Le calcul sécurisé : calculer sur des données chiffrées ou privées

Introduction et études de cas

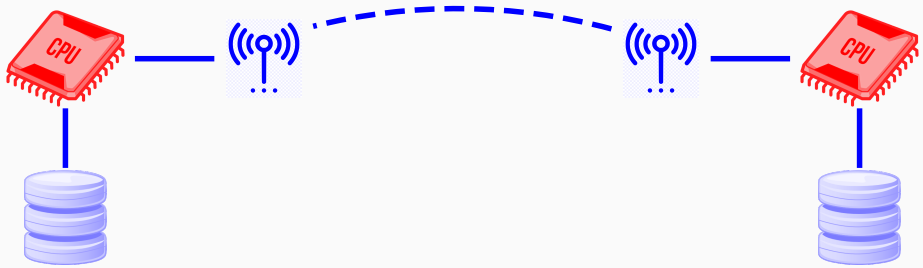
Xavier Leroy

2025-11-06

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

Introduction

Sécuriser les données avec la cryptographie

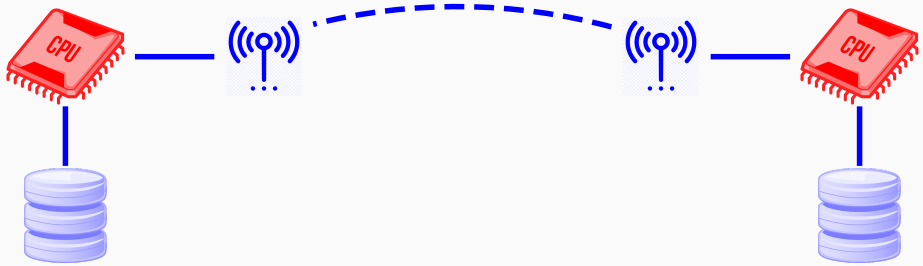


Solides techniques cryptographiques pour protéger les données

- au repos (systèmes de fichiers, bases de données);
- en transit (sur les réseaux de communication).

Garanties : confidentialité (chiffrement); intégrité (signatures).

Sécuriser les calculs ?



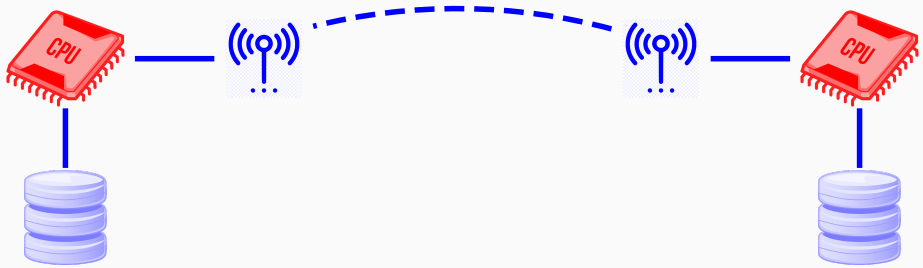
Généralement, les données sont déchiffrées avant calcul.

Exemple : requête sur une base de données chiffrée.

Risque de fuites d'information pendant les calculs.

Besoin de confier les clés de déchiffrement à un tiers pas toujours de confiance (*cloud computing*).

Sécuriser les calculs avec la cryptographie !



≪Il faut bien déchiffrer avant de calculer !≫ …Vraiment ?

Des techniques cryptographiques récentes :

- Chiffrement homomorphe : calculer sur des données chiffrées sans connaître la clé de déchiffrement.
- Calcul multipartite sécurisé : plus généralement, calculer sur des données privées sans les révéler aux calculateurs.

Sécuriser le calcul : approche ≪systèmes et langages≫ (cours de 2022)

Quelques principes et de nombreux mécanismes pour :

- Isolation matérielle et logiciel des calculs.
  (Virtualisation, *sandboxing*, typage, …)
- Contrôle d'accès.
  (Permissions, capacités, mots de passe, …)
- Contrôle des flux d'information.

Efficace pour garantir l'intégrité, moins pour la confidentialité.

Vulnérable aux attaques ≪par en-dessous≫
(fuites d'info par les temps d'exécution, contourner la virtualisation, …)

Sécuriser le calcul : approche ≪cryptographique≫ (ce cours)

Les données chiffrées ou ≪masquées≫ ne révèlent aucune information à qui n'a pas la clé / le masque.

Si on calcule sur de telles données, on peut donc les laisser ≪fuiter≫, tant que la clé reste secrète.

Sécuriser le calcul : approche ≪cryptographique≫ (ce cours)

Les données chiffrées ou ≪masquées≫ ne révèlent aucune information à qui n'a pas la clé / le masque.

Si on calcule sur de telles données, on peut donc les laisser ≪fuiter≫, tant que la clé reste secrète.

$\rightarrow$ La sécurité d'une petite donnée (clé, masque, …)
peut suffire à garantir la sécurité d'un gros calcul.

(Une extension du principe de Kerckhoff : ≪la sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé≫.)

On data banks and privacy homomorphisms (1978)

ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

Massachusetts Institute of Technology
Cambridge, Massachusetts

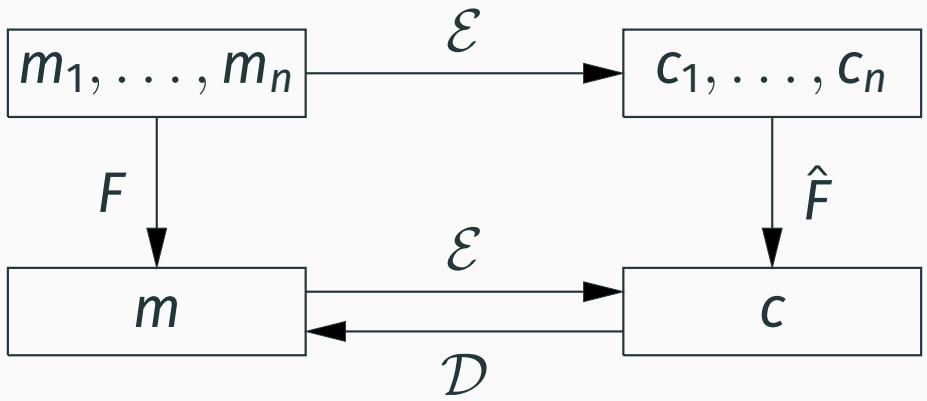
I. INTRODUCTION

Encryption is a well-known technique for preserving the privacy of sensitive information. One of the basic, apparently inherent, limitations of this technique is that an information system working with encrypted data can at most store or retrieve the data for the user; any more complicated operations seem to require that the data be decrypted before being operated on.

On data banks and privacy homomorphisms (1978)

This limitation follows from the choice of encryption functions used, however, and although there are some truly inherent limitations on what can be accomplished, we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call "privacy homomorphisms"; they form an interesting subset of arbitrary encryption schemes (called "privacy transformations").

Chiffrement homomorphe



Soit $F(m_1, \ldots, m_n)$ une fonction sur les données en clair.

Peut-on trouver un chiffrement $(\mathcal{E}, \mathcal{D})$ et une fonction $\hat{F}(c_1, \ldots, c_n)$ sur les données chiffrées tels que

$$\hat{F}(\mathcal{E}(m_1), \ldots, \mathcal{E}(m_n)) = \mathcal{E}(F(m_1, \ldots, m_n)) \qquad ?$$

Si oui, on peut faire calculer $F(m_1, \ldots, m_n)$ à un tiers sans lui révéler les m_i :

$$F(m_1, \ldots, m_n) = \mathcal{D}(\hat{F}(\mathcal{E}(m_1), \ldots, \mathcal{E}(m_n))$$

RSA est homomorphe pour la multiplication

Chiffrement RSA : (e, N clé publique ; d clé secrète)

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

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.

