Algorithmes arithmétiques pour la cryptologie — 12 octobre 2015, Paris

Algèbre linéaire et NFS

Razvan Barbulescu
CNRS et IMJ-PRG

Outline of the talk

- Algèbre linéaire

- Le crible algébrique (NFS)

Algèbre linéaire dans le contexte de la cryptologie

- Que résoudre? Les algorithmes modernes de logs discrets et factorisation, de complexité $L(1/3)$ ou moins demandent de résoudre $Mw = 0$, pour une matrice creuse M (peu d'entrées non nulles par ligne).
- $K = ?$ Pour la factorisation $K = \mathbb{F}_2$, alors que pour DLP $K = \mathbb{F}_\ell$ avec $|\ell|$ entre 200 et 600 bits
- logiciel
  - très peu d'implantations disponibles pour l'algèbre linéaire sur des corps finis grands, e.g. LINBOX se restreint aux cotps de 32 bits
  - CADO: algèbre linéaire sur $\mathbb{F}_2$; Hamza Jeljeli: logiciel pour $\mathbb{F}_p$
- difficulté supplémentaire Pour certains algorithmes de logs discrets la matrice a quelques (en pratique entre 1 et 6) colonnes lourdes qui doivent être traitées séparément et requièrent beaucoup de RAM.

Algèbre linéaire pleine ou creuse?[1]

Algorithmes pour matrices pleines

- espace $O(N^2)$, représentation en mémoire sous forme de tableau bidimensionnel
- temps $O(N^\omega)$, même que pour la multiplication, où
  - Gauss (algorithme naïf) $\omega = 3$;
  - Strassen $\omega = 2.81$;
  - Coppersmith-Winograd $\omega = 2.38$;
  - problème ouvert: $\omega = 2$?

Algorithmes pour matrices creuses

- espace $N\lambda$,
  - matrice stockée comme liste de $N\lambda$ paires $(i, j_{i,0}, v_{i,0}), \ldots, (i, j_{i,k_i}, v_{i,k_i})$ contenant les positions des entrées non nulles et leur valeur;
  - matrice en lecture seule, on implante le produit matrice fois vecteur et utilise cela comme boîte noire (building block)
- temps $O(N^2\lambda)$, correspondant à $O(N)$ appels à la boîte noire.

[1] remerciements à E. Thomé

Algorithme de Wiedemann: idée de base

Problème

Étant donné un corps K et une matrice $M \in \mathrm{Mat}_{N\times N}(K)$, avec λ entrées non nulles par ligne, telle que $\det M = 0$. Dans $O(N^2\lambda)$ operations surr K, trouvez une solution non nulle de

$$Mw = 0.$$

Esquisse de la solution

1. Trouver un polynôme h dans $K[x]$ tel que $h(M) = 0$ (par exemple le polynôme charactéristique).
2. Remarquer que $\det M = 0$ implique que $h(x) = xh^{-}(x)$ pour un polynôme h^{-}.
3. Prendre un vecteur aléatoire u et évaluer $w = h^{-}(M)u$.
4. Remarquer que $Mw = Mh^{-}(M)u = h(M)u = 0u = 0$.
5. Remarquer que, si h a été choisi de degré minimal alors $h^{-}(M) \neq 0$. Aussi, comme u est aléatoire on montrera plus loin que, avec une grande probabilité, $w \neq 0$.

Récupération du générateur linéaire

Problème

Étant donnée une séquence (suite finie) générée par une récurrence linéaire, trouvez le générateur linéaire:

- pour $1, 10, 100, 1000, \ldots$, on trouve $1 - 10x$ (c-à-d $u_n = 10u_{n-1}$);
- pour $1, 1, 2, 3, 5, 8, 13, 21, \ldots$, on trouve $1 - x - x^2$ (c-à-d $u_n = u_{n-1} + u_{n+2}$).

