Filtres à mot-clés secrets et réseaux euclidiens

Carlos Aguilar Melchor[2] **Laurent Fousse** [1] Philippe Gaborit [2]

[1] Laboratoire J. Kuntzmann
Université Grenoble 1

[2] XLIM
Université de Limoges

26 mars 2009

Plan

MATHÉMATIQUES APPLIQUÉES - INFORMATIQUE

Plan

Filtre à mot-clés secrets (*Private Searching*)

Définition informelle

Protocole permettant de faire une recherche parmi un flux de documents sans dévoiler sa requête.

Filtre à mot-clés secrets (*Private Searching*)

Définition informelle

Protocole permettant de faire une recherche parmi un flux de documents sans dévoiler sa requête.

Définition plus formelle

Un obfuscateur à clef publique transforme un programme f en un couple de programmes (F, Dec) tels que

$$Dec(F(x)) = f(x)$$

pour tout x et la donnée de F ne permet pas de distinguer f de la classe de programmes considérés pour un adversaire polynomial.

Plan

Exemple de filtre *non*-privé

Google Alerts

- Mot-clefs et adresse email soumis par formulaire ;
- Résultats envoyés périodiquements par email.

Google Alerts

Foire aux questions | Connexion

Google Alertes

Alertes Google (version Bêta)

Alertes Google - Bienvenue

Les Alertes Google sont envoyées par courrier électronique quand des articles publiés en ligne correspondent aux sujets que vous indiquez.

Exemples d'utilisation des Alertes Google :

- suivre l'évolution d'une affaire ;
- obtenir des informations à jour sur un concurrent ou un secteur d'activités ;
- connaître l'actualité d'une personnalité ;
- suivre les résultats d'une équipe sportive.

Créez une alerte à l'aide du formulaire situé sur votre droite.

Vous pouvez également vous **connecter pour gérer vos alertes**

Créer une Alerte Google

Entrez le sujet sur lequel vous souhaitez obtenir des informations.

Termes recherchés: rolex rater sa vie

Type: Tous les types

Fréquence: une fois par jour

Email: seguela@havas.com

Créer l'Alerte

Google s'engage à ne pas vendre et à ne pas communiquer votre adresse de messagerie.

MATHÉMATIQUES APPLIQUÉES - INFORMATIQUE

Google Alerts, mais aussi Google Mail

Gmail Calendar Documents Photos Reader Web more ▾ [redacted]@gmail.com | Settings | Older version | Help | Sign out

Gmail by Google BETA

Search Mail | Search the Web | Show search options | Create a filter

Compose Mail

Inbox
Starred
Chats
Sent Mail
Drafts
All Mail
Spam (360)
Trash

Contacts

[redacted]

Search, add, or invite

Labels
Edit labels

Invite a friend
Give Gmail to:
Send Invite 98 left
Preview Invite

« Back to Inbox | Archive | Report spam | Delete | Move to ▾ | Labels ▾ | More actions ▾ | 1 of 18 Older ›

Carrière de Carlita Inbox | X

Nicolas S. to me — show details 11:10 AM (0 minutes ago) — Reply

Salut Jacques,

Tu avances sur la promo du dernier CD de Carla ? On compte tous sur toi. Tu conserves toute ma confiance.

Nico.

Reply | Forward

New window
Print all

Sponsored Links (feedback)

$99 Luxury Watches
High Quality Guarantee
Christmas sales promotion, Buy Now
HandMadeRolex.com

ManifattureFirenze strap
Top quality leather watch straps & accessories fully Italian handmade
www.manifatturefirenze.it

Replica Panerai Watches
Sale Repliques Panerai , Breitling IWC, Chopard, Hublot, Montblanc
www.watch88.com/Panerai-Montres

Luxury Watches For Cheap
5000+ Styles For Luxury Watches 120% Low Prices. Free Shipping
www.Rolex2u.com

Luxury Watches (£87/$130)
UK & USA Based! Beautiful Watches, All With Money Back Guarantee.
www.thewatchdome.com

About these links

« Back to Inbox | Archive | Report spam | Delete | Move to ▾ | Labels ▾ | More actions ▾ | 1 of 18 Older ›

Filtre à mot-clés secrets (*Private Searching*)

Définition plus technique

Trois algorithmes :

- `QueryConstruction` : génère une requête Q à partir d'un ensemble de mots-clefs.

Filtre à mot-clés secrets (*Private Searching*)

Définition plus technique

Trois algorithmes :

- `QueryConstruction` : génère une requête Q à partir d'un ensemble de mots-clefs.
- `StreamSearch` : génère une réponse à partir de Q et des documents du flux.

Filtre à mot-clés secrets (*Private Searching*)

Définition plus technique

Trois algorithmes :

- `QueryConstruction` : génère une requête Q à partir d'un ensemble de mots-clefs.
- `StreamSearch` : génère une réponse à partir de Q et des documents du flux.
- `FileReconstruction` : déchiffre la réponse pour produire les documents correspondant à la requête.

Plan

Définition

Informelle

Protocole permettant à un utilisateur d'obtenir un élément d'une base de données en occultant auprès de celle-ci de quel élément il s'agit.

Définition

Informelle

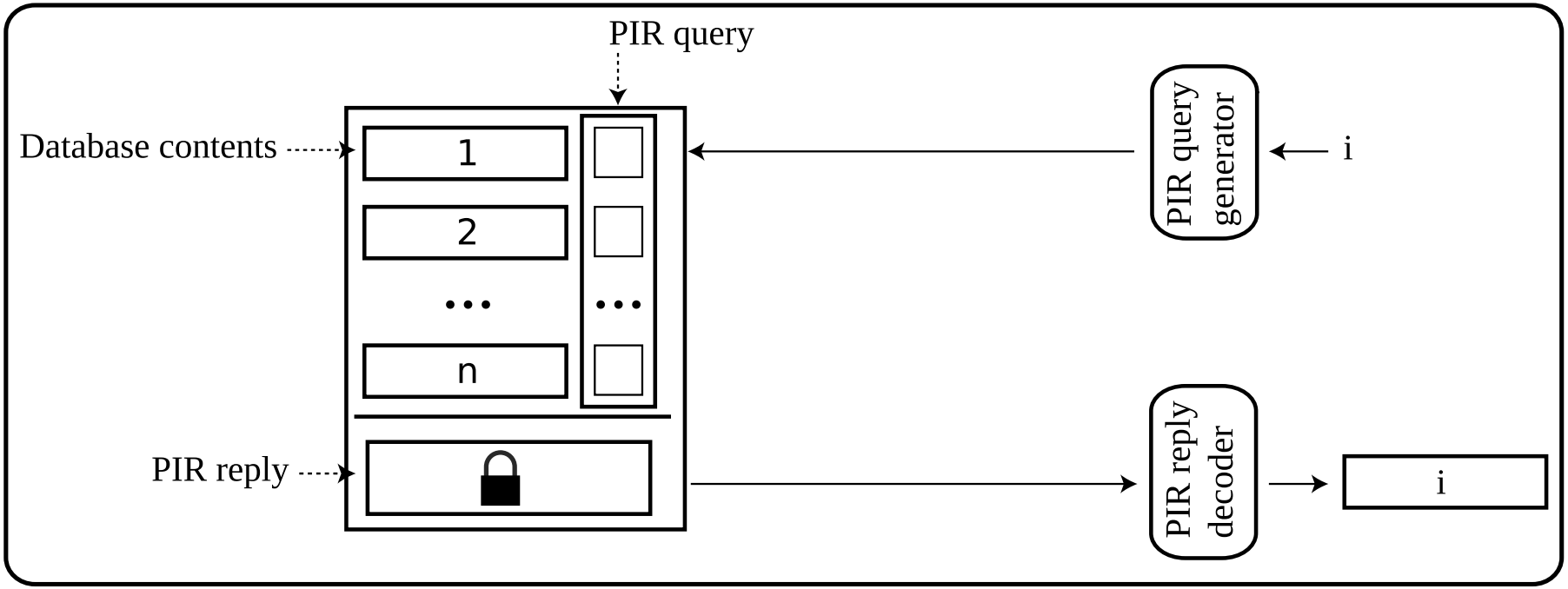
Protocole permettant à un utilisateur d'obtenir un élément d'une base de données en occultant auprès de celle-ci de quel élément il s'agit.