($\rightarrow 2^e$ cours pour d'autres exemples)

Calcul multipartite sécurisé (*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

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

Avec un tiers de confiance :

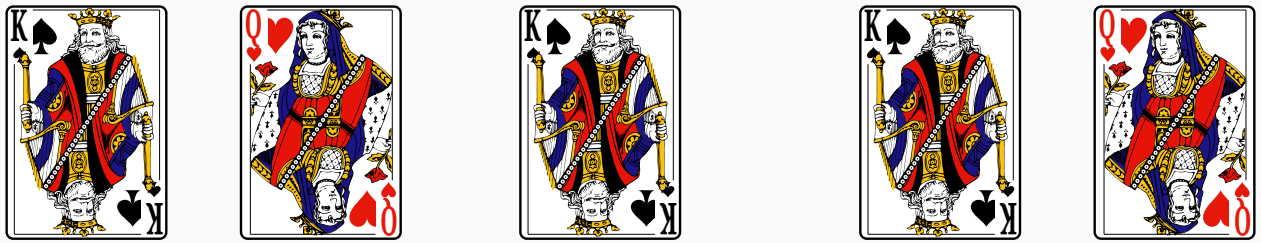
- Chaque participant i envoie son offre x_i au tiers.
- Le tiers détermine j t.q. $x_j = \min(x_1, \ldots, x_n)$ et l'annonce.
- Les participants apprennent que l'offre de j est minimale.
- Les participants n'apprennent rien d'autre sur les montants des offres des autres participants.

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 ?

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.

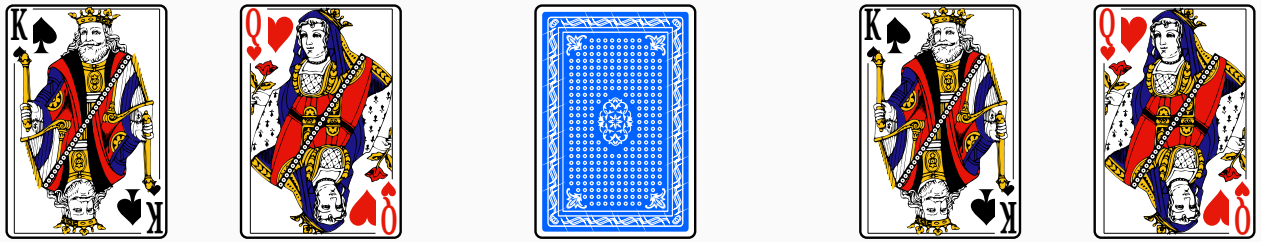


On pose un roi (face cachée).

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.

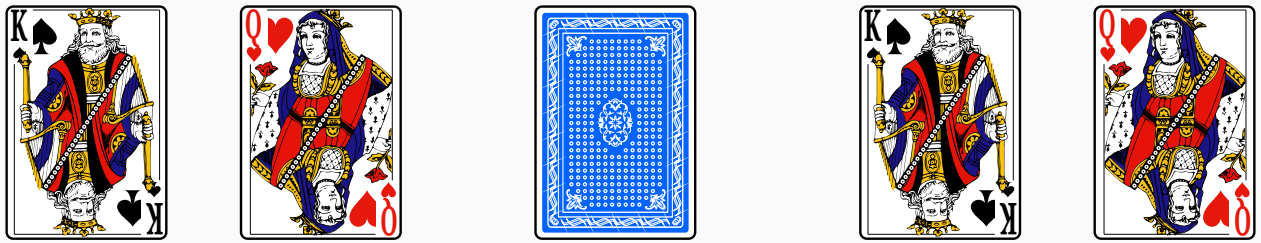


On pose un roi (face cachée).

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.



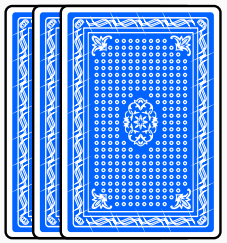
On pose un roi (face cachée).

Alice pose roi puis dame si "oui", dame puis roi si "non".

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.



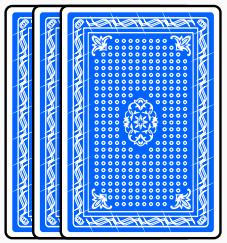
On pose un roi (face cachée).

Alice pose roi puis dame si "oui", dame puis roi si "non".

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.



On pose un roi (face cachée).

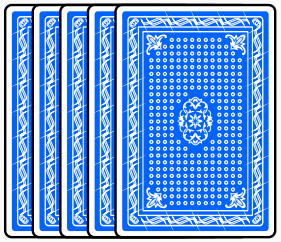
Alice pose roi puis dame si "oui", dame puis roi si "non".

Bob pose dame puis roi si "oui", roi puis dame si "non".

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.



On pose un roi (face cachée).

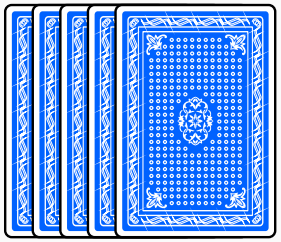
Alice pose roi puis dame si "oui", dame puis roi si "non".

Bob pose dame puis roi si "oui", roi puis dame si "non".

Exemple : calcul bipartite d'un ≪et≫ logique

Alice et Bob décident s'ils vont se revoir, et souhaitent éviter la situation embarrassante où l'un dit ≪oui≫ et l'autre ≪non≫.

Un protocole avec 5 cartes : 3 rois, 2 dames.



On pose un roi (face cachée).

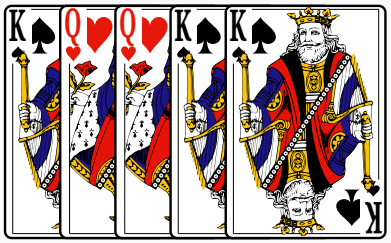
Alice pose roi puis dame si "oui", dame puis roi si "non".

Bob pose dame puis roi si "oui", roi puis dame si "non".

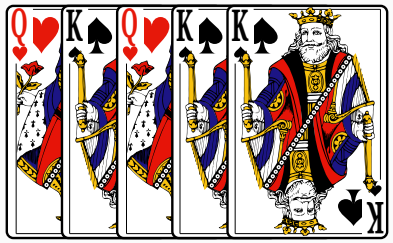
Ils coupent le jeu et le retournent.

Calcul multipartite d'un ≪et≫ logique

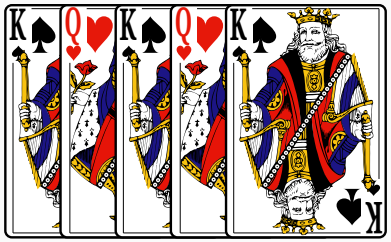
À rotation (permutation circulaire) des cartes près, on a 4 configurations possibles :



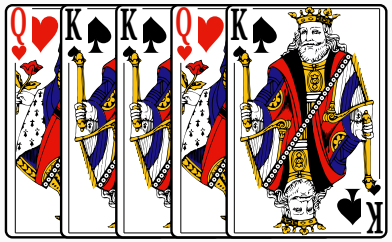
oui / oui



oui / non



non / oui



non / non

Le résultat est ≪oui≫ si et seulement si les deux dames sont adjacentes à rotation près.

Les configurations oui/non, non/oui, non/non sont égales à rotation près. Le résultat ≪non≫ ne révèle pas qui a voté ≪non≫.

Plan du cours

Chiffrement semi-homomorphe (pour une seule opération).

Chiffrement faiblement homomorphe (pour $+$ et $\times$ mais limité).

Chiffrement totalement homomorphe par *bootstrap*.

Les problèmes LWE et RLWE et leurs utilisations.

Le schéma BGV.

Pour aller plus loin :

CKKS $\rightarrow$ séminaire de Damien Stehlé, 20 nov.

TFHE $\rightarrow$ séminaire d'Ilaria Chillotti, 27 nov.