Formalisme

Soit $a_0, a_1, \ldots$ une suite donnée par la récurrence

$$\forall k,\ a_k = -\lambda_1 a_{k-1} - \lambda_2 a_{k-2} - \cdots - \lambda_n a_{k-n}.$$

Alors, le générateur linéaire $\Lambda = 1 + \lambda_1 x + \cdots + \lambda_n x^n$ vérifie

$$(a_0 + a_1 x + \cdots a_{2n-1} x^{2n-1})\Lambda(x) \equiv (b_0 + \cdots + b_{n-1} x^{n-1}) \mod x^{2n},$$

pour des scalaires $b_0, \ldots, b_{n-1}$.

Solutions

- $(s_0, r_0) = (0, x^{2n})$ et $(s_1, r_1) = (1, \sum_{i=0}^{2n-1} a_i x^i)$
- si (s, r) et (s', r') sont solutions, alors toute combinaison $(\alpha(x)s + \beta(x)s', \alpha(x)r + \beta(x)r')$ est aussi solution, $\alpha, \beta \in K[x]$.

Algorithme d'Euclide étendu (EEA)

Algorithme

Input deux polynômes $f, g \in K[x]$ avec $\deg f, \deg g \leq 2n$ et $\deg(\gcd(f,g)) < n$.

Output une séquence de triplets $(r_i, t_i, s_i) \in K[x]^3$ tels que $r_i = t_i f + s_i g$, et
$\deg r_0 \geq \deg r_1 \geq \ldots \geq \deg r_k$, où $r_k = \gcd(f,g)$

$$\begin{pmatrix} r_1 & t_1 & s_1 \\ r_0 & t_0 & s_0 \end{pmatrix} \leftarrow \begin{pmatrix} g & 0 & 1 \\ f & 1 & 0 \end{pmatrix}$$

$i \leftarrow 1$

while $\deg r_i \geq n$ **do**

$\quad q_i \leftarrow r_{i-1} \operatorname{div} r_i$

$$\quad \begin{pmatrix} r_{i+1} & t_{i+1} & s_{i+1} \\ r_i & t_i & s_i \end{pmatrix} \leftarrow \begin{pmatrix} -q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} r_i & t_i & s_i \\ r_{i-1} & t_{i-1} & s_{i-1} \end{pmatrix}$$

$\quad i \leftarrow i+1$

end while

return $\{(r_0, t_0, s_0), \ldots, (r_i, t_i, s_i)\}$

Propriétés

- La séquence $(\deg r_i)$ décroît strictement, alors que $(\deg t_i)$ et $(\deg s_i)$ croissent.
- Pour tout i, $\deg s_i = \deg q_0 + \cdots + \deg q_{i-1} = \deg r_0 - \deg r_{i-1}$.

Complexité quand $\deg f, \deg g \leq n$

EEA a un coût de $O(n^2)$, alors que des variantes rapides coûtent $O(M(n)(\log n))$.

Calcul du générateur linéaire avec EEA

Theorem

EEA appliqué à $f = x^{2n}$ et $g = \sum_{k=0}^{2n-1} a_k x^k$ renvoie (r, t, s) tels que s est un générateur linéaire.

Proof.

- $r_0 = x^{2n}$ et $r_1 = \sum_{k=0}^{2n-1} a_k x^k$ ont un pgcd de degré inférieur à n car la séquence (a_k) n'est pas nulle. Alors $\deg r_i < n$ à partir d'un certain rang i.
- Soit i_0 la dernière valeur de i dans l'algorithme, c-à-d.

$$\deg r_{i_0} < n \leq \deg r_{i_0-1}.$$

On a $\deg s_{i_0} = \deg r_0 - \deg r_{i_0-1} = 2n - \deg r_{i-1} \leq n$.
- La paire (s, r) satisfait les conditions qui définissent un générateur linéaire.

$\square$

Berlekamp-Massey (calcul alternatif du générateur linéaire)

- vient de la théorie des codes correcteurs d'erreurs (BCH);
- complexité $O(n^2)$, même que celle obtenue avec EEA;
- variantes rapides de complexité $O(M(n)(\log n))$ (comme pour EEA);
- contrairement à EEA, se généralise au calcul de générateurs linéaires matriciels (Thomé 2003, etc).