Un peu plus formelle

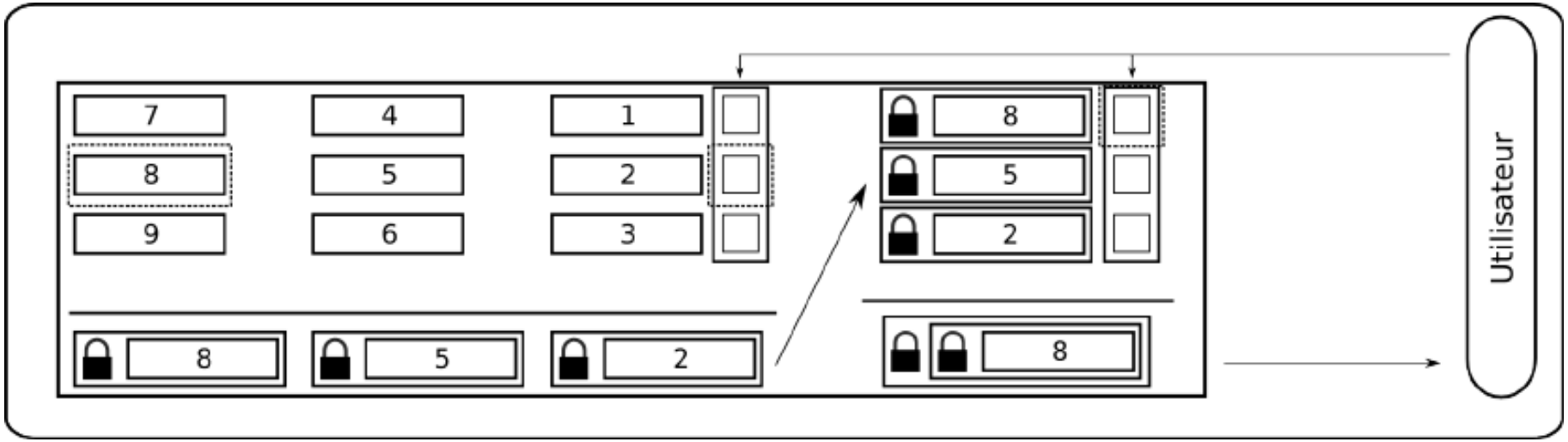
Un protocole est dit un protocole RIP si :

- il permet aux utilisateurs d'obtenir n'importe quel élément d'une base de données, quel que soit le contenu de celle-ci ;
- pour tout couples d'éléments e_1, e_2 de la base, les requêtes des utilisateurs correspondant à e_1 sont indistingables de celles correspondant à e_2.

Protocole PIR



Récursion



- La première requête permet d'obtenir une ligne.
- La deuxième requête opère sur la réponse générée.
  - Permet d'obtenir un élément de la ligne.
- Requête : $2\sqrt{n}$, facteur d'expansion de la réponse : $F_{RIP} * F_{RIP}$.
- Généralisation triviale $\Rightarrow$ requête : $d \times n^{1/d}$, facteur d'expansion : F_{RIP}^{d}.

Chiffrement homomorphe

Définition

$$D(E(m_1) \oplus E(m_2)) = m_1 \otimes m_2$$

où $\oplus$ et $\otimes$ désignent des opérations de groupe.

Chiffrement homomorphe

Définition

$$D(E(m_1) \oplus E(m_2)) = m_1 \otimes m_2$$

où $\oplus$ et $\otimes$ désignent des opérations de groupe.

Exemple : Paillier

- Clef publique : $n = pq$ entier RSA avec $p < q$, $p \nmid q - 1$.
- Chiffrement : $(m, r) \in \mathbb{Z}/n\mathbb{Z} \times (\mathbb{Z}/n\mathbb{Z})^* \mapsto c = (1+n)^m r^n \bmod n^2$.
- Déchiffrement : $c = r^n \bmod n$ donne r.
- Homomorphisme :

$$D(E(m_1) \times E(m_2)) = m_1 + m_2$$

Paillier

Keygen

```
def keygen(ks):
    todo = True
    while todo:
        p = random_prime(1 << ks)
        q = random_prime(1 << ks)
        if p > q:
            t = p
            p = q
            q = t
        if (q - 1) % p != 0:
            todo = False
    return (p, q, p * q)
```

Paillier

Encryption

```
def encrypt(m, n):
    r = randint(0, n - 1)
    while gcd(r, n) == 0:
        r = randint(0, n - 1)
    s = n * n
    u = Mod(1 + n, s)
    u = u^m
    r = Mod(r, s)
    r = r^n
    return u * r
```

Paillier

Decryption

```
def decrypt(c, n, p):
    u = Mod(c, n)
    s = n * n
    phi = (p - 1) * (n/p - 1)
    e = Mod(n, phi)
    d = 1 / e
    r = u^d
    v = Mod(r, s) ^ n
    t = Mod(c, s) / v
    return int(t - 1) / n
```

Paillier

Usage

```
sage : (p, q, n) = keygen(20)
```

Paillier

Usage

```
sage : (p, q, n) = keygen(20)
sage : (p, q, n)
(192557, 720089, 138658177573)
```

Paillier

Usage

```
sage : (p, q, n) = keygen(20)
sage : (p, q, n)
(192557, 720089, 138658177573)
sage : c1 = encrypt(1, n)
sage : c2 = encrypt(1, n)
```

Paillier

Usage

```
sage : (p, q, n) = keygen(20)
sage : (p, q, n)
(192557, 720089, 138658177573)
sage : c1 = encrypt(1, n)
sage : c2 = encrypt(1, n)
sage : c1
16768997473026340839836
sage : c2
14672963152557506378498
```

Paillier

Usage

```
sage : (p, q, n) = keygen(20)
sage : (p, q, n)
(192557, 720089, 138658177573)
sage : c1 = encrypt(1, n)
sage : c2 = encrypt(1, n)
sage : c1
16768997473026340839836
sage : c2
14672963152557506378498
sage : decrypt(c1, n, p)
1
```

