Université de Bordeaux
Collège Sciences et Technologies
Master 1 - Cryptographie et Sécurité Informatique

Protocole d'interrogation anonyme de base de données.

Réalisé par :
AKA Leaticia,
DIALLO Mamadou Aliou,
SADICK Youssouf Charaf

Encadré par :
M. Guilhem Castagnos

Année Scolaire 2024/2025

Plan

Introduction

- Comment permettre à un utilisateur de récupérer une donnée d'une base sans que le serveur sache quelle donnée a été consultée ?

Introduction

- Comment permettre à un utilisateur de récupérer une donnée d'une base sans que le serveur sache quelle donnée a été consultée ?
- Les protocoles de Private Information Retrieval (PIR).

Un peu d'histoire

Origines du PIR

- Le concept de *Private Information Retrieval* (PIR) a été introduit en 1995 par Chor, Goldreich, Kushilevitz et Sudan.
- Les premiers protocoles utilisaient plusieurs serveurs avec des copies de la base, supposés ne pas partager leurs informations.
- Ces méthodes étaient théoriques.
- En 1997, Kushilevitz et Ostrovsky ont proposé le premier protocole PIR à base de données unique.
- Les protocoles modernes exploitent des techniques cryptographiques comme le chiffrement homomorphe (ex. Paillier) pour améliorer l'efficacité.

Plan

Chiffrement homomorphe

C'est quoi le chiffrement homomorphe ?

Le chiffrement homomorphe permet de réaliser des opérations sur des données **chiffrées** sans nécessiter leur déchiffrement préalable.

Chiffrement homomorphe

C'est quoi le chiffrement homomorphe ?

Le chiffrement homomorphe permet de réaliser des opérations sur des données **chiffrées** sans nécessiter leur déchiffrement préalable.

Deux grandes catégories

- **FHE** — Fully Homomorphic Encryption
- **PHE** — Partial Homomorphic Encryption :
  - Ne supporte qu'un **sous-ensemble d'opérations** :
    - Addition (ex. Paillier, Goldwasser–Micali)
    - Multiplication (ex. RSA, ElGamal)

$$\prod_{i=1}^{l} E(\delta_{i,j})^{m_i} = E\left(\sum_{i=1}^{l} \delta_{i,j} \cdot m_i\right) = E(m_j), \quad \text{où } \delta_{i,j} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

$$\text{Si } c_1 = E(m_1),\ c_2 = E(m_2) \Rightarrow$$

$$c_1 \cdot c_2 \equiv E(m_1 + m_2) \mod n^2, \quad c_1^k \equiv E(k \cdot m_1) \mod n^2$$

Chiffrement de Paillier

Génération des clefs

- Choisir deux grands nombres premiers p, q, calculer $n = p \cdot q$.
- Calculer $\lambda = \mathrm{lcm}(p-1, q-1)$, puis choisir $g = n + 1$.
- Clé publique : (n, g), Clé privée : λ.

Chiffrement de Paillier

Génération des clefs

- Choisir deux grands nombres premiers p, q, calculer $n = p \cdot q$.
- Calculer $\lambda = \mathrm{lcm}(p-1, q-1)$, puis choisir $g = n + 1$.
- Clé publique : (n, g), Clé privée : λ.

Chiffrement

$$c = g^m \cdot r^n \mod n^2 \quad \text{avec } r \in \mathbb{Z}_n^*, \text{ et } m \in \mathbb{Z}_n$$

Chiffrement de Paillier

Génération des clefs

- Choisir deux grands nombres premiers p, q, calculer $n = p \cdot q$.
- Calculer $\lambda = \mathrm{lcm}(p-1, q-1)$, puis choisir $g = n + 1$.
- Clé publique : (n, g), Clé privée : λ.