Wiedemann

Algorithme

Input An $N \times N$ singular matrix M over a field K

Output a non-trivial solution of $Mu = 0$

```
x←Random(K^{N×1}), y←Random(K^{1×N}),
[Krylov] Compute a_i = yM^i x for i in [0, 2N − 1]
[Linear generator] Compute the linear generator Λ = Σ_{i=0}^{deg Λ} c_{deg Λ−i} x^i of (a_n)_{n∈ℕ}
h(x) ← Σ_{i=0}^{deg Λ} c_i x^i
h^−(x) ← x^{−val_x h} h(x)
v ←Random(K^{N×1});                                  ▷ can be v ← x
[Make solution] u ← h^−(M)v
repeat
    u ← Mu
until  Mu = 0 and u ≠ 0                              ▷ ≤ N + 1 − deg Λ times
```

Complexité

- un produit matrix fois vecteur coûte $N\lambda$ multiplications dans K
- Krylov: $2N^2\lambda + N^2$ opérations dans K
- Linear generator: N^2 (complexité de EEA). Mais il existe des variantes rapides de complexité $O(N(\log N)^2)$.
- make solution: $N^2\lambda$ (algorithme de Horner)
- total: $(3 + o(1))N^2\lambda$.

Correction

Notation

- μ=le polynôme unitaire de degré minimal tel que $\mu(M) = 0$;
- μ_x=le polynôme unitaire de degré minimal tel que $\mu_x(M)x = 0$;
- $\mu_{x,y}$=le polynôme unitaire de degré minimal tel que $y^t\mu_{x,y}(M)x = 0$.

Theorem

Si x et y sont choisis aléatoirement dans le corps fini K, alors, avec probabilité supérieure ou égale à $1 - \min(1, O(\frac{N}{\#K}))$,

$$\mu = \mu_x = \mu_{x,y}.$$

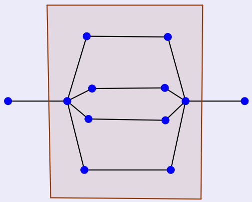
Proof.

- Clairement $\mu_{x,y}$ divise μ_x, qui divise μ.
- Étant donné un polynôme μ', $\mu'(M)x = 0$ si et seulement si x est dans un sous-espace vectoriel, donc cela se réalise avec probabilité $O(1/\#K)$. Comme μ a au plus N cofacteurs de facteurs irréductibles, la probabilité d'échec vaut $\min(1, O(N/\#K))$
- Étant donné un vecteur x et un polynôme μ' tel que $\mu'(M)x \neq 0$, on a $y^t\mu'(M)x \neq 0$ sauf si on a choisi par erreur y dans le hyperplan des vecteurs perpendiculairs sur $\mu'(M)x$. Cela se passe avec probabilité $O(1/\#K)$.

□

Deux niveaux de parallélisme

À l'intérieure de la boîte noire

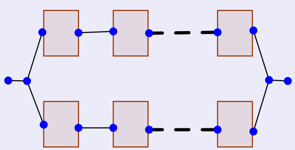


⋆ le produit des matrices est fait par block

⋆ chaque unité de calcul (node de coeurs CPU ou paire de GPUs)
stoque un block de la matrice,
donc on parallélise la mémoire

⋆ synchronisation après chaque appel à la boîte noire: vitesse de c
élevée (InfiniBand ou deux GPUs sur le même PC)

À l'extérieur de la boîte noire



⋆ la variante de Widemann par blocks
a des séquences de calcul indépendantes

⋆ pas de communications (à ce niveau de parallélisme)

⋆ chaque séquence stoque toute la matrice
en mémiore: pas de parallélisme sur la mémoire

⋆ compatible avec le parallélisme à l'intérieur de la boîte noire

Autres algorithmes

Lanczos

- emprenté à la communauté d'EDP;
- effectue des orthogonalisations de Gram-Schimdt par rapport à $M^t M$;
- coûte $(2 + o(1))N^2\lambda$.