Paillier

Usage

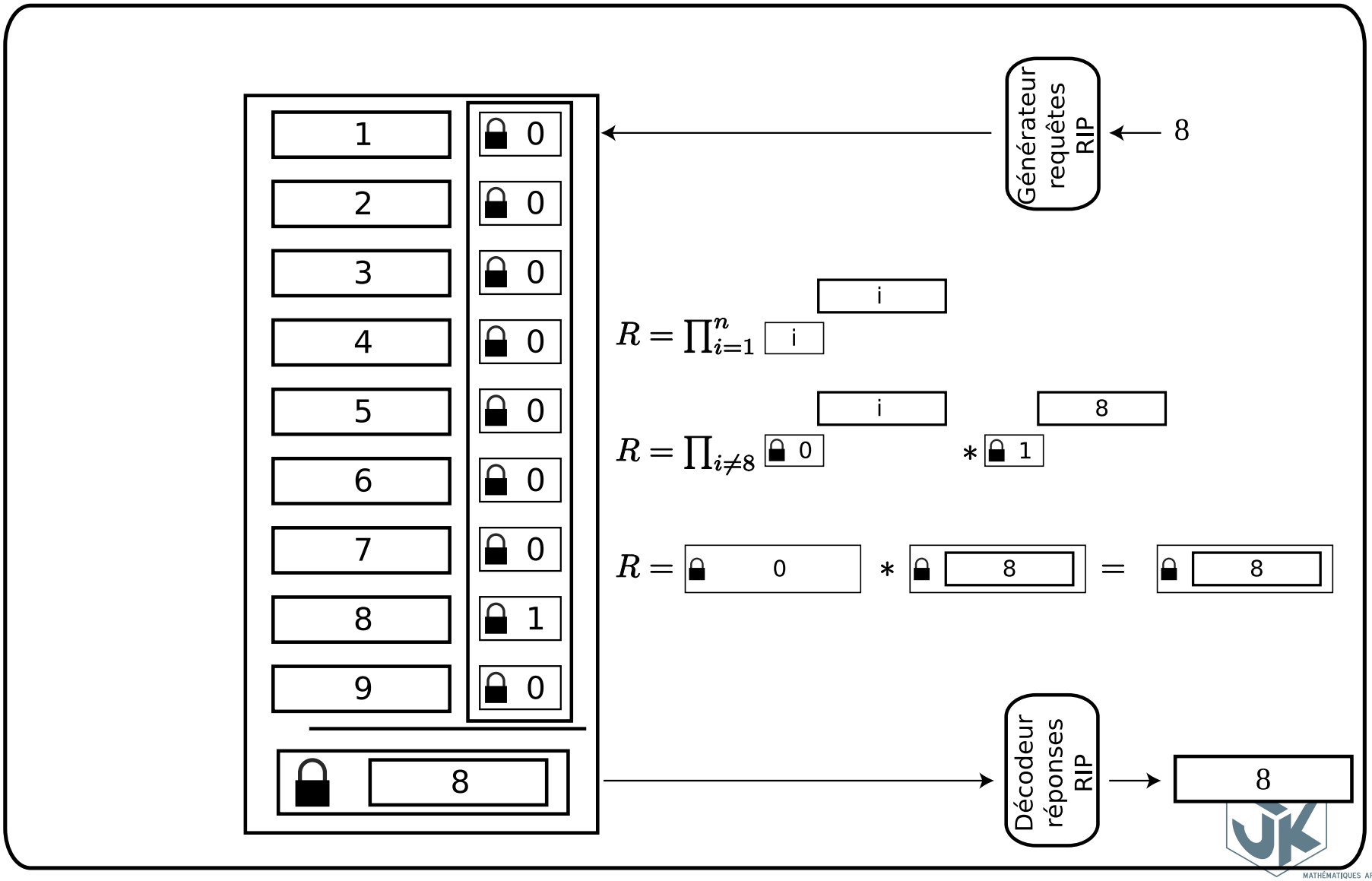
```
sage : (p, q, n) = keygen(20)
sage : (p, q, n)
(192557, 720089, 138658177573)
sage : c1 = encrypt(1, n)
sage : c2 = encrypt(1, n)
sage : c1
16768997473026340839836
sage : c2
14672963152557506378498
sage : decrypt(c1, n, p)
1
sage : decrypt(c1 * c2, n, p)
2
```

Paillier

Usage

```
sage : (p, q, n) = keygen(20)
sage : (p, q, n)
(192557, 720089, 138658177573)
sage : c1 = encrypt(1, n)
sage : c2 = encrypt(1, n)
sage : c1
16768997473026340839836
sage : c2
14672963152557506378498
sage : decrypt(c1, n, p)
1
sage : decrypt(c1 * c2, n, p)
2
sage : decrypt(encrypt(6, n)^7, n, p)
42
```

Protocole PIR par chiffrement homomorphe



Plan

Du PIR au PS

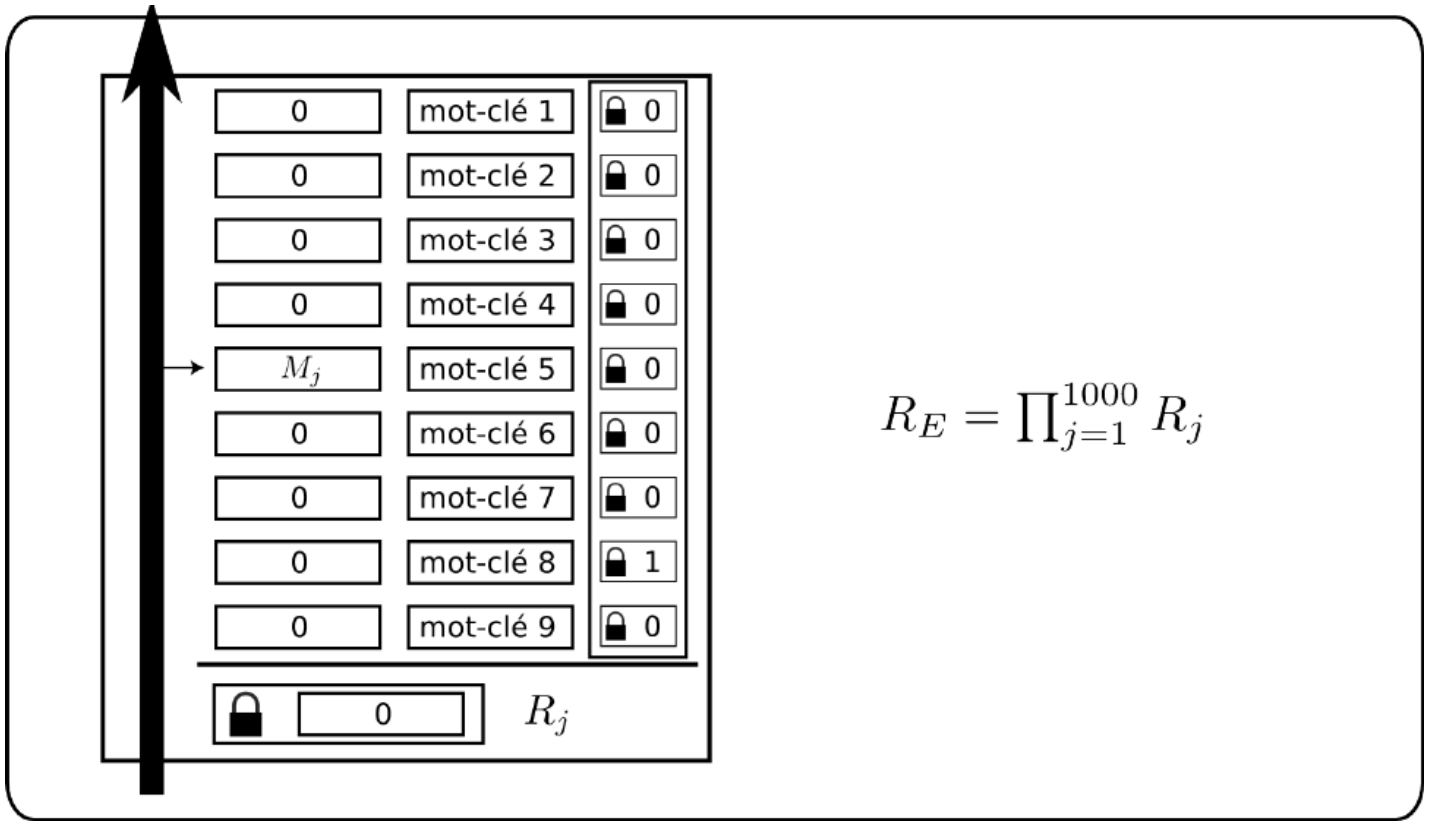
	PIR	PS
Requête	indice de la base	ensemble de mots-clefs
Base de données	Ensemble figé de documents	Flux de documents
Réponse	Immédiate	Périodique

Du PIR au PS

Solution

- Liste de mots-clefs admissibles $W = \{w_1, \ldots, w_{|W|}\}$.
- Requête R code les indices des mots-clefs recherchés.
- Chaque document du flux devient une BD virtuelle de taille $|W|$.
- La requête PS est appliquée comme requête PIR sur cette BD.
- Un document du flux $\rightarrow$ une réponse PIR.
- Comment stocker ces réponses PIR ?

Du PIR au PS


0
mot-clé 1
0
0
mot-clé 2
0
0
mot-clé 3
0
0
mot-clé 4
0
M_j
mot-clé 5
0
0
mot-clé 6
0
0
mot-clé 7
0
0
mot-clé 8
1
0
mot-clé 9
0
0
R_j
$R_E = \prod_{j=1}^{1000} R_j$

Stratégies d'accumulation

Ostrovsky et Skeith

- On récupère au plus m documents ;
- Réponse PS = buffer de $2\gamma m$ réponses PIR ;
- Une réponse PIR est stockée au hasard dans γ positions ;
- Collision $\rightarrow$ emplacement perdu.

Stratégies d'accumulation

Ostrovsky et Skeith

- On récupère au plus m documents ;
- Réponse PS = buffer de $2\gamma m$ réponses PIR ;
- Une réponse PIR est stockée au hasard dans γ positions ;
- Collision $\rightarrow$ emplacement perdu.

