]\!]_5$ $[\![x]\!]_6$ $[\![x]\!]_7$ $[\![x]\!]_8$ $[\![x]\!]_9$ $[\![x]\!]_{10}$ $[\![x]\!]_{11}$ $[\![x]\!]_{12}$ $[\![x]\!]_{13}$ $[\![x]\!]_{14}$ $[\![x]\!]_{15}$ $[\![x]\!]_{16}$

Compresser les Preuves : Arbre GGM



$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_3$ $[\![x]\!]_4$ $[\![x]\!]_5$ $[\![x]\!]_6$ $[\![x]\!]_7$ $[\![x]\!]_8$ $[\![x]\!]_9$ $[\![x]\!]_{10}$ $[\![x]\!]_{11}$ $[\![x]\!]_{12}$ $[\![x]\!]_{13}$ $[\![x]\!]_{14}$ $[\![x]\!]_{15}$ $[\![x]\!]_{16}$

MPC-in-the-Head au NIST

Il reste encore 14 soumissions en lice dans la compétition post-quantique du NIST. Parmi celles-ci, 5 utilisent le paradigme MPC-in-the-Head.

Nom	Hypothèse	Taille de signature (kB)
Mirath	MinRank	5.6
MQOM	Systèmes Polynomiaux	6.3
PERK	Permuted Kernel Problem	6.1
RYDE	Problème de Décodage en Métrique rang	6.0
SDitH	Problème de Décodage en métrique de Hamming	8.2