Cours 4 et 5 : (27 novembre et 4 décembre 2025)
Calcul multipartite sécurisé : calculer sur des données privées

Partage de secret : additif, de Shamir, linéaire.

Calcul distribué sur les secrets partagés.

Résistance aux attaques actives.

Les circuits brouillés de Yao.

Le transfert inconscient (*oblivious transfer*).

Pour aller plus loin : séminaire de Geoffroy Couteau, 4 déc.

Preuves interactives. Protocoles «sigma».

Exemples de preuves ZK pour des problèmes particuliers.

Preuves non-interactives.

Une construction générique de preuves SNARK
(*succint non-interactive argument of knowledge*).

Pour aller plus loin : séminaire de Michele Orrù, même jour.

Introduction au chiffrement fonctionnel.

Pour aller plus loin : séminaire de David Pointcheval, même jour.

Obfuscation indistinguable.

Bilan du cours.

Programme du séminaire

13/11 Pierrick Gaudry (CNRS)

Outils cryptographiques pour le vote électronique.

20/11 Damien Stehlé (CryptoLab)

Chiffrement totalement homomorphe CKKS.

27/11 Ilaria Chillotti (DESILO Inc)

Chiffrement totalement homomorphe : panorama, applications et nouvelles directions.

04/12 Geoffroy Couteau (CNRS)

Calcul sécurisé et aléa corrélé, de la théorie à la pratique.

11/12 Michele Orrù (CNRS)

Des preuves zero-knowledge à l'anonymat en ligne.

18/12 David Pointcheval (Cosmian)

Le chiffrement fonctionnel : agréger des données sensibles.

Étude de cas : calcul multipartite sécurisé d’une moyenne

Le problème du salaire moyen

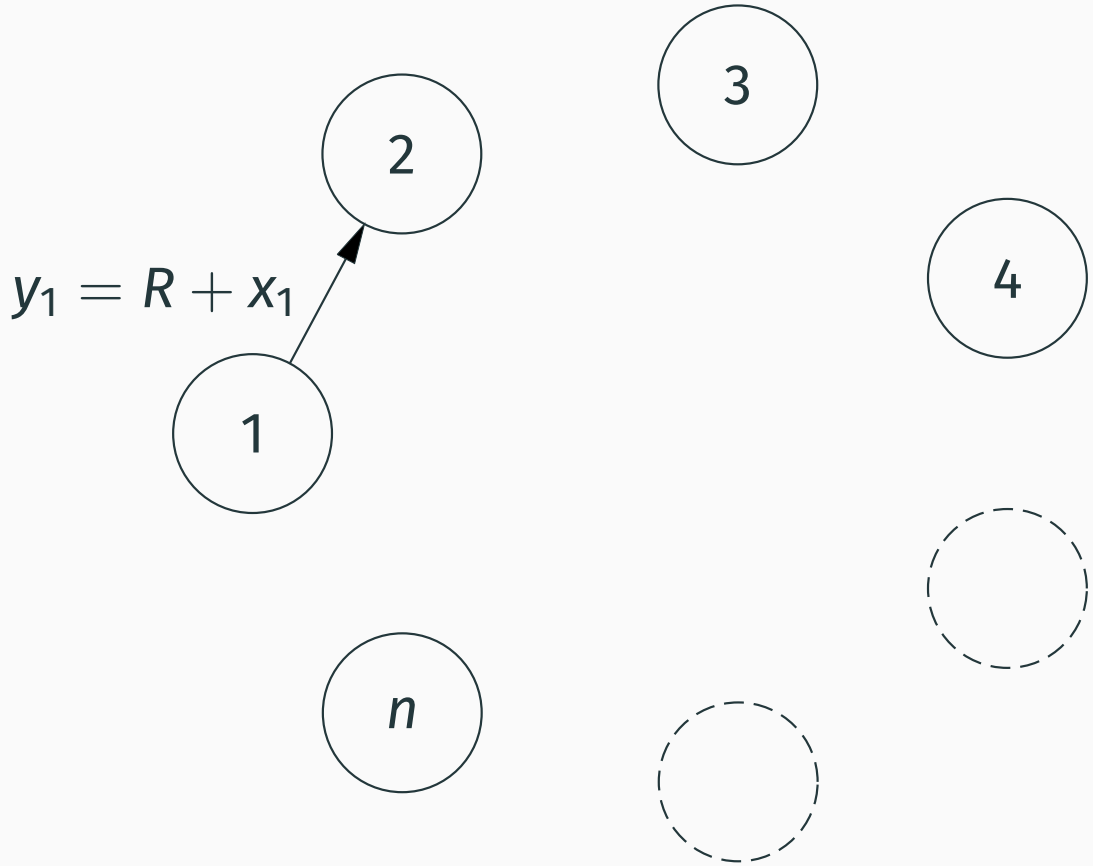
n personnes veulent calculer leur salaire moyen,
et le partager entre tous les participants,
sans révéler leurs salaires.

Une solution avec un tiers de confiance (TDC) :

- Chaque participant communique (confidentiellement) son salaire au TDC.
- Le TDC calcule la moyenne et l'annonce.
- Le TDC oublie toutes les informations reçues.

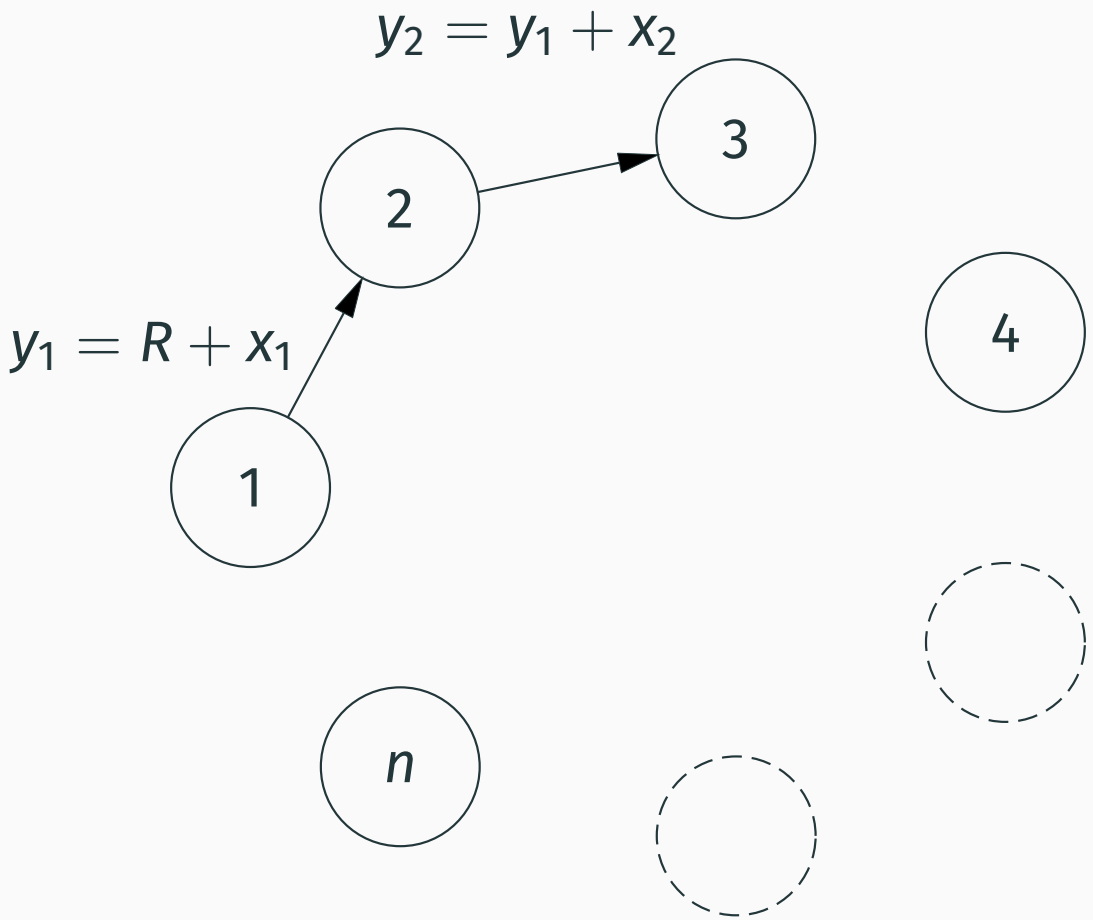
Peut-on obtenir le même résultat et les mêmes garanties sans faire intervenir un tiers de confiance ?