Bethencourt et al.

- Plusieurs buffers (données et méta-données) ;
- Réponse PS = buffer de réponses PIR ;
- Réponse PIR copiée dans la moitié des positions en moyenne ;
- Filtre de Bloom pour le nombre de matchs du j-ème document ;

Stratégies d'accumulation

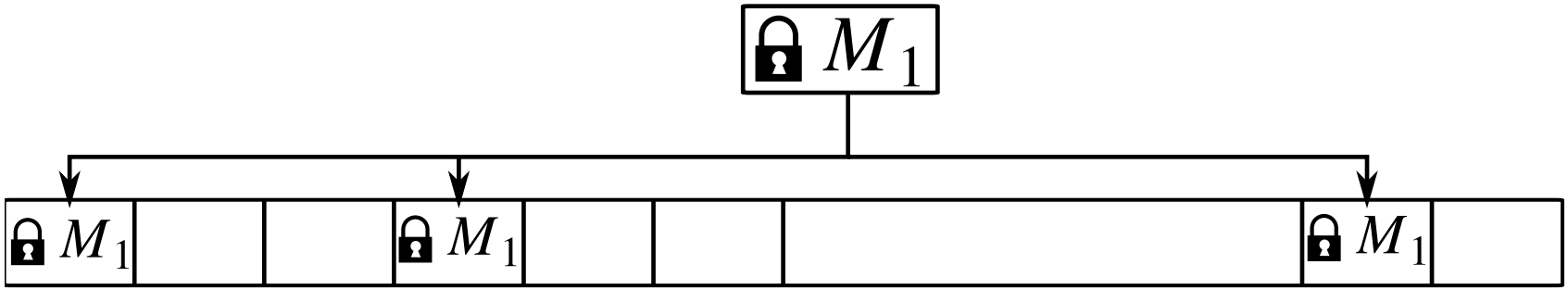
Ostrovsky et Skeith

- On récupère au plus m documents ;
- Réponse PS = buffer de $2\gamma m$ réponses PIR ;
- Une réponse PIR est stockée au hasard dans γ positions ;
- Collision $\rightarrow$ emplacement perdu.

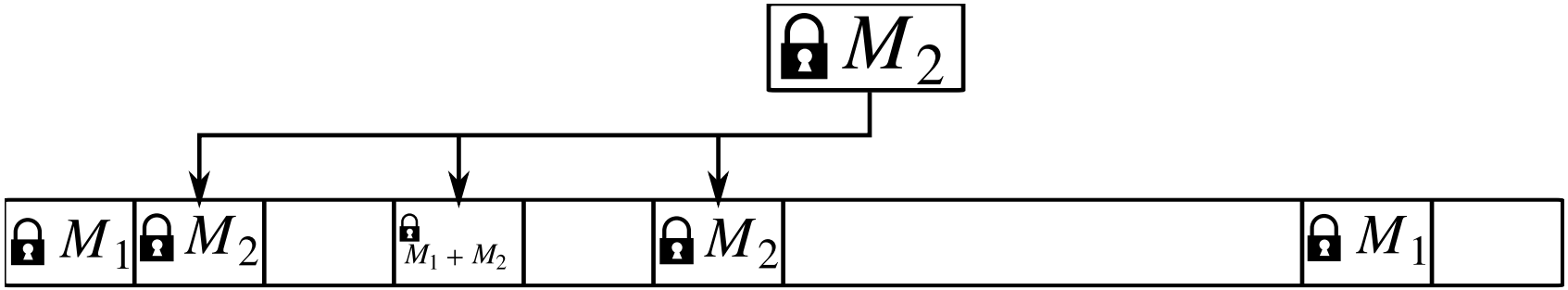
Bethencourt et al.

- Plusieurs buffers (données et méta-données) ;
- Réponse PS = buffer de réponses PIR ;
- Réponse PIR copiée dans la moitié des positions en moyenne ;
- Filtre de Bloom pour le nombre de matchs du j-ème document ;
- Besoin de pouvoir accumuler les résultats.

Accumulation des résultats



Accumulation des résultats



Plan

Principe

- Bethencourt et al. améliore Ostrovsky et Skeith III par l'utilisation de filtres de Bloom ;

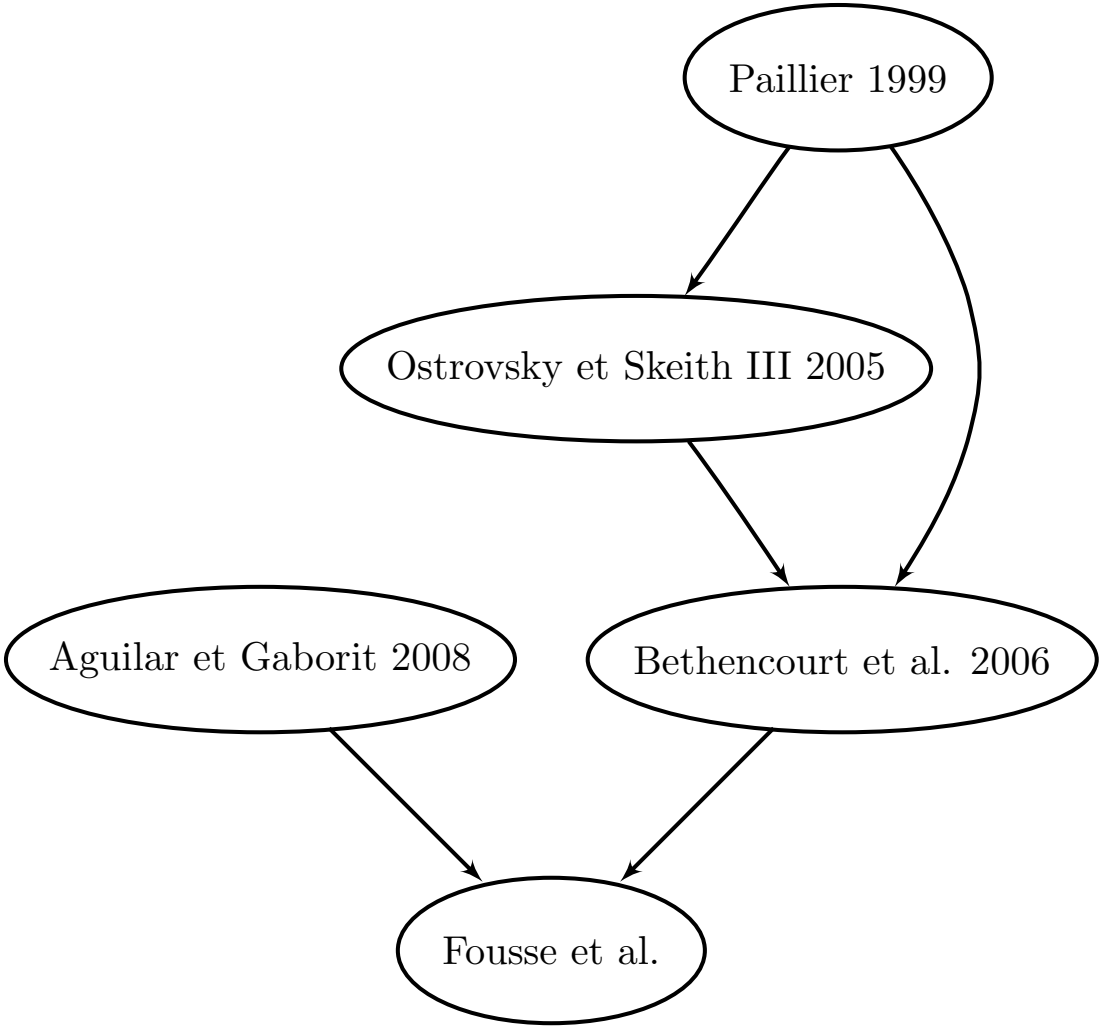
Principe

- Bethencourt et al. améliore Ostrovsky et Skeith III par l'utilisation de filtres de Bloom ;
- Le chiffrement d'Aguilar et Gaborit est plusieurs fois homomorphe ;