Chiffrement

$$c = g^m \cdot r^n \mod n^2 \quad \text{avec } r \in \mathbb{Z}_n^*, \text{ et } m \in \mathbb{Z}_n$$

Déchiffrement

$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n, \quad L(x) = \frac{x-1}{n}$$

Déchiffrement - Démonstration (1/3)

Le déchiffrement utilise la clé privée λ et est défini par :

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n, \quad \text{avec } L(x) = \frac{x-1}{n}$$

Déchiffrement - Démonstration (1/3)

Le déchiffrement utilise la clé privée λ et est défini par :

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n, \quad \text{avec } L(x) = \frac{x-1}{n}$$

- Soit $c = g^m \cdot r^n \bmod n^2$, avec $r \in \mathbb{Z}_n^*$ et $m \in \mathbb{Z}_n$

Déchiffrement - Démonstration (1/3)

Le déchiffrement utilise la clé privée λ et est défini par :

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n, \quad \text{avec } L(x) = \frac{x-1}{n}$$

- Soit $c = g^m \cdot r^n \bmod n^2$, avec $r \in \mathbb{Z}_n^*$ et $m \in \mathbb{Z}_n$
- On calcule :

$$c^\lambda = (g^m \cdot r^n)^\lambda \bmod n^2 = g^{m\lambda} \cdot r^{n\lambda} \bmod n^2$$

Déchiffrement - Démonstration (1/3)

Le déchiffrement utilise la clé privée λ et est défini par :

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n, \quad \text{avec } L(x) = \frac{x-1}{n}$$

- Soit $c = g^m \cdot r^n \bmod n^2$, avec $r \in \mathbb{Z}_n^*$ et $m \in \mathbb{Z}_n$
- On calcule :

$$c^\lambda = (g^m \cdot r^n)^\lambda \bmod n^2 = g^{m\lambda} \cdot r^{n\lambda} \bmod n^2$$

- Comme $r \in \mathbb{Z}_n^*$ et λ est un multiple de $\varphi(n)$, on a :

$$r^{\lambda n} \equiv 1 \bmod n^2$$

Déchiffrement - Démonstration (1/3)

Le déchiffrement utilise la clé privée λ et est défini par :

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n, \quad \text{avec } L(x) = \frac{x-1}{n}$$

- Soit $c = g^m \cdot r^n \bmod n^2$, avec $r \in \mathbb{Z}_n^*$ et $m \in \mathbb{Z}_n$
- On calcule :

$$c^\lambda = (g^m \cdot r^n)^\lambda \bmod n^2 = g^{m\lambda} \cdot r^{n\lambda} \bmod n^2$$

- Comme $r \in \mathbb{Z}_n^*$ et λ est un multiple de $\varphi(n)$, on a :

$$r^{\lambda n} \equiv 1 \bmod n^2$$

- D'où : $c^\lambda = g^{m\lambda} \bmod n^2$

Déchiffrement - Démonstration (2/3)

On applique ensuite la fonction L :

$$L(c^\lambda \bmod n^2) = \frac{c^\lambda - 1}{n} = \frac{g^{\lambda m} - 1}{n}$$

Déchiffrement - Démonstration (2/3)

On applique ensuite la fonction L :

$$L(c^\lambda \bmod n^2) = \frac{c^\lambda - 1}{n} = \frac{g^{\lambda m} - 1}{n}$$

De plus, $\quad L(g^\lambda \bmod n^2) = \frac{g^\lambda - 1}{n}$

Déchiffrement - Démonstration (2/3)

On applique ensuite la fonction L :

$$L(c^\lambda \bmod n^2) = \frac{c^\lambda - 1}{n} = \frac{g^{\lambda m} - 1}{n}$$

De plus, $L(g^\lambda \bmod n^2) = \frac{g^\lambda - 1}{n}$

Ainsi, pour $g = n + 1$:

$$L(c^\lambda \bmod n^2) = \frac{(n+1)^{\lambda m} - 1}{n}, \quad L(g^\lambda \bmod n^2) = \frac{(n+1)^\lambda - 1}{n}$$

Déchiffrement - Démonstration (2/3)

On applique ensuite la fonction L :

$$L(c^\lambda \bmod n^2) = \frac{c^\lambda - 1}{n} = \frac{g^{\lambda m} - 1}{n}$$

De plus, $L(g^\lambda \bmod n^2) = \frac{g^\lambda - 1}{n}$

Ainsi, pour $g = n + 1$:

$$L(c^\lambda \bmod n^2) = \frac{(n+1)^{\lambda m} - 1}{n}, \quad L(g^\lambda \bmod n^2) = \frac{(n+1)^\lambda - 1}{n}$$

Le rapport donne :

$$\frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} = \frac{\frac{(n+1)^{\lambda m} - 1}{n}}{\frac{(n+1)^\lambda - 1}{n}} = \frac{(n+1)^{\lambda m} - 1}{(n+1)^\lambda - 1}$$