Block Wiedemann

- Krylov et Make solution permetent le parallélisme à l'extérieure de la boîte noire;
- polynôme minimal ayant comme coefficients des matrices $m \times n$, e.g. 2×2;
- générateur linéaire relenti par un petit facteur dépendant de m et n;
- quand $K = \mathbb{F}_2$ et $n = 32$, 32 appels de la boîte noire coûte autant qu'un appel quand $K = \mathbb{F}_\ell$ où ℓ a 32 bits.

Block Lanczos

Si les calculs sont faits sur n coers, on communique après chaque itération, donc N/n points de communication. C'est mieux que Wiedemann, mais moins bien que Block Wiedemann.

En practique

- log discret: on peut utiliser tous les algorithmes;
- factorisation: Lanczos et Wiedemann échouent avec probabilité non nulle.

Systèmes non-homogènes et noyaux à gauche

Noyau à gauche

Les matrices creuses sont stoquées comme listes $\{(0, i_{0,0}), \ldots, (1, i_{1,0}), \ldots\}$. On peut trier la liste sur la deuxième coordonnée, en temps quasi-linéaire $O(N\lambda)$ et on obtient la représentation creuse de M^t. Alors on applique Wiedemann à M^t.

Résolution $Mw = b$: première méthode

Appliquez l'algorithme pour systèmes homogènes au système

$$\left(\begin{array}{ccc|c} & M & & b^t \\ \hline 0 & \cdots & 0 & 0 \end{array}\right) (x|x_{N+1})^t = 0.$$

Multiplier la solution par un scalaire pour que la dernière coordonnée soit -1.

Systèmes non-homogènes: 2e méthode

Résolution $Mw = b$, 2e méthode

On modifie Wiedemann comme suit:

1. Calculer un polynôme h tel que $h(M)b = 0$.
2. Écrire $h = xh^{-}(x) + h_0$ et calculer $w = \frac{-1}{h_0}h^{-}(M)b$.

Block Wiedemann se modifie aussi.

Effacer le coût des colonnes lourdes (Thomé/Joux-Pierrot)

- Dans le record de calcul de logs discrets dans $\mathbb{F}_p$ effectué en 2014 par Caramel, le coût supplémentaire dû à 2 colonnes lourdes a pris la moitié du temps de l'algèbre linéaire (on a doublé le temps par rapport à un système homogène).
- E. Thomé a implanté le nouvel algorithme dans CADO, de manière que le coût des colonnes lourdes est nul.

Outline of the talk

- Algèbre linéaire
- Le crible algébrique (NFS)

L'intérêt des diagrammes commutatifs

Exemple pour log discret (avec les entiers de Gauss)

- But: Logs discrets dans $\mathbb{F}_p$ pour $p \equiv 1 \mod 4$.
- Calculer une racine de $r^2 + 1 = 0$ dans $\mathbb{F}_p$ et mettre $f = x - r$ et $g = x^2 + 1$.
- Calculer les paires d'entiers (a, b) telles que $F(a, b) = a - rb$ et $G(a, b) = a^2 + b^2$ sont friables.
- Factoriser $a - br = \prod q_i^{e_i}$ et $(a - \sqrt{-1}b) = \prod(\pi_j + \sigma_j\sqrt{-1})^{\epsilon_j}$ ($\mathbb{Z}[\sqrt{-1}]$ est factoriel).
  Puisque $G(a, b) = a^2 + b^2 = \prod_j(\pi_j^2 + \sigma_j^2)$, tous q_i, π_j et σ_j sont petits.
- On obtient, dans $\mathbb{F}_p^*$:

$$\prod q_i^{e_i} \equiv a - br \equiv \prod(\pi_j + \sigma_j r)^{\epsilon_j} \mod p.$$

- Prendre le logarithme discret des deux côtés de l'équation.
- Continuer comme dans Index Calculus.