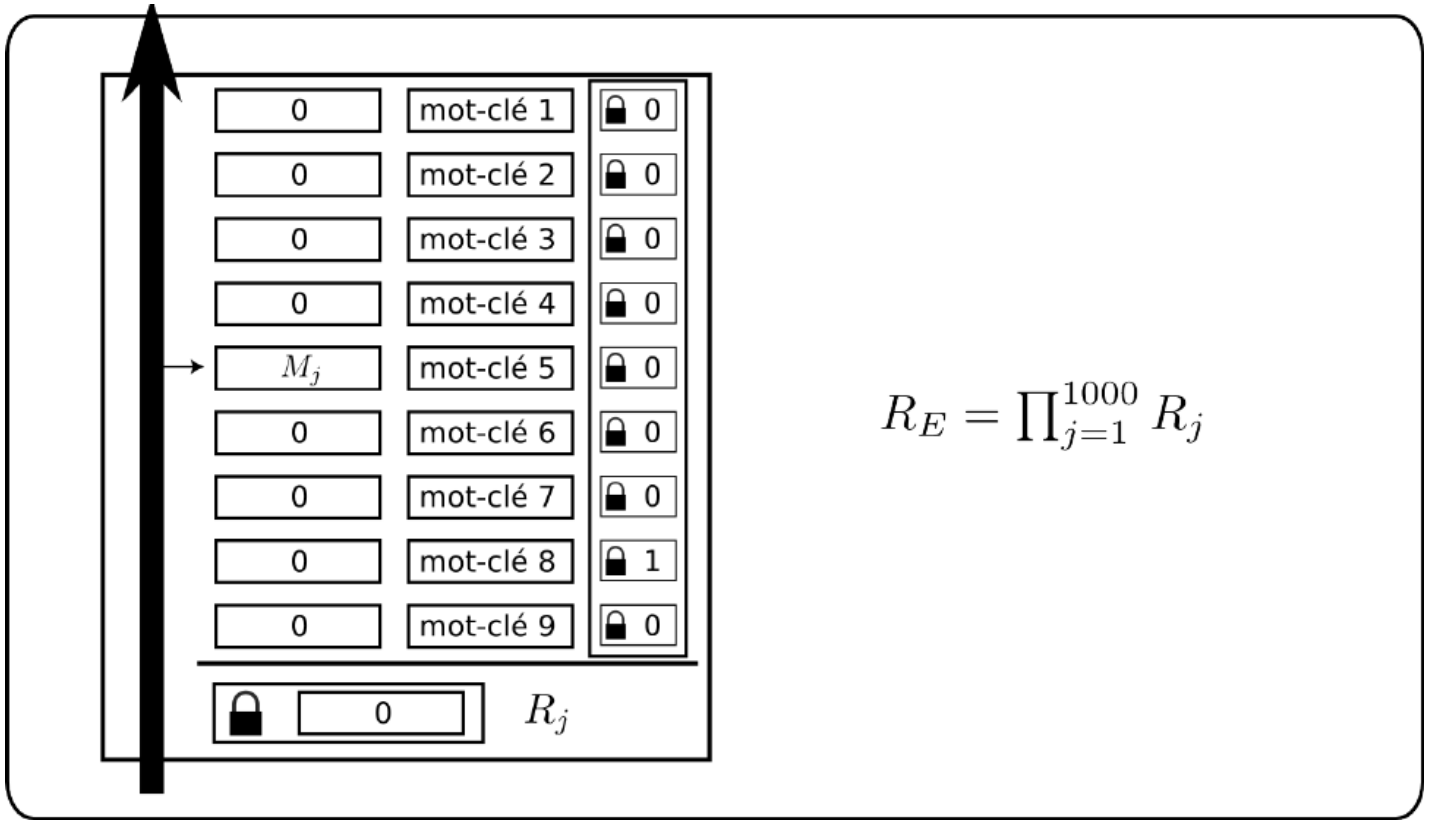
Principe

- Bethencourt et al. améliore Ostrovsky et Skeith III par l'utilisation de filtres de Bloom ;
- Le chiffrement d'Aguilar et Gaborit est plusieurs fois homomorphe ;
- $\Rightarrow$ utiliser le chiffrement par le bruit à la place de Paillier dans le logiciel de Bethencourt et al ;
- et ajouter la récursion.

Vue d'ensemble



Ostrovsky et Skeith III CRYPTO'05



Performances

- Les éléments à zéro n'engendrent pas un coût calculatoire.
- Requêtes grandes mais changent peu souvent.

New Techniques for Private Stream Searching (Bethencourt et al. 2006)

QueryConstruction

- Les éléments Q_i de la requête sont des chiffrés de 0 ou de 1.

New Techniques for Private Stream Searching (Bethencourt et al. 2006)

StreamSearch

Pour le document f_i du flux :

- $n_i = \#$mots-clefs correspondant à la requête.
- $c_i = E(n_i)$.
- 3 filtres de Bloom F, C et I :
  - F contient les données ($F_j = E(\sum_{g(i,j)=1} c_i f_i)$).
  - C contient le nombre de mots-clefs matchant par document du flux ($C_j = E(\sum_{g(i,j)=1} c_i)$).
  - I code les indices des fichiers correspondant à au moins un mot-clef.

New Techniques for Private Stream Searching (Bethencourt et al. 2006)

`FileReconstruction`

- Déchiffrement des buffers ;
- Calcul des indices matchants potentiels (avec I) ;
- Résolution d'un système linéaire pour retrouver n_i.

Problèmes avec Paillier

$$E(m_1) \times E(m_2) \simeq E(m_1 + m_2)$$

Problèmes avec Paillier

$$\begin{aligned} E(m_1) \times E(m_2) &\simeq E(m_1 + m_2) \\ E(E(m_1)) \times E(E(m_2)) &\simeq E(E(m_1) + E(m_2)) \simeq ? \end{aligned}$$

Problèmes avec Paillier

$$
\begin{aligned}
E(m_1) \times E(m_2) &\simeq E(m_1 + m_2) \\
E(E(m_1)) \times E(E(m_2)) &\simeq E(E(m_1) + E(m_2)) \simeq ? \\
E(E(m_1)) + E(E(m_2)) &\simeq ?
\end{aligned}
$$

Problèmes avec Paillier

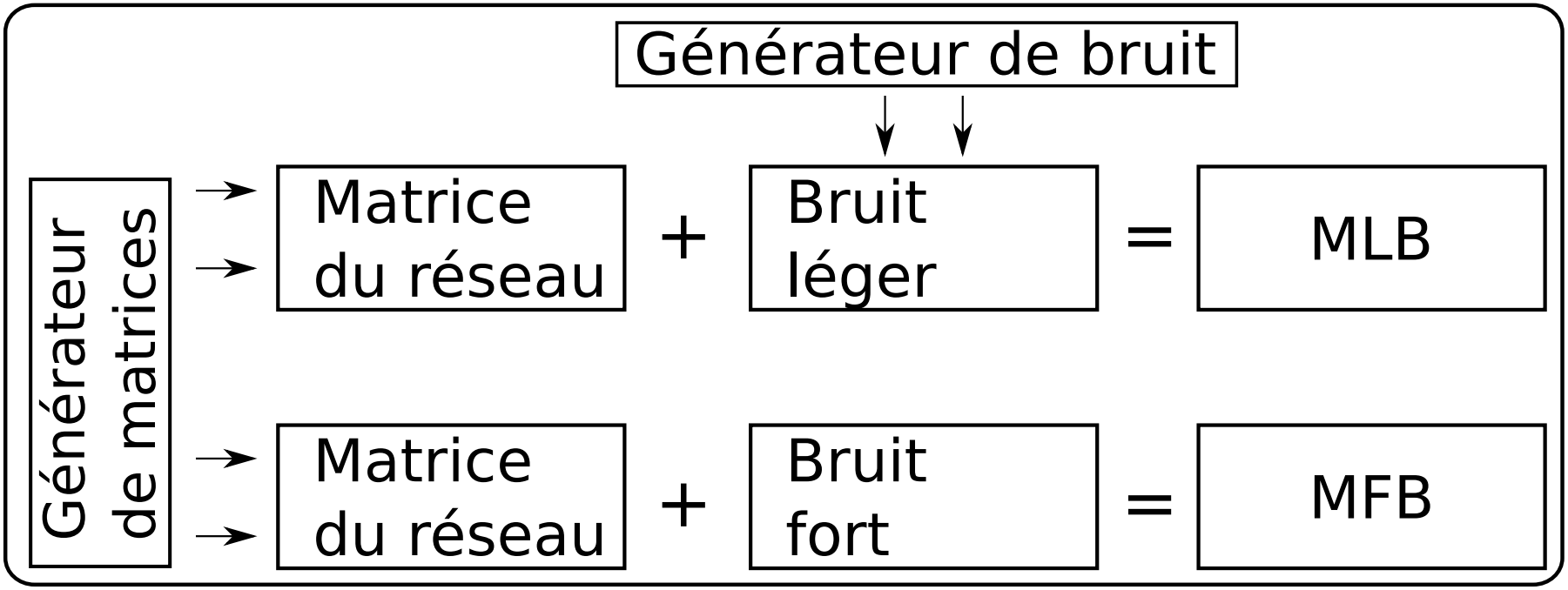
$$
\begin{aligned}
E(m_1) \times E(m_2) &\simeq E(m_1 + m_2) \\
E(E(m_1)) \times E(E(m_2)) &\simeq E(E(m_1) + E(m_2)) \simeq ? \\
E(E(m_1)) + E(E(m_2)) &\simeq ?
\end{aligned}
$$