Déchiffrement - Démonstration (3/3)

- Or $(n+1)^{\lambda m} = 1 + (m\lambda)n \bmod n^2$ et $(n+1)^{\lambda} = 1 + \lambda n \bmod n^2$

$$\frac{L(c^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} = \frac{1 + \lambda mn - 1}{1 + \lambda n + -1} = \frac{\lambda mn}{\lambda n} = m$$

Déchiffrement - Démonstration (3/3)

- Or $(n+1)^{\lambda m} = 1 + (m\lambda)n \bmod n^2$ et $(n+1)^{\lambda} = 1 + \lambda n \bmod n^2$

$$\frac{L(c^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} = \frac{1 + \lambda mn - 1}{1 + \lambda n + -1} = \frac{\lambda mn}{\lambda n} = m$$

- D'où le résultat :

$$m = \frac{L(c^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n$$

Sécurité du chiffrement de paillier

Sécurité

Le chiffrement de Paillier repose sur l'**hypothèse de résidualité composite (DCRA)** :

- Sémantiquement sûr.

Sécurité du chiffrement de paillier

Sécurité

Le chiffrement de Paillier repose sur l'**hypothèse de résidualité composite (DCRA)** :

- Sémantiquement sûr.
- Homomorphique additif.

Sécurité du chiffrement de paillier

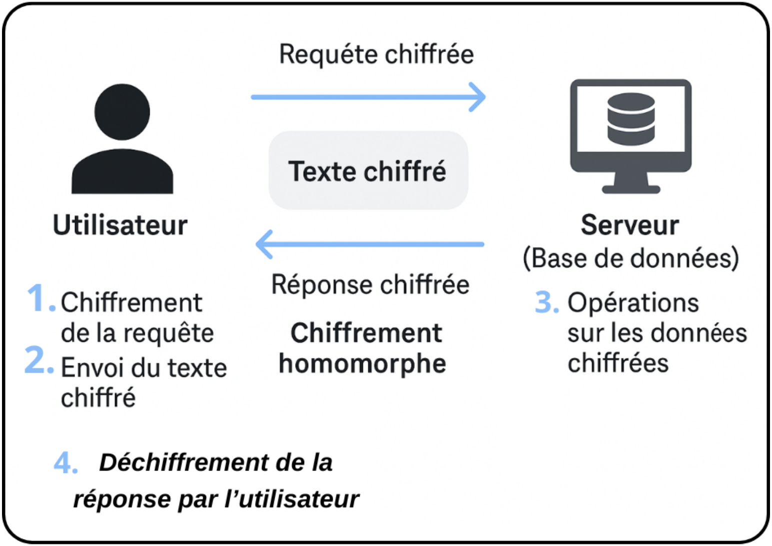
Sécurité

Le chiffrement de Paillier repose sur l'**hypothèse de résidualité composite (DCRA)** :

- Sémantiquement sûr.
- Homomorphique additif.
- Self-blindable.

Plan

Principe général



PIR 2D - init

On considère une base de données de taille $l \times l$, où i et j parcourent respectivement les lignes et les colonnes et $i, j \in [l]$ où les éléments de x sont pris dans $\mathbb{Z}_n$. Le client veut obtenir l'élément $x(i^*, j^*)$.

$$x = \begin{bmatrix} 1 & 0 & 3 \\ 2 & 1 & 4 \\ 4 & 2 & 1 \end{bmatrix} \quad I = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad J = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Initialisation

Le client génère deux vecteurs de requêtes α_t et β_t:

$$\alpha_t = \begin{bmatrix} E(0), & E(1), & E(0) \end{bmatrix} \quad \beta_t = \begin{bmatrix} E(0), & E(0), & E(1) \end{bmatrix}$$

PIR 2D - Opérations homomorphes (1/2)

Calcul du produit σ_i

Le serveur calcule pour chaque ligne t :

$$\sigma_i = \prod_{t=0}^{l-1} (\beta_t)^{x_{i,t}} \mod n^2$$

$$x = \begin{bmatrix} 1 & 0 & 3 \\ 2 & 1 & 4 \\ 4 & 2 & 1 \end{bmatrix}$$

$$\sigma_1 = E(0)^1 \cdot E(0)^0 \cdot E(1)^3 = 75456207059$$
$$\sigma_2 = E(0)^2 \cdot E(0)^1 \cdot E(1)^4 = E(0 + 0 + 4)$$
$$\sigma_2 = E(4) = 59900541076$$
$$\sigma_3 = E(0)^4 \cdot E(0)^2 \cdot E(1)^1 = 36364248569$$