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$



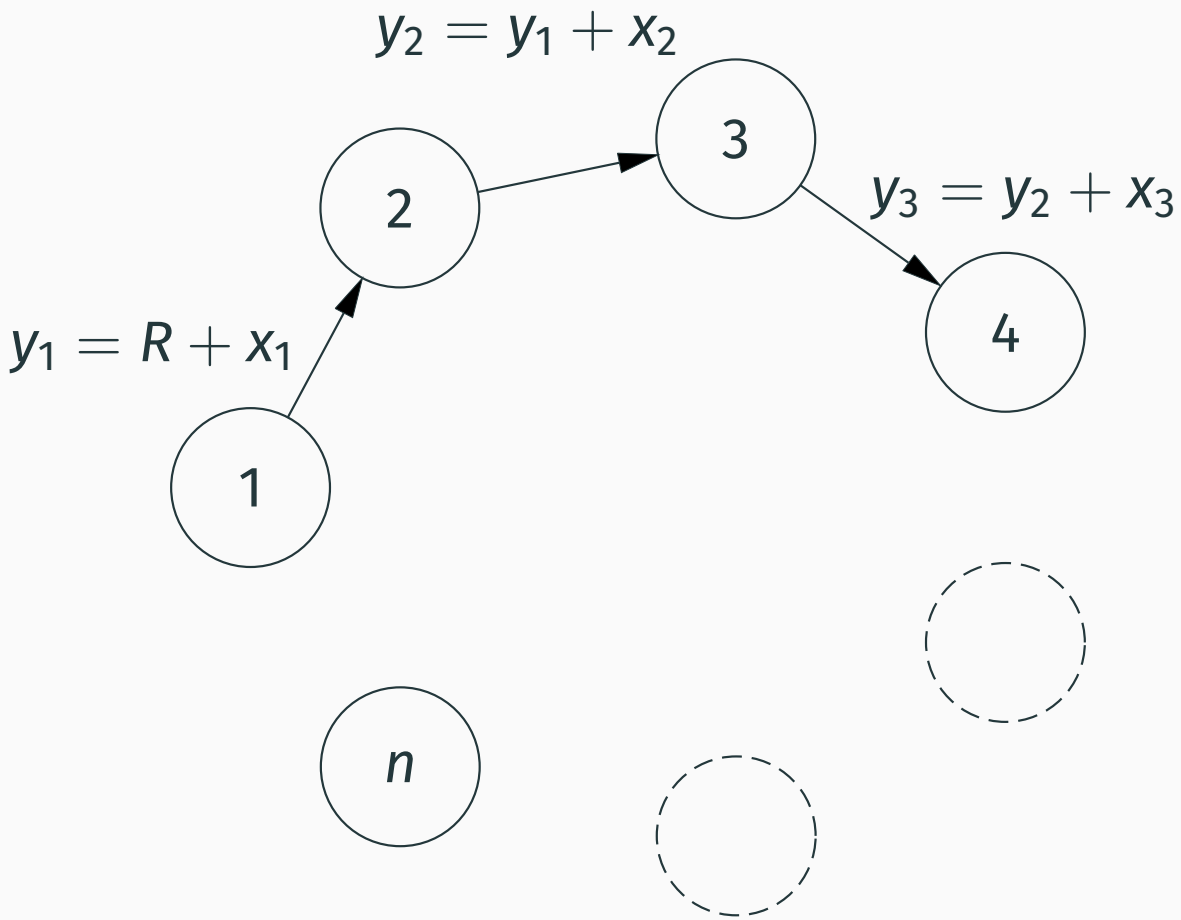
Le participant 1 tire au hasard un nombre $R \gg x_1, \ldots, x_n$ et transmet $R + x_1$ au participant 2. (Masquage de x_1 par R.)

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$



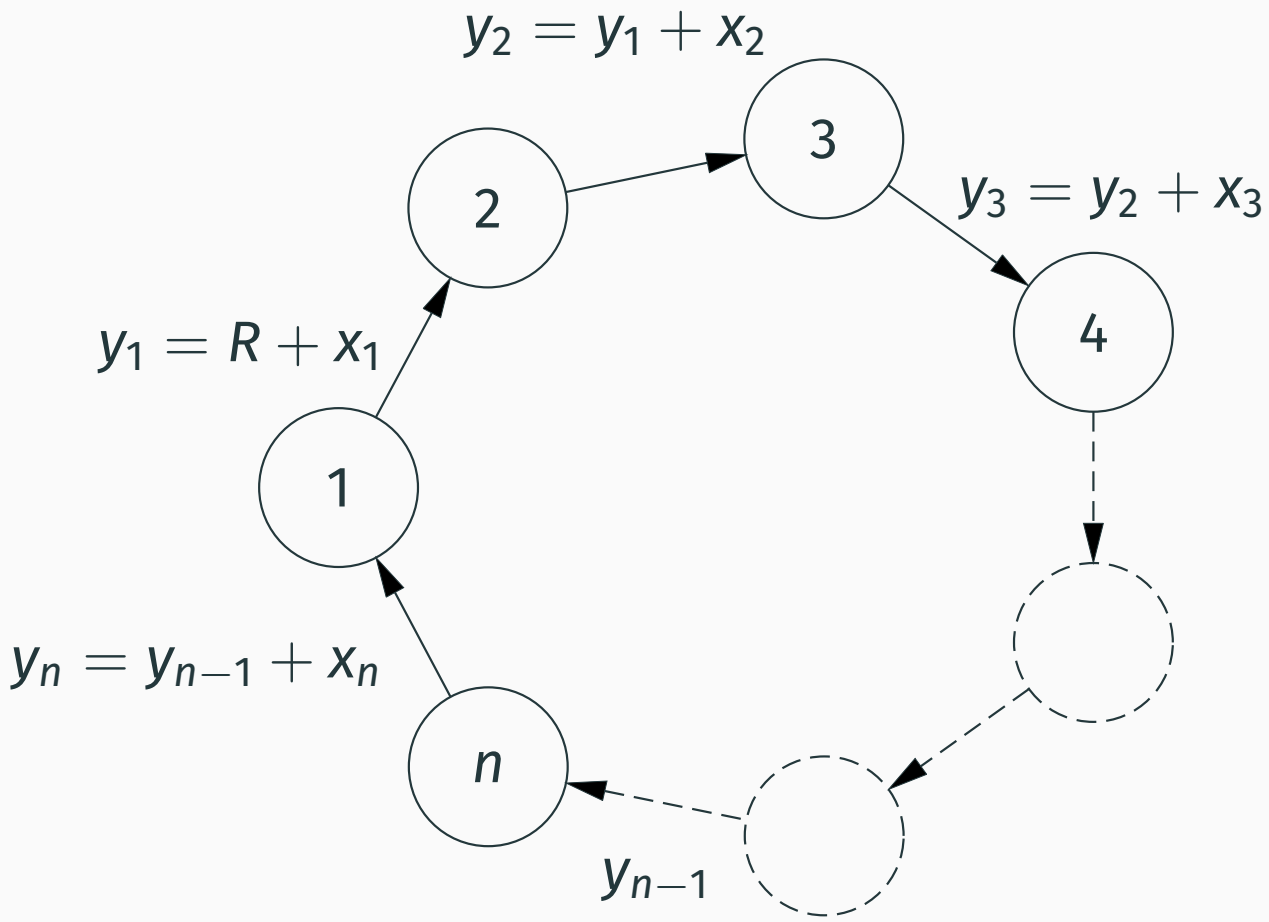
Chaque participant $i = 2, \ldots, n$ reçoit y_{i-1}, ajoute x_i, et transmet cette somme au participant suivant $i + 1$.

