Logarithme discret dans les courbes hyperelliptiques

Jean-François Biasse
(LIX-École Polytechnique)

21 octobre 2008

Courbes hyperelliptiques

Une courbe hyperelliptique $\mathcal{H}$ sur un corps $\mathbb{K}$ est la donnée des solutions de l'équation :

$$Y^2 + v(X)Y + u(X) = 0$$

Courbes hyperelliptiques

Une courbe hyperelliptique $\mathcal{H}$ sur un corps $\mathbb{K}$ est la donnée des solutions de l'équation :

$$Y^2 + v(X)Y + u(X) = 0$$

Tel qu'il existe un entier g appelé le **genre** de la courbe tel que

- $deg(u) = 2g + 1$.
- $deg(v) \leq g$

Jacobienne

Une courbe hyperelliptique $\mathcal{H}$ sur un corps $\mathbb{K}$ définit une variété algébrique $\mathcal{J}$ appelée la **Jacobienne**. C'est un groupe abélien dont les éléments sont appelés des **diviseurs**. On a en particulier :

$$\#\mathcal{J} \sim \#\mathbb{K}^g$$

Jacobienne

Une courbe hyperelliptique $\mathcal{H}$ sur un corps $\mathbb{K}$ définit une variété algébrique $\mathcal{J}$ appelée la **Jacobienne**. C'est un groupe abélien dont les éléments sont appelés des **diviseurs**. On a en particulier :

$$\#\mathcal{J} \sim \#\mathbb{K}^g$$

Tout diviseur D de $\mathcal{J}$ peut être représenté de manière unique sous la forme (a, b) avec $a, b \in \mathbb{K}[X]$ et :

- $deg(a) \leq g$
- $deg(b) < deg(a)$

Arithmétique et log discret

On peut définir $(a, b) = (a_1, b_1) + (a_2, b_2)$ sur $\mathcal{J}$ via deux opération :

Composition

- $d = gcd(a_1, a_2, b_1 + b_2 + v) = u_1 a_1 + u_2 a_2 + u_3(b_1 + b_2 + v)$
- $a = \frac{a_1 a_2}{d^2}$
- $b = b_1 + \frac{u_1 a_1 (b_2 - b_1) - u_3 (b_1^2 + b_1 v - u)}{d} \mod a$

Arithmétique et log discret

On peut définir $(a, b) = (a_1, b_1) + (a_2, b_2)$ sur $\mathcal{J}$ via deux opération :

Composition

- $d = gcd(a_1, a_2, b_1 + b_2 + v) = u_1 a_1 + u_2 a_2 + u_3(b_1 + b_2 + v)$
- $a = \frac{a_1 a_2}{d^2}$
- $b = b_1 + \frac{u_1 a_1 (b_2 - b_1) - u_3 (b_1^2 + b_1 v - u)}{d} \ mod \ a$

Réduction

- $a_k = \frac{b_{k-1}^2 + b_{k-1} v - u}{a_{k-1}}$
- $b_k = -b_{k-1} - v \ mod \ a_k$

Arithmétique et log discret

On peut définir $(a, b) = (a_1, b_1) + (a_2, b_2)$ sur $\mathcal{J}$ via deux opération :

Composition

- $d = gcd(a_1, a_2, b_1 + b_2 + v) = u_1 a_1 + u_2 a_2 + u_3(b_1 + b_2 + v)$
- $a = \frac{a_1 a_2}{d^2}$
- $b = b_1 + \frac{u_1 a_1 (b_2 - b_1) - u_3 (b_1^2 + b_1 v - u)}{d} \ mod \ a$

Réduction

- $a_k = \frac{b_{k-1}^2 + b_{k-1} v - u}{a_{k-1}}$
- $b_k = -b_{k-1} - v \ mod \ a_k$

Étant donnés deux diviseurs D_1 et D_2, résoudre le **DLP** c'est trouver α tel que :