PIR 2D - Opérations homomorphes (2/2)

Séparation et filtrage final

- Le serveur sépare chaque σ_i en deux parties u_i et v_i :

$$\sigma_i = u_i \cdot n + v_i$$

- il calcule u et v qu'il envoie au client :

$$u = \prod_{t=0}^{l-1} (\alpha_t)^{u_t} \mod n^2 \qquad v = \prod_{t=0}^{l-1} (\alpha_t)^{v_t} \mod n^2$$

Avec l'exemple précédent, on obtient :

$$u = 39057682565 \quad et \quad v = 63082257868$$

PIR 2D - Réconstruction

L'utilisateur récupère $x(i^*, j^*)$ en appliquant le déchiffrement de Paillier :

$$x(i^*, j^*) = D_g(D_g(u) \cdot n + D_g(v))$$

$$c_1 = D_g(u) = 200273 \quad et \quad c_2 = D_g(v) = 288687$$

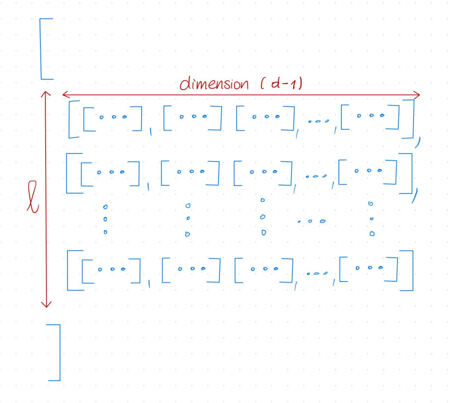
Puis,

$$x(i^*, j^*) = D_g(c_1 \cdot n + c_2) = 4$$

PIR à d dimensions (1/3)

Comment ça marche ?

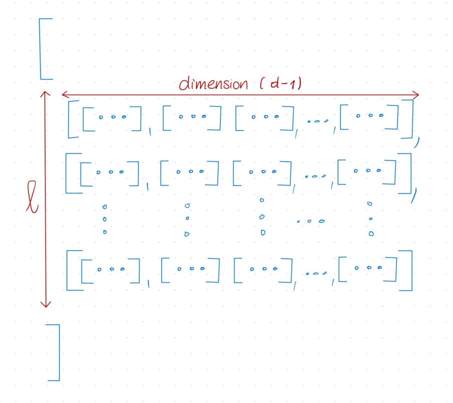
Soit x une base de données de dimension d et de taille l^d dont les éléments sont pris dans $\mathbb{Z}_n$:



PIR à d dimensions (1/3)

Comment ça marche ?

Soit x une base de données de dimension d et de taille l^d dont les éléments sont pris dans $\mathbb{Z}_n$:

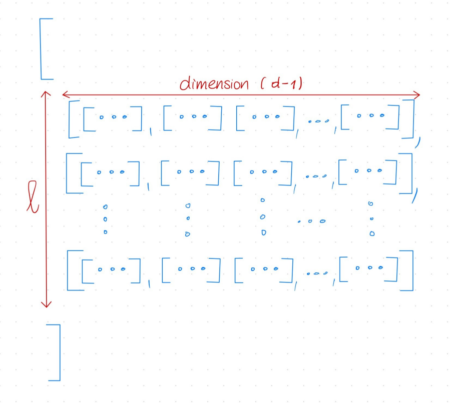


- Initialisation des requêtes : $x(i_1^*, i_2^*, \ldots, i_d^*) \in \mathbb{Z}_n$.

PIR à d dimensions (1/3)

Comment ça marche ?