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$



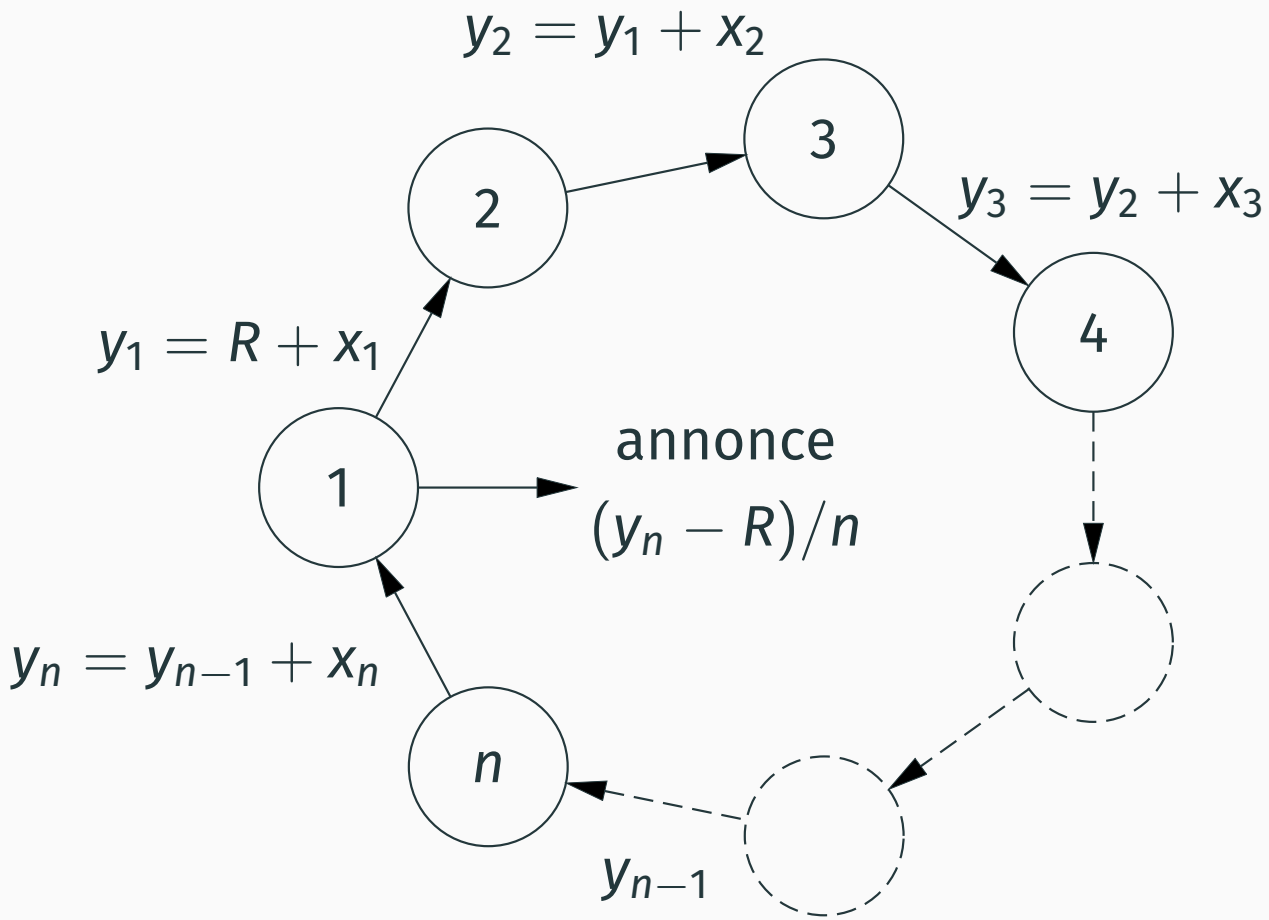
Chaque participant $i = 2, \ldots, n$ reçoit y_{i-1}, ajoute x_i, et transmet cette somme au participant suivant $i + 1$.

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$



Chaque participant $i = 2, \ldots, n$ reçoit y_{i-1}, ajoute x_i, et transmet cette somme au participant suivant $i + 1$.

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$



Le participant n reçoit y_n, calcule $(y_n - R)/n$, (dé-masquage)
et annonce ce résultat.

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$

$$y_1 = R + x_1 \qquad R \text{ aléatoire, } R \gg x_1, \ldots, x_n$$
$$y_i = y_{i-1} + x_i \qquad \text{pour } i = 2, \ldots, n$$

Correction : on a

$$y_i = R + x_1 + \cdots + x_i$$

d'où

$$\frac{y_n - R}{n} = \frac{x_1 + \cdots + x_n}{n}$$

Calcul multipartite de la moyenne de $x_1, \ldots, x_n$

$$y_1 = R + x_1 \qquad R \text{ aléatoire, } R \gg x_1, \ldots, x_n$$
$$y_i = y_{i-1} + x_i \qquad \text{pour } i = 2, \ldots, n$$

Confidentialité : le nombre y_i reçu par le participant $i+1$ est de la forme «grand nombre aléatoire + petit nombre» («masquage»), et donc ne révèle rien sur le «petit nombre» $x_1 + \ldots + x_i$, *a fortiori* rien sur chaque $x_1, \ldots, x_i$.

Le nombre y_n reçu par le participant 1 révèle $x_1 + \cdots + x_n$ mais rien de plus sur chaque x_i.

Quelques attaques possibles

Attaque passive par écoute des messages :

- Un attaquant qui peut écouter les messages y_{i-1} et y_i reçus / émis par le participant i en déduit $x_i = y_i - y_{i-1}$.

Contre-mesure : chiffrer les communications entre participants.

Quelques attaques possibles

Attaque passive par écoute des messages :

- Un attaquant qui peut écouter les messages y_{i-1} et y_i reçus / émis par le participant i en déduit $x_i = y_i - y_{i-1}$.

Contre-mesure : chiffrer les communications entre participants.

Attaque active par des participants malveillants :

- Collusion entre les participants i et $i+2$:
  $P(i)$ envoie 0 à $P(i+1)$; $P(i+2)$ reçoit x_{i+1}.
- Choix malveillant de R par le participant 1 : p.ex. R assez grand pour provoquer un débordement et plantage de $P(2)$.

Contre-mesure : programmation défensive de chaque participant.