Un protocole de filtre basé sur Paillier ne permet pas la récursion :

- Requête linéaire en la taille du dictionnaire ;
- W petit : beaucoup de collisions ;
- W grand : plus de calculs et plus grande requête.

Plan

Aguilar et Gaborit WeWORC'07, ISIT'08



Hypothèse de sécurité : *the differential hidden lattice problem*

Il est difficile de distinguer les bases légèrement perturbées des bases fortement perturbées d'un réseau dont on ne connaît pas une base non perturbée.

Aguilar et Gaborit WeWORC'07, ISIT'08

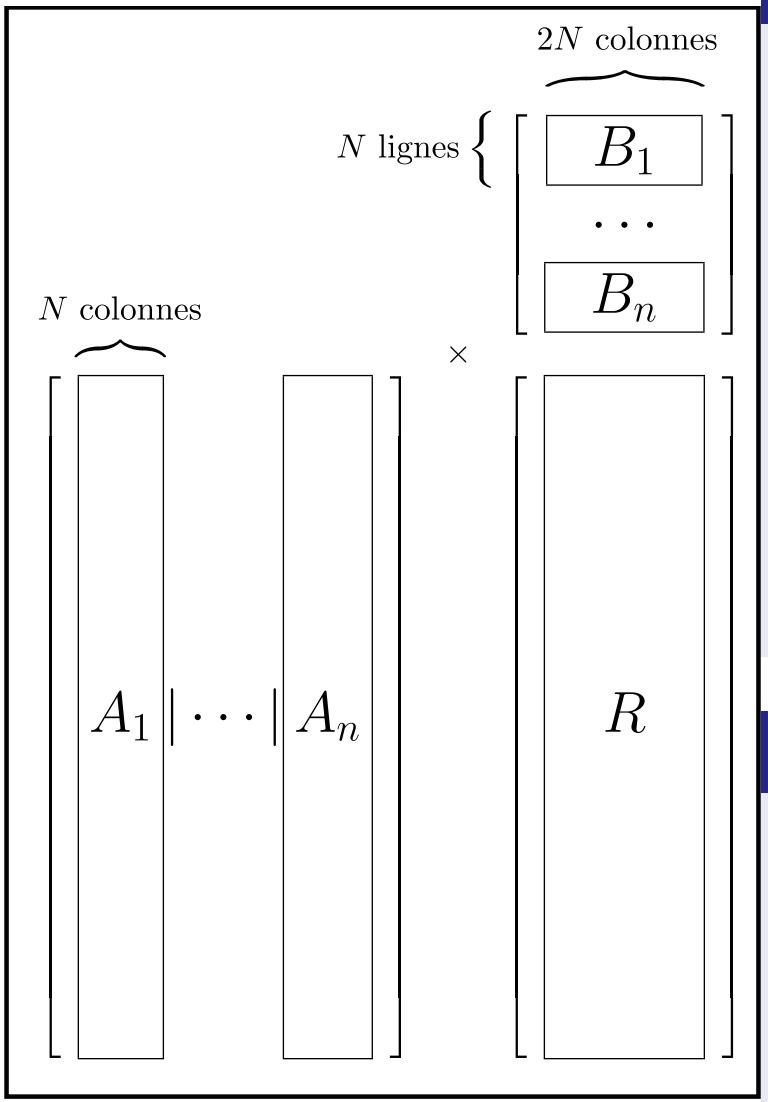
$$\begin{pmatrix} e_{1,1} * \\ e_{1,2} * \\ e_{1,3} * \end{pmatrix} \begin{bmatrix} \text{BASE 1} \end{bmatrix} + \begin{bmatrix} +1 & -1 & +1 & 0 & 0 & 0 \\ +1 & +1 & -1 & 0 & 0 & 0 \\ -1 & +1 & +1 & 0 & 0 & 0 \end{bmatrix} \Delta$$

$$\begin{pmatrix} e_{2,1} * \\ e_{2,2} * \\ e_{2,3} * \end{pmatrix} \begin{bmatrix} \text{BASE 2} \end{bmatrix} + \begin{bmatrix} +64 & +1 & -1 & 0 & 0 & 0 \\ -1 & +64 & -1 & 0 & 0 & 0 \\ +1 & +1 & +64 & 0 & 0 & 0 \end{bmatrix} \Delta$$

Hypothèse de sécurité : *the differential hidden lattice problem*

Il est difficile de distinguer les bases légèrement perturbées des bases fortement perturbées d'un réseau dont on ne connaît pas une base non perturbée.

Aguilar et Gaborit WEWoRC'07, ISIT'08



Génération de la réponse

- La base de données est représentée comme une matrice.
  - Un élément $\rightarrow$ un bloc A_i de N colonnes.
- Les matrices de la requête sont concaténées.
  - Un élément $\rightarrow$ un bloc B_i de N lignes et $2N$ colonnes.
- On réalise une multiplication par blocs.

Coût

- Chaque scalaire de A multiplie une ligne d'une matrice de la requête.
- $N = 50 \Rightarrow$ coût par bit : 100 sommes de 64 bits.

Principe

Détails du chiffrement :

$$M = [M_1|M_2], \quad M_1 \text{ inversible } (N \times 2N)$$

Principe

Détails du chiffrement :

$$\begin{aligned} M &= [M_1|M_2]\,, \quad M_1 \text{ inversible } (N \times 2N) \\ P_i &: \text{ inversible, } (N \times N) \end{aligned}$$

Principe

Détails du chiffrement :

$$\begin{aligned} M &= [M_1|M_2]\,, \quad M_1 \text{ inversible } (N \times 2N) \\ P_i &: \text{ inversible, } (N \times N) \\ \Delta &: \text{ inversible, } (2N \times 2N) \end{aligned}$$

Principe

Détails du chiffrement :

$$
\begin{aligned}
M &= [M_1 | M_2]\,, \quad M_1 \text{ inversible } (N \times 2N) \\
P_i &: \text{ inversible, } (N \times N) \\
\Delta &: \text{ inversible, } (2N \times 2N) \\
B_i &= [P_i M_1 | P_i M_2 + D_i]\,\Delta, \quad D_i \text{ bruit faible } \pm 1
\end{aligned}
$$

Principe

Détails du chiffrement :

$$
\begin{array}{lcl}
M & = & [M_1|M_2]\,, \quad M_1 \text{ inversible } (N \times 2N) \\
P_i & : & \text{inversible, } (N \times N) \\
\Delta & : & \text{inversible, } (2N \times 2N) \\
B_i & = & [P_i M_1 | P_i M_2 + D_i]\,\Delta, \quad D_i \text{ bruit faible } \pm 1 \\
B_j & = & [P_j M_1 | P_j M_2 + D_j]\,\Delta, \quad D_j \text{ bruit fort}
\end{array}
$$