Que change?

Si f et g ont des petits coefficients, on remplace la probabilité de friabilité d'un grand entier par celle que deux entiers très petits soient friables au même temps.

Sélection polynomiale

But

Trouver deux polynômes f et g ayant une racine commune modulo un entier donné (N composé pour la factorisation et p premier pour le log discret).

Entiers de Gauss

Dans l'exemple précédent on utilise la reconstruction rationnelle(EEA) pour écrire $r \equiv u/v \mod p$ avec $u, v \approx \sqrt{p}$. Remplacer f par $u - xv$, donc $\|f\|_\infty \approx \sqrt{p}$. Alors

1. $F(a, b) \approx \sqrt{p}$,
2. $G(a, b)$ très petit.

C'est comme si on testait la friabilité de nombres de taille $\sqrt{p}$ au lieu de p.

Base-m

On pose $m = \lfloor N^{1/d} \rfloor$ et on écrit $N = m^d + N_{d-1}m^{d-1} + \cdots N_1 m + N_0$ en base M et pose

- $f = x^d + \cdots + N_1 x + N_0$;
- $g = x - m$.

On a $|F(a, b)| \approx E^d m$ et $|G(a, b)| \approx Em$ où E est le majorant de $|a|$ et $|b|$.

Changement de complexité: $L(1/3)$

Taille des normes?

On a $|F(a,b)| \approx E^d m$ et $|G(a,b)| \approx Em$ où E est la borne sur $|a|$ et $|b|$.

But

Le point clé des algorithmes de log discret et factorisation est que les entiers qui doivent être friables soient choisis aussi petits que possible.

Ordre de grandeur

- Écrire la borne de friabilité $B = L(1/3)$, la borne sur les pairs (a,b), $E = L(1/3)$ et $d = (\log N / \log\log N)^{1/3}$. Alors
  - $F(a,b) = L(1/3)^d L(1)^{1/d} = L(2/3)$;
  - $G(a,b) = L(1/3) L(1)^{1/d} = L(2/3)$;
- probabilité de friabilité $= 1/L(2/3 - 1/3) = 1/L(1/3)$ donc on requiert $L(1/3)$ paires a, b.
- OK car $E = L(1/3)$.

Le crible algébrique (NFS): diagramme

NFS for DLP in $\mathbb{F}_p$

Soient $f, g \in \mathbb{Z}[x]$ deux polynômes irréductibles, qui ont une racine commune m modulo p.

$$\begin{tikzcd}
 & a - bx \in \mathbb{Z}[x] \arrow[dl] \arrow[dr] & \\
\mathbb{Z}[x]/\langle f(x) \rangle \arrow[dr] & & \mathbb{Z}[x]/\langle g(x) \rangle \arrow[dl] \\
 & \mathbb{Z}/p\mathbb{Z} &
\end{tikzcd}$$

Calculs dans $\mathbb{Z}[\alpha_f]$?

- Les parties mathématiques sont négligeables en temps, mais cause de bugs.
- Implantations disponibles: PARI/GP, Magma, CADO.

Le crible algébrique (NFS): diagramme

NFS for DLP in $\mathbb{F}_p$

Soient $f, g \in \mathbb{Z}[x]$ deux polynômes irréductibles, qui ont une racine commune m modulo p.

$$\begin{tikzcd}
 & a - bx \in \mathbb{Z}[x] \arrow[dl, "x \mapsto \alpha_f"'] \arrow[dr, "x \mapsto \alpha_g"] & \\
\mathbb{Z}[\alpha_f] \arrow[dr, "\alpha_f \mapsto m"'] & & \mathbb{Z}[\alpha_g] \arrow[dl, "\alpha_g \mapsto m"] \\
 & \mathbb{Z}/p\mathbb{Z} &
\end{tikzcd}$$

Calculs dans $\mathbb{Z}[\alpha_f]$?

- Les parties mathématiques sont négligeables en temps, mais cause de bugs.
- Implantations disponibles: PARI/GP, Magma, CADO.