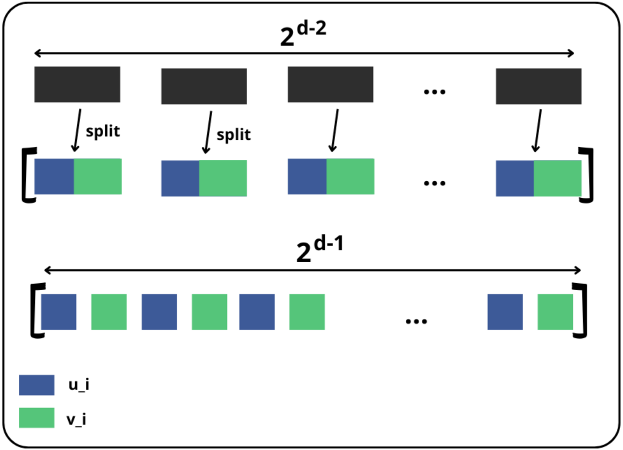
Soit x une base de données de dimension d et de taille l^d dont les éléments sont pris dans $\mathbb{Z}_n$:



- Initialisation des requêtes : $x(i_1^*, i_2^*, \ldots, i_d^*) \in \mathbb{Z}_n$.
- Opérations du serveur :
  - l appels récursifs du PIR en dimension $d - 1$

PIR à *d* dimensions (2/3)

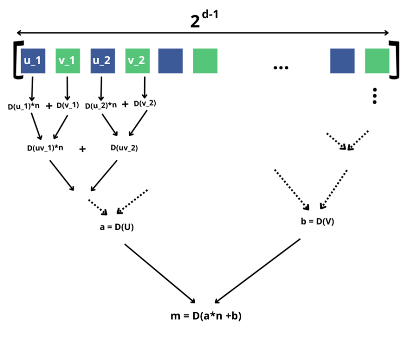
- Séparation et filtrage



PIR à d dimensions (3/3)

Reconstruction

Le client reçoit 2^{d-1} chiffrés en sortie du serveur. Il effectue alors $d-1$ étapes successives pour reconstituer l'élément $x(i_1^*, i_2^*, \ldots, i_d^*) \in \mathbb{Z}_n$.



- Regrouper les chiffrés par paires (u_i, v_i).
- Déchiffrer chaque valeur $Dg(u_i)$ et $Dg(v_i)$.
- Calculer à chaque étape :

$$Dg(u_i) \cdot n + Dg(v_i)$$

- Répéter le processus en $d-1$ étapes.
- À la fin, obtenir l'élément ciblé $m = x(i_1^*, \ldots, i_d^*)$.

Mise en œuvre du PIR

Opérations côté serveur

```
def server_op_rec_cD(dim, database, request, public_key):
    n, g = public_key
    n2, L = n**2, len(database)
    result = [1] * (2**(dim-1))

    if dim == 1:
        return server_op1D(database, request[-1], public_key)

    for d in range(L):
        res = server_op_rec_cD(dim-1, database[d], request[1:], public_key)
        for i in range(2**(dim-2)):
            u, v = ZZ(res[i]) // n, res[i] % n
            result[2*i]   = (result[2*i]   * pow(request[0][d], u, n2)) % n2
            result[2*i+1] = (result[2*i+1] * pow(request[0][d], v, n2)) % n2
    return result
```

Gmplib

Approche en C : basée sur GMP

Plan

Conclusion

- L’étude du chiffrement homomorphe de Paillier a montré qu’on peut effectuer des opérations sur des données chiffrées sans les déchiffrer.
- Le protocole PIR se base sur ce type de chiffrement pour préserver l’anonymat des interrogations à une base de données.
- Le PIR constitue une solution cryptographique efficace et évolutive pour les applications sensibles(ex : santé, finance, vote électronique, etc..).

Références

[1] Rafail Ostrovsky, William E. Skeith III, *A Survey of Single-Database PIR: Techniques and Applications*, https://eprint.iacr.org/2007/059.pdf

[2] Pascal Paillier, *Public Key Cryptosystems based on Composite Degree Residue Classes*, https://link.springer.com/content/pdf/10.1007/3-540-48910-X_16.pdf

[3] Yan-Cheng Chang, *Single Database Private Information Retrieval with Logarithmic Communication*, https://eprint.iacr.org/2004/036.pdf

[4] Craig Gentry, *Fully Homomorphic Encryption Using Ideal Lattices*, https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf

[5] Pascal Paillier, David Pointcheval, *Efficient public-key cryptosystems provably secure against active adversaries*, https://link.springer.com/chapter/10.1007/978-3-540-48000-6_14