Principe

Détails du chiffrement :

$$
\begin{aligned}
M &= [M_1|M_2]\,, \quad M_1 \text{ inversible } (N \times 2N) \\
P_i &: \text{inversible, } (N \times N) \\
\Delta &: \text{inversible, } (2N \times 2N) \\
B_i &= [P_i M_1 | P_i M_2 + D_i]\,\Delta, \quad D_i \text{ bruit faible } \pm 1 \\
B_j &= [P_j M_1 | P_j M_2 + D_j]\,\Delta, \quad D_j \text{ bruit fort} \\
R &= \sum A_i B_i \\
&= \sum_{soft} [A_i P_i M_1 | A_i P_i M_2 + A_i D_i]\,\Delta + \sum_{hard} [A_j P_j M_1 | A_j P_i M_2 + A_j D_j]\,\Delta.
\end{aligned}
$$

Principe

On récupère

$$R' = \sum_{soft} A_i D_i + \sum_{hard} A_j D_j$$

Principe

On récupère

$$\begin{aligned} R' &= \sum_{soft} A_i D_i + \sum_{hard} A_j D_j \\ &= q \sum_{hard} A_j + noise \end{aligned}$$

Exemple (1/3)

Génération de la requête

Matrices 2x4
GF(521)
Clef privé : M_1, M_2, Δ
Contenu de la requête : B_1, B_2, 521, 3

Bruit léger $\{-1, +1\}$
Bruit fort $\{+64, -64\}$

$$M_1 = \begin{pmatrix} 255 & 323 \\ 133 & 449 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 269 & 469 \\ 159 & 58 \end{pmatrix}$$

$$M = \begin{pmatrix} 255 & 323 & 269 & 469 \\ 133 & 449 & 159 & 58 \end{pmatrix}$$

$$\Delta = \begin{pmatrix} 517 & 411 & 328 & 340 \\ 269 & 172 & 35 & 467 \\ 474 & 7 & 136 & 496 \\ 89 & 233 & 197 & 403 \end{pmatrix}$$

$$P_1 = \begin{pmatrix} 285 & 115 \\ 322 & 3 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 259 & 326 \\ 38 & 232 \end{pmatrix}$$

$$D_1 = \begin{pmatrix} 1 & 0 \\ 1 & 520 \end{pmatrix}$$

$$D_2 = \begin{pmatrix} 64 & 520 \\ 1 & 64 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} 7 & 313 & 77 & 60 \\ 36 & 356 & 65 & 518 \end{pmatrix}$$

$$B_2 = \begin{pmatrix} 365 & 204 & 137 & 71 \\ 262 & 297 & 91 & 490 \end{pmatrix}$$

Exemple (2/3)

Génération de la réponse

- Couper les éléments en morceaux de 3 bits
- Multiplier la matrice données avec la matrice requête

$$\begin{aligned} A_1 &= \begin{pmatrix} 3 & 7 \end{pmatrix} \\ A_2 &= \begin{pmatrix} 5 & 2 \end{pmatrix} \\ R &= \begin{pmatrix} 17 & 356 & 511 & 452 \end{pmatrix} \end{aligned}$$

Exemple (3/3)

Décodage de la réponse

1 - Permuter
2 - Récupérer le vecteur du réseau
3 - Récupérer le bruit
4 - Déamplifier le bruit
5 - Filtrer le bruit
6 - Reconstruire

Exemple

$$R\Delta^{-1} = \begin{pmatrix} 360 & 243 & 248 & 490 \end{pmatrix}$$

Exemple (3/3)

Décodage de la réponse

1 - Permuter
2 - Récupérer le vecteur du réseau
3 - Récupérer le bruit
4 - Déamplifier le bruit
5 - Filtrer le bruit
6 - Reconstruire

Exemple

$$
\begin{aligned}
R\Delta^{-1} &= \begin{pmatrix} 360 & 243 & 248 & 490 \end{pmatrix} \\
R'_1 &= \begin{pmatrix} 360 & 243 \end{pmatrix} \\
R'_2 &= \begin{pmatrix} 248 & 490 \end{pmatrix} \\
R'_2 - R'_1 M_1^{-1} M_2 &= \begin{pmatrix} 332 & 116 \end{pmatrix} \\
332 &= 5 \cdot 64 + 12 \\
116 &= 2 \cdot 64 - 12
\end{aligned}
$$

Intérêt du chiffrement utilisé

Le chiffrement transforme une addition en une addition :

$$\begin{aligned}
\mathrm{PIR}(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_n) &= \mathrm{PIR}(\mathbf{A}, \mathbf{B}) \\
\mathrm{PIR}(\mathbf{A}_1, \mathbf{B}) + \mathrm{PIR}(\mathbf{A}_2, \mathbf{B}) &\simeq \mathrm{PIR}(\mathbf{A}_1 + \mathbf{A}_2, \mathbf{B}) \\
\mathrm{PIR}(\mathbf{A}'_1, \mathbf{B}) + \mathrm{PIR}(\mathbf{A}'_2, \mathbf{B}) &\simeq \mathrm{PIR}(\mathbf{A}'_1 + \mathbf{A}'_2, \mathbf{B})
\end{aligned}$$

Intérêt du chiffrement utilisé

Le chiffrement transforme une addition en une addition :

$$\begin{aligned}
\mathrm{PIR}(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_n) &= \mathrm{PIR}(\mathbf{A}, \mathbf{B}) \\
\mathrm{PIR}(\mathbf{A}_1, \mathbf{B}) + \mathrm{PIR}(\mathbf{A}_2, \mathbf{B}) &\simeq \mathrm{PIR}(\mathbf{A}_1 + \mathbf{A}_2, \mathbf{B}) \\
\mathrm{PIR}(\mathbf{A}'_1, \mathbf{B}) + \mathrm{PIR}(\mathbf{A}'_2, \mathbf{B}) &\simeq \mathrm{PIR}(\mathbf{A}'_1 + \mathbf{A}'_2, \mathbf{B})
\end{aligned}$$

Récursion en dimension 2 :

$$\begin{aligned}
\mathrm{PIR}(\mathrm{PIR}(\mathbf{A}_1, \mathbf{B}_1), \mathrm{PIR}(\mathbf{A}_2, \mathbf{B}_1), \ldots, \mathrm{PIR}(\mathbf{A}_n, \mathbf{B}_1), \mathbf{B}_2) &+ \\
\mathrm{PIR}(\mathrm{PIR}(\mathbf{A}'_1, \mathbf{B}_1), \mathrm{PIR}(\mathbf{A}'_2, \mathbf{B}_1), \ldots, \mathrm{PIR}(\mathbf{A}'_n, \mathbf{B}_1), \mathbf{B}_2) &\simeq \\
\mathrm{PIR}(\mathrm{PIR}(\mathbf{A}_1 + \mathbf{A}'_1, \mathbf{B}_1), \mathrm{PIR}(\mathbf{A}_2 + \mathbf{A}'_2, \mathbf{B}_1), \ldots, \mathrm{PIR}(\mathbf{A}_n + \mathbf{A}'_n, \mathbf{B}_1), \mathbf{B}_2)
\end{aligned}$$

Paramètres du protocole

Paramètres du chiffrement

- Taille l_0 des entiers de $\mathbf{A}$.
- Dimensions L et N des matrices ;
- Taille du premier p ;
- Taille des textes clairs ;
- Facteur d'expansion.

Paramètres globaux

- Tailles des buffers F, I et C ;
- Paramètre de correction ϵ ;
- Nombre de mot-clés de la requête ;
- Nombre moyen de mot-clés présents dans un document ;
- Nombre moyen de mot-clés de la requête présents dans un document ;
- Nombre de documents du flux traités.

Contraintes

Hypothèses

- Un emplacement du buffer accumulera en moyenne la moitié des t messages du flux traités ;
- Un message donné correspond à la requête avec une probabilité bornée ;
- Les matrices A contiennent des entiers sur l_0 bits ;
- Le nombre de mot-clés présents dans chaque document est borné.