Généralisation

Le même algorithme permet le calcul multipartite sécurisé de $\sum f(x_i)$ et $\prod f(x_i)$ pour toute fonction f. En particulier :

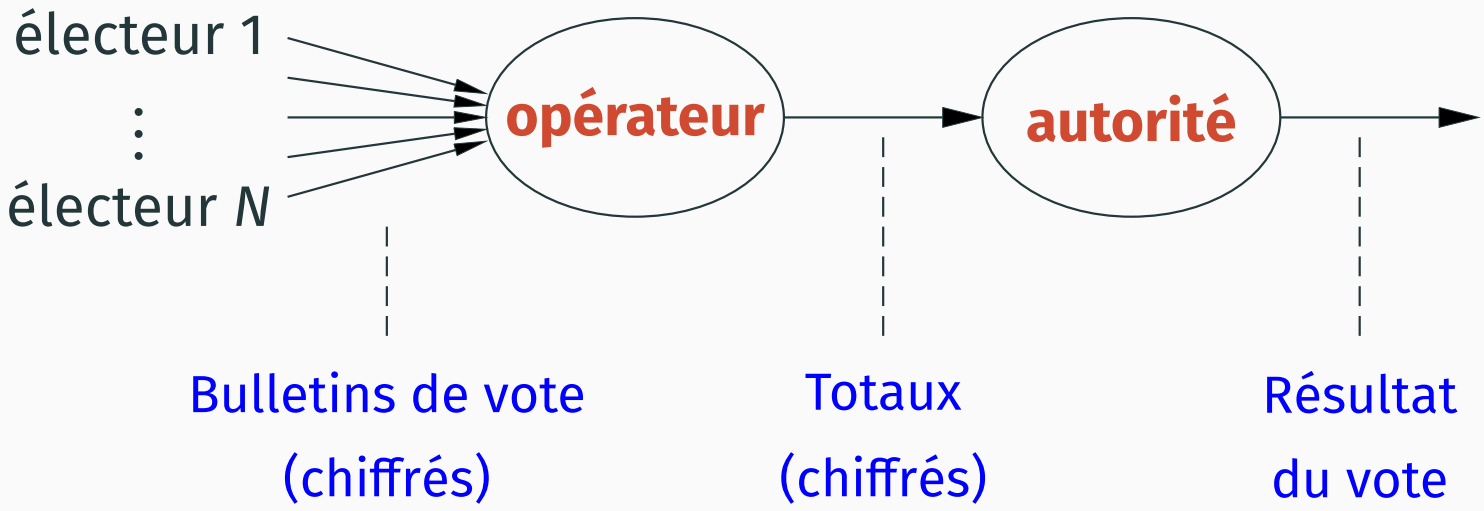
- Moyennes arithmétique, harmonique, géométrique, d'ordre p, avec ou sans pondération. $\sqrt[p]{\frac{1}{n}\sum x_i^p}$
- Variance et écart-type. $\sum x_i^2 - (\sum x_i)^2$
- Une approximation du maximum : la moyenne d'ordre p quand p est grand.

Ne permet pas de calculer l'indice i du plus grand des x_i.

(Le ≪problème des millionaires≫ de Yao $\rightarrow$ 5$^{\text{e}}$ cours)

Étude de cas : dépouillement d’un vote électronique simplifié

Un exemple de vote électronique



L'autorité du vote a la clé de déchiffrement mais n'a pas accès aux bulletins.

L'opérateur du vote collecte les bulletins mais n'a pas la clé de déchiffrement.

$\rightarrow$ Calcul des totaux par chiffrement homomorphe.

Une implémentation naïve de ce protocole

Chiffrement utilisé : RSA (homomorphe pour la multiplication).

L'électeur i encode son vote v_i comme

$$v_i = 2 \text{ pour Alice} \quad v_i = 3 \text{ pour Bob} \quad v_i = 1 \text{ pour blanc}$$

et chiffre son bulletin : $b_i = \mathcal{E}_{pk}(v_i)$.

L'opérateur collecte les bulletins b_i et calcule le produit
$T = b_1 \cdots b_n \pmod{N}$. ($N$ = module de la clé RSA)

L'autorité déchiffre T et factorise :

$$\mathcal{D}_{sk}(T) = 2^A \cdot 3^B$$

A est le nombre de voix pour Alice, B le nombre de voix pour Bob.

De nombreux problèmes !

Un électeur peut envoyer plusieurs bulletins.

L'opérateur peut ajouter des bulletins (≪bourrage d'urne≫).

De nombreux problèmes !

Un électeur peut envoyer plusieurs bulletins.
L'opérateur peut ajouter des bulletins (≪bourrage d'urne≫).

Les bulletins ne sont pas secrets : RSA étant déterministe,
on retrouve v_i en comparant b_i avec $\mathcal{E}_{pk}(1)$, $\mathcal{E}_{pk}(2)$ et $\mathcal{E}_{pk}(3)$.

De nombreux problèmes !

Un électeur peut envoyer plusieurs bulletins.
L'opérateur peut ajouter des bulletins (≪bourrage d'urne≫).

Les bulletins ne sont pas secrets : RSA étant déterministe,
on retrouve v_i en comparant b_i avec $\mathcal{E}_{pk}(1)$, $\mathcal{E}_{pk}(2)$ et $\mathcal{E}_{pk}(3)$.

Un électeur peut ≪bourrer≫ son bulletin, p.ex.
$v_i = 4$ (deux voix pour Alice) ou $v_i = 27$ (trois voix pour Bob).

De nombreux problèmes !

Un électeur peut envoyer plusieurs bulletins.
L'opérateur peut ajouter des bulletins (≪bourrage d'urne≫).

Les bulletins ne sont pas secrets : RSA étant déterministe,
on retrouve v_i en comparant b_i avec $\mathcal{E}_{pk}(1)$, $\mathcal{E}_{pk}(2)$ et $\mathcal{E}_{pk}(3)$.

Un électeur peut ≪bourrer≫ son bulletin, p.ex.
$v_i = 4$ (deux voix pour Alice) ou $v_i = 27$ (trois voix pour Bob).

Le décompte des bulletins peut facilement déborder :

$$\mathcal{D}_{sk}(T) = v_1 \cdots v_n \bmod N \neq v_1 \cdots v_n \quad \text{si } v_1 \cdots v_n \geq N$$

Quelques solutions

Un électeur peut envoyer plusieurs bulletins.
L'opérateur peut ajouter des bulletins (≪bourrage d'urne≫).

Authentifier les électeurs.
Leur faire signer numériquement leur bulletin.

Quelques solutions

Un électeur peut envoyer plusieurs bulletins.
L'opérateur peut ajouter des bulletins (≪bourrage d'urne≫).

Authentifier les électeurs.
Leur faire signer numériquement leur bulletin.

Les bulletins ne sont pas secrets : RSA étant déterministe, on retrouve v_i en comparant b_i avec $\mathcal{E}(1)$, $\mathcal{E}(2)$ et $\mathcal{E}(3)$.

Utiliser un chiffrement ≪randomisé≫, de sorte que chiffrer plusieurs fois le même clair produit des chiffrés différents.

Quelques solutions

Débordement arithmétique dès que $v_1 \cdots v_n \geq N$.

Utiliser un chiffrement homomorphe pour l'addition (d'entiers ou de tuples d'entiers) plutôt que pour la multiplication :

$$\mathcal{E}(m_1 + m_2) = \mathcal{E}(m_1) \boxplus \mathcal{E}(m_2) \pmod{N}$$

Exprimer les votes comme $(1, 0)$ = Alice, $(0, 1)$ = Bob, $(0, 0)$ = blanc.

Le chiffrement d'ElGamal exponentiel (EEG)

Un groupe fini G d'ordre q engendré par g.

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

Chiffrement randomisé :

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

Homomorphe pour l'addition :

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

($\rightarrow 2^{\text{e}}$ cours)

Quelques solutions

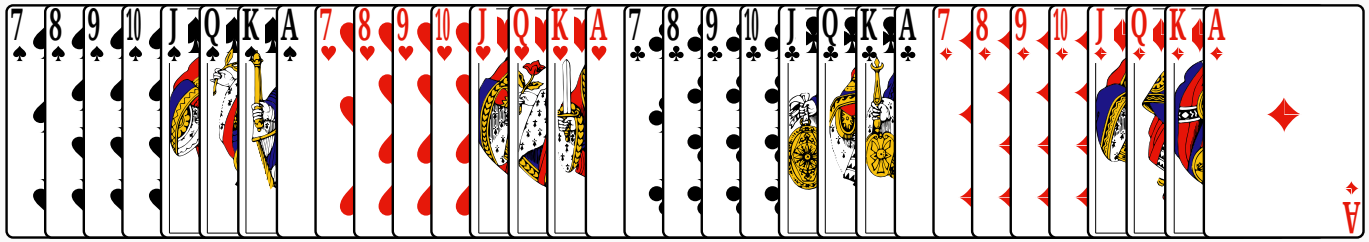
Un électeur peut «bourrer» son bulletin en chiffrant une value «impossible», p.ex. 4 (deux voix pour Alice).