NFS: algorithme pour logs discrets

Input a finite field $\mathbb{F}_{p^n}$, two elements t (generator) and s
Output $\log_t s$

1: (Polynomial selection) Choose two polynomials f and g in $\mathbb{Z}[x]$ such that one has the diagram presented before;

2: (Sieve) Collect coprime pairs (a, b) such that $F(a, b)$ and $G(a, b)$ are B-smooth (for a parameter B);

3: Write a linear equation for each pair (a, b) found in the Sieve stage.

4: (Linear algebra) Solve the linear system to find (virtual) logarithms of the prime ideals of norm less than B;

5: (Individual logarithm) Write $\log_t s$ in terms of the previously computed logs.

Base de facteurs

Attention: on factorise en idéaux et non pas en éléments. On a $F(a, b)G(a, b)$ friable si et seulement si $(a - \alpha_f b)$ et $(a - b\alpha_g)$ se factorise en idéaux de la base de facteurs.

NFS: algorithme pour factorisation

Input an integer N
Output with probability 50% a non-trivial factor of N

1: (Polynomial selection) Choose two polynomials f and g in $\mathbb{Z}[x]$ such that one has the diagram presented before;

2: (Sieve) Collect coprime pairs (a, b) such that $F(a, b)$ and $G(a, b)$ are B-smooth (for a parameter B);

3: Write an exponent vector for each pair (a, b) found in the Sieve stage, <u>modulo 2</u>.

4: (Linear algebra) Find a linear combination of the rows of M which sum to zero;

5: (Square root) Compute a product in the number fields to obtain $X^2 \equiv Y^2 \mod N$.

Probabilité de Succés

Quand on utilise Block Wiedemann on calcule au moins 32 solutions à la fois. On répète uniquement le calcule de racine carré (x et y à partir de $x^2 \equiv y^2 \pmod{N}$). On réussit avec probabilité $1 - 2^{-32}$.

NFS: crible

Algorithme naïf (1989)

1. Pour f, enumérer les entiers a et , pour chacun, cribler le polynôme $F(a,x)$; obtenir les paires (a,b) pour lequelles $F(a,b)$ est friable.
2. Pour g, faire de même pour trouver les paires (a,b) pour lesquelles $G(a,b)$ est friable.
3. Intersecter les deux ensembles.

Special-q (1993)

Étant donné un premier q, et une racine r de $f(r) \equiv 0 \mod q$, on calcule deux reconstructions rationnelles (EEA)

$$r \equiv \frac{a_0}{b_0} \equiv \frac{a_1}{b_1} \mod q,$$

avec $a_0, b_0, a_1, b_1 \approx \sqrt{q}$. Ensuite on trouve par crible les paires (i,j) telles que

- $F(a_0 i + a_1 j, b_0 i + b_1 j)/q$ est B-friable;
- $G(a_0 i + a_1 j, b_0 i + b_1 j)$ est B-friable.

<u>intérêt</u> Sans ajouter quasimment pas de travail, on vise précisément les paires qui ont un facteur q grand, mais inférieur à la borne de friabilité.

Franke-Kleinjung (2009)

Calculer l'intersection d'un réseau de dimension 2 c'est rapide.

Conclusion

- Les algorithmes de cryptanayse requièrent de l'algèbre creuse
  - $\mathbb{F}_2$ pour factorisation;
  - $\mathbb{F}_\ell$ avec ℓ grand pour le log discret;
  - le parallélisme est un problème, mais certains algorithmes comme Block-Wiedemann permettent un parallélisme parfait pour (de l'ordre de) 10 centre de calcul.
- NFS est le meilleur algorithme aussi bien pour la factorisation que pour le log discret en $\mathbb{F}_p$ (et grande caractéristique).
- NFS a une complexité de $L(1/3)$ car il teste la frabilité de nombres de taille $L(2/3)$.
- NFS a été accéléré, particulièrement sur le crible et sur le parallélisme de l'algèbre linéaire.