Contraintes

Bornes

(Rappel) On récupère

$$R' = \sum_{soft} A_i D_i + \sum_{hard} A_j D_j$$

Contraintes

Bornes

(Rappel) On récupère

$$\begin{aligned} R' &= \sum_{soft} A_i D_i + \sum_{hard} A_j D_j \\ &= q \sum_{hard} A_j + noise \end{aligned}$$

Contraintes

Bornes

(Rappel) On récupère

$$\begin{aligned} R' &= \sum_{soft} A_i D_i + \sum_{hard} A_j D_j \\ &= q \sum_{hard} A_j + noise \end{aligned}$$

Borne sur le bruit :

$$noise \leq |a_{ij}| \left(m(N-1) + \hat{m}N \right)$$

Contraintes

Bornes

(Rappel) On récupère

$$\begin{aligned} R' &= \sum_{soft} A_i D_i + \sum_{hard} A_j D_j \\ &= q \sum_{hard} A_j + noise \end{aligned}$$

Borne sur le bruit :

$$noise \leq |a_{ij}| \left(m(N-1) + \hat{m}N \right)$$

Borne sur le signal :

$$data \leq q|a_{ij}|m$$

où m désigne le nombre total, pour l'emplacement considéré, des mot-clés trouvés dans les documents traités ($m + \hat{m}$ le nombre total de mot-clés présents dans les documents).

Contraintes

Bornes

- Le bruit ne doit pas déborder dans le signal :

$$q > 2^{l_0} N(m + \hat{m}) \tag{1}$$

- Le signal ne doit pas déborder mod p :

$$p > q 2^{l_0} m \tag{2}$$

Contraintes

Bornes

- Le bruit ne doit pas déborder dans le signal :

$$q > 2^{l_0} N(m + \hat{m}) \tag{1}$$

- Le signal ne doit pas déborder mod p :

$$p > q 2^{l_0} m \tag{2}$$

- On peut supposer que le facteur d'expansion des entiers est 3.

Analyse de la récursion

Avec une récursion d'ordre d ($|W| = n^d$) :

Communication Client vers Serveur

- Coût n^d sans récursion.
- Coût $n + 3n + 9n + \ldots + 3^{d-1}n$ avec récursion.
- $\Rightarrow$ facteur $\frac{3}{2}\left(\frac{3}{n}\right)^{d-1}$ (meilleur avec récursion !)

Analyse de la récursion

Avec une récursion d'ordre d ($|W| = n^d$) :

Communication Client vers Serveur

- Coût n^d sans récursion.
- Coût $n + 3n + 9n + \ldots + 3^{d-1}n$ avec récursion.
- $\Rightarrow$ facteur $\frac{3}{2}\left(\frac{3}{n}\right)^{d-1}$ (meilleur avec récursion !)

Coût de calcul du serveur

- Même analyse que pour la taille de la requête ?

Analyse de la récursion

Avec une récursion d'ordre d ($|W| = n^d$) :

Communication Client vers Serveur

- Coût n^d sans récursion.
- Coût $n + 3n + 9n + \ldots + 3^{d-1}n$ avec récursion.
- $\Rightarrow$ facteur $\frac{3}{2}\left(\frac{3}{n}\right)^{d-1}$ (meilleur avec récursion !)

Coût de calcul du serveur

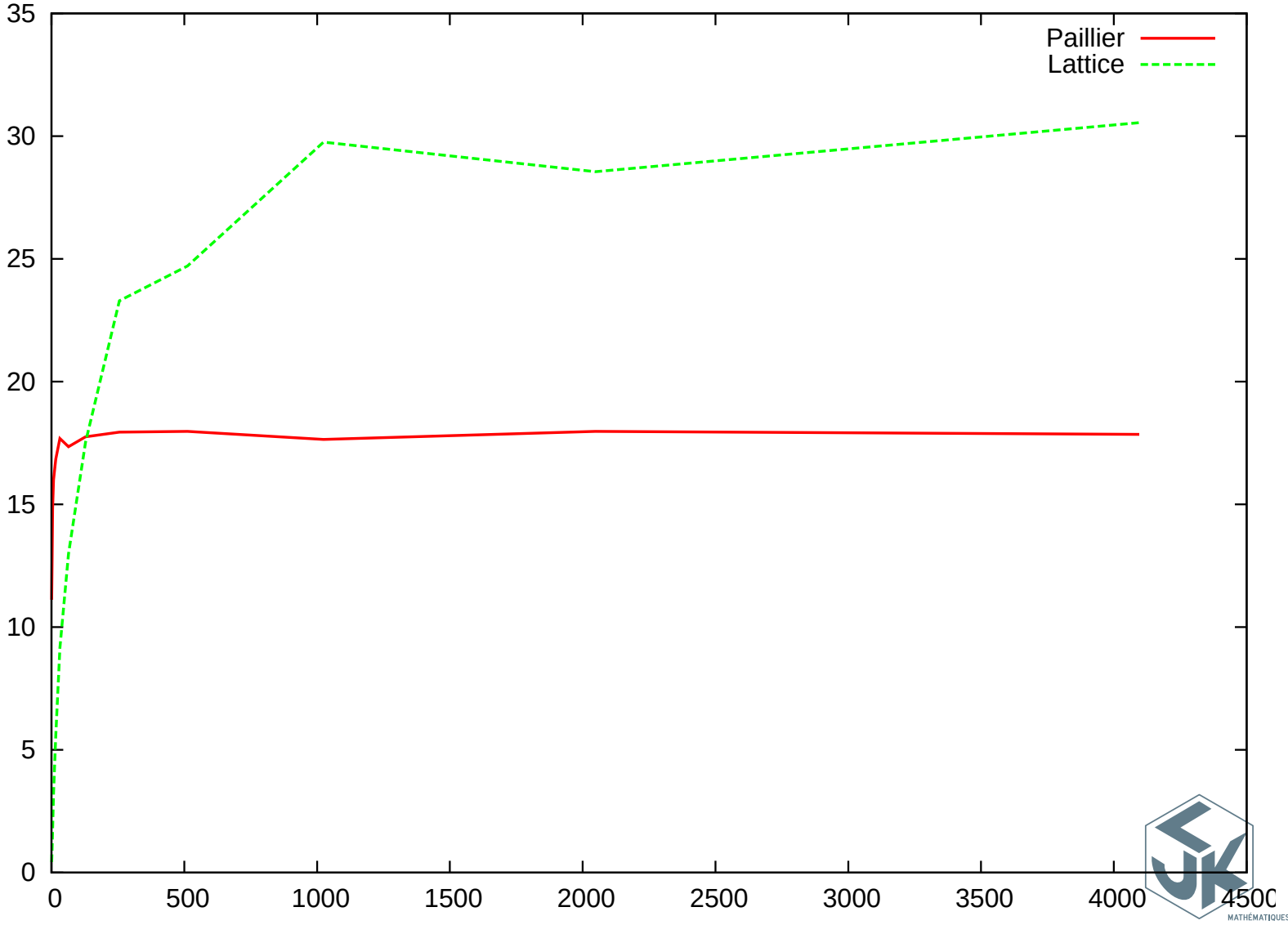
- Même analyse que pour la taille de la requête ?
- Mais un mot-clé non présent n'entraine pas de coût de calcul !

Taille de la réponse

- Expansion 6^d, trop cher ?
- On peut diminuer le nombre de buffers.

Plan

Vitesse en ko/s, taille des fichiers en ko



Temps

$$
\begin{aligned}
|I| &= 315 \\
|C| &= 135 \\
|F| &= 512
\end{aligned}
$$

	Réseaux	Paillier
`QueryConstruction`	20s	3s
`FileReconstruction`	44s	36s

Conclusion

- Avec le chiffrement basé sur les réseaux, la récursion est possible pour le *private searching* ;
- Déjà plus performant malgré un code non optimisé

Conclusion

- Avec le chiffrement basé sur les réseaux, la récursion est possible pour le *private searching* ;
- Déjà plus performant malgré un code non optimisé (et même grossièrement non optimisé).

Perspectives

- Implémenter la récursion ;
- Utiliser LinBox au lieu de NTL ;
- Trouver automatiquement les bons paramètres et mieux comparer ;
- Trouver l'Application-qui-tue (*killer app*) qui montre qu'on est bien meilleur que les autres ;
- Application au serveur de mail anonyme (cf. *Pynchon Gate*).