Utilisation de preuves à divulgation nulle de connaissance (preuves *zero-knowledge*, ZK).

L'électeur fournit une preuve que son bulletin est bien $\mathcal{E}_{pk}(v)$ pour une valeur v autorisée par le vote, sans révéler la valeur v.

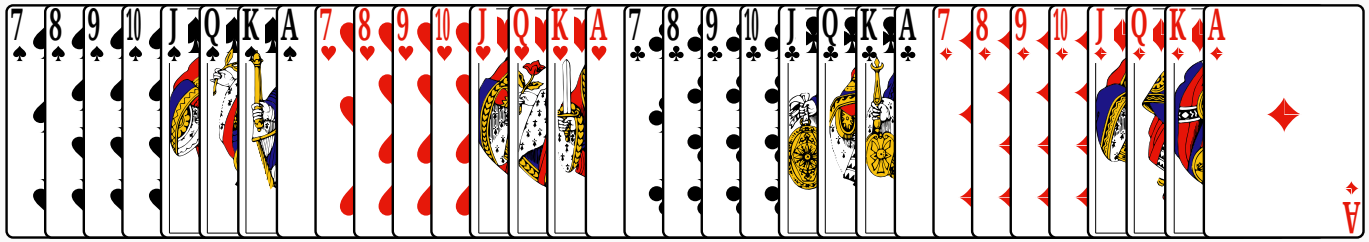
Un exemple de preuve *zero-knowledge*

Pierre (le prouveur) tire une carte à jouer et veut convaincre Véronique (la vérificatrice) que c'est une carte rouge, mais sans lui montrer la carte.



Un exemple de preuve *zero-knowledge*

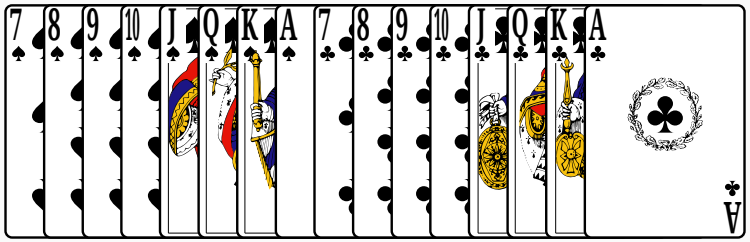
Pierre (le prouveur) tire une carte à jouer et veut convaincre Véronique (la vérificatrice) que c'est une carte rouge, mais sans lui montrer la carte.

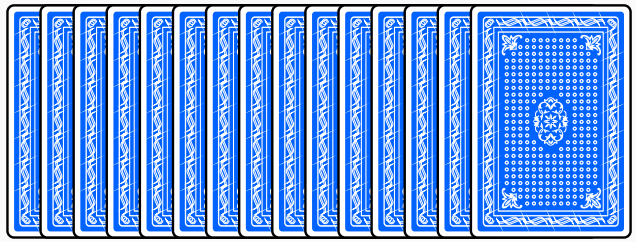


1. Pierre et Véronique vérifient le jeu de cartes :
16 rouges, 16 noires.

Un exemple de preuve *zero-knowledge*

Pierre (le prouveur) tire une carte à jouer et veut convaincre Véronique (la vérificatrice) que c'est une carte rouge, mais sans lui montrer la carte.

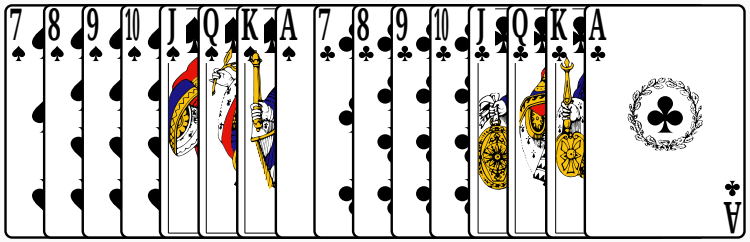


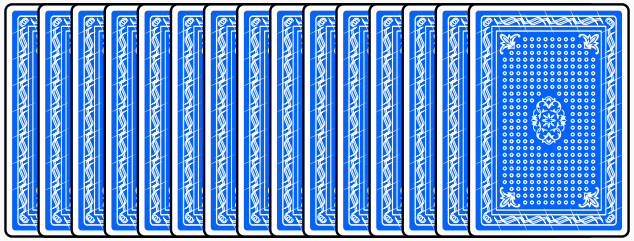


2. Pierre récupère les cartes, en garde une, pose les cartes restantes, et retourne 16 cartes noires.

Un exemple de preuve *zero-knowledge*

Pierre (le prouveur) tire une carte à jouer et veut convaincre Véronique (la vérificatrice) que c'est une carte rouge, mais sans lui montrer la carte.





3. Véronique est convaincue que la carte tirée par Pierre est rouge, mais ne sait rien de plus sur cette carte.

Preuves *zero-knowledge* pour EEG

Prouver que (A, B) est un chiffrement de 0 : (Chaum-Pedersen)

i.e. qu'il existe r tel que $(A, B) = (g^r, h^r)$, sans révéler r.

Engagement : Pierre publie un autre chiffrement de 0 $(A', B') = (g^u, h^u)$, en gardant u pour lui.

Défi : Véronique envoie un entier c aléatoire.

Réponse : Pierre envoie $t = u + cr \pmod{q}$.

Validation : Véronique vérifie que $g^t = A' \cdot A^c$ et $h^t = B' \cdot B^c$.

Preuves *zero-knowledge* pour EEG

Prouver que (A, B) **est un chiffrement de 0 :** (Chaum-Pedersen)

i.e. qu'il existe r tel que $(A, B) = (g^r, h^r)$, sans révéler r.

Prouver que (A, B) **est un chiffrement de 1 :**

i.e. qu'il existe r tel que $(A, B) = (g^r, h^r \cdot g)$, sans révéler r.

Une variante de l'algorithme précédent.

Preuves *zero-knowledge* pour EEG

Prouver que (A, B) **est un chiffrement de 0 :** (Chaum-Pedersen)

i.e. qu'il existe r tel que $(A, B) = (g^r, h^r)$, sans révéler r.

Prouver que (A, B) **est un chiffrement de 1 :**

i.e. qu'il existe r tel que $(A, B) = (g^r, h^r \cdot g)$, sans révéler r.

Prouver que (A, B) **est un chiffrement de 0 ou 1 :**

Une construction générique : étant donnés deux protocoles ≪sigma≫, un pour P et un pour Q, on obtient un protocole ≪sigma≫ pour $P \vee Q$.

($\rightarrow$ 6e cours)

Étude de cas : intersection de deux ensembles privés

Le problème PSI (*Private Set Intersection*)

Un client a un ensemble de noms C.
Un serveur a un ensemble de noms S.

Le client souhaite calculer $C \cap S$
i.e. apprendre lesquels de ses noms sont connus du serveur.

Le serveur ne doit rien apprendre sur C :

- ni les noms nouveaux pour lui ($C \setminus S$)
- ni les noms en commun ($C \cap S$).

Le client ne doit rien apprendre sur $S \setminus C$.

Applications du problème PSI

Découverte de contacts sur les réseaux sociaux :

Le client envoie son carnet d'adresses (tél, email).

Le serveur répond avec la liste des contacts qui sont déjà membre du réseau social (et qui ont accepté d'être ≪découvrables≫).

Applications du problème PSI

Découverte de contacts sur les réseaux sociaux :

Détection de mots de passe piratés :

Le client envoie ses identifiants et mots de passe (hachés).

Le serveur calcule l'intersection avec les listes de comptes piratés qui circulent sur le Web.

Applications du problème PSI

Découverte de contacts sur les réseaux sociaux :

Détection de mots de passe piratés :

Détection d'images dont la détention est illégale :

Le client envoie des empreintes des nouvelles images.

Le serveur détermine celles qui figurent dans les bases d'images illégales de la police.

Le client bloque les images illégales.

Solutions non satisfaisantes

Le client envoie la liste C en clair.

- Le serveur apprend C.

Solutions non satisfaisantes

Le client envoie la liste C en clair.

- Le serveur apprend C.

Le client envoie la liste C après hachage.

- I.e. le client envoie $\{H(c) \mid c \in C\}$ et le serveur calcule l'intersection avec $\{H(s) \mid s \in S\}$.
- Le serveur apprend $C \cap S$.
- Si l'espace des noms est petit (numéros de téléphone), le serveur peut précalculer $H(x)$ pour tout x, et donc apprendre $C \setminus S$ aussi.

Solutions non satisfaisantes

Le client envoie la liste C en clair.

- Le serveur apprend C.

Le client envoie la liste C après hachage.

- I.e. le client envoie $\{H(c) \mid c \in C\}$ et le serveur calcule l'intersection avec $\{H(s) \mid s \in S\}$.
- Le serveur apprend $C \cap S$.
- Si l'espace des noms est petit (numéros de téléphone), le serveur peut précalculer $H(x)$ pour tout x, et donc apprendre $C \setminus S$ aussi.

Le serveur envoie un filtre de Bloom pour S.

- Révèle trop d'informations sur S au client.
- Si le filtre est chiffré : trop coûteux.

Évaluation homomorphe d'un polynôme

Le client peut construire un polynôme P dont les racines sont exactement les éléments c_i de sa liste C :

$$P = (X - c_1)(X - c_2) \cdots (X - c_n) = \sum_{i=0}^{n} a_i X^i$$

On se donne un chiffrement homomorphe pour l'addition et la multiplication par une constante :

$$\mathcal{E}(x + x') = \mathcal{E}(x) \boxplus \mathcal{E}(x') \qquad \mathcal{E}(n \cdot x) = n \boxdot \mathcal{E}(x)$$

On peut alors évaluer homomorphiquement le polynôme P :

$$\begin{aligned} & x^n \boxdot \mathcal{E}(a_n) \boxplus \cdots \boxplus x \boxdot \mathcal{E}(a_1) \boxplus \mathcal{E}(a_0) \\ = \quad & \mathcal{E}(x^n \cdot a_n + \cdots + x \cdot a_1 + a_0) \;=\; \mathcal{E}(P(x)) \end{aligned}$$

Le protocole de Freedman, Nissim, Pinkas (2004)

(*Efficient Private Matching and Set Intersection*, Eurocrypt 2004, LNCS 3027.)

1. Le client choisit une clé (pk, sk),
   forme le polynôme $P = (X - c_1) \cdots (X - c_n) = \sum a_i X^i$,
   et transmet pk et les coefficients chiffrés $\mathcal{E}(a_i)$ au serveur.

Le protocole de Freedman, Nissim, Pinkas (2004)

(*Efficient Private Matching and Set Intersection*, Eurocrypt 2004, LNCS 3027.)

1. Le client choisit une clé (pk, sk),
   forme le polynôme $P = (X - c_1) \cdots (X - c_n) = \sum a_i X^i$,
   et transmet pk et les coefficients chiffrés $\mathcal{E}(a_i)$ au serveur.

2. Pour chaque $s_i \in S$, le serveur calcule homomorphiquement

   $$y_i = \mathcal{E}(r_i \cdot P(s_i) + s_i) \qquad \text{avec } r_i \text{ grand entier aléatoire}$$

   et transmet tous ces chiffrés y_i au client.

Le protocole de Freedman, Nissim, Pinkas (2004)

(*Efficient Private Matching and Set Intersection*, Eurocrypt 2004, LNCS 3027.)

1. Le client choisit une clé (pk, sk),
   forme le polynôme $P = (X - c_1) \cdots (X - c_n) = \sum a_i X^i$,
   et transmet pk et les coefficients chiffrés $\mathcal{E}(a_i)$ au serveur.

2. Pour chaque $s_i \in S$, le serveur calcule homomorphiquement

   $$y_i = \mathcal{E}(r_i \cdot P(s_i) + s_i) \qquad \text{avec } r_i \text{ grand entier aléatoire}$$

   et transmet tous ces chiffrés y_i au client.

3. Le client déchiffre les messages y_i et conserve ceux qui appartiennent à C :

   $$C \cap S = \{\mathcal{D}(y_i) \mid \mathcal{D}(y_i) \in C\}$$

Le protocole de Freedman, Nissim, Pinkas (2004)

Correction. Le déchiffrement $x_i = \mathcal{D}(y_i)$ est de la forme

$$x_i = r_i \cdot P(s_i) + s_i \quad \text{avec } s_i \in S \text{ (inconnu) et } r_i \text{ aléatoire (inconnu)}$$

Si $s_i \in C$, on a $P(s_i) = 0$, donc $x_i = s_i$ et le test $x_i \in C$ réussit.

Si $s_i \notin C$, on a $P(s_i) \neq 0$, donc x_i est aléatoire et très probablement le test $x_i \in C$ échoue.

Le protocole de Freedman, Nissim, Pinkas (2004)

Correction. Le déchiffrement $x_i = \mathcal{D}(y_i)$ est de la forme

$$x_i = r_i \cdot P(s_i) + s_i \quad \text{avec } s_i \in S \text{ (inconnu) et } r_i \text{ aléatoire (inconnu)}$$

Si $s_i \in C$, on a $P(s_i) = 0$, donc $x_i = s_i$ et le test $x_i \in C$ réussit.

Si $s_i \notin C$, on a $P(s_i) \neq 0$, donc x_i est aléatoire et très probablement le test $x_i \in C$ échoue.

Confidentialité. Le serveur n'apprend rien sur les c_i, puisque les coefficients du polynôme sont chiffrés.

Le client n'apprend rien sur les $s_i \notin C$, puisque les messages y_i correspondants se déchiffrent en du bruit.

Réduire la complexité du protocole

Le polynôme P est de degré $|C|$, donc son évaluation prend un temps $\mathcal{O}(|C|)$ et la phase 2 du protocole prend un temps $\mathcal{O}(|C| \cdot |S|)$.

On peut partitionner C en B morceaux de taille $\approx |C|/B$ à l'aide d'une fonction de hachage (non cryptographique) choisie par le client.

Le serveur reçoit B polynômes de degré $|C|/B$, d'où un temps $\mathcal{O}(\frac{|C| \cdot |S|}{B})$ pour la phase 2.

Freedman *et al.* proposent un schéma de hachage qui reste sûr avec $B = |C|/\ln\ln|C|$, d'où une complexité en $\mathcal{O}(|S| \ln\ln|C|)$.

Bibliographie

Bibliographie

Background général en cryptographie :

- *La cryptographie déchiffrée : une introduction pratique au chiffrement moderne*, Jean-Philippe Aumasson, Dunod, 2024.
- *Cryptography made simple*, Nigel P. Smart, Springer, 2016.

Sur le vote électronique :

- *Le vote électronique : les défis du secret et de la transparence*, Véronique Cortier et Pierrick Gaudry, Odile Jacob, 2022.

Le protocole PSI décrit dans ce cours :

- Michael J. Freedman, Kobbi Nissim et Benny Pinkas, “Efficient Private Matching and Set Intersection”, *Advances in cryptology – Eurocrypt 2004*, LNCS 3027, Springer, 2004.
  https://doi.org/10.1007/978-3-540-24676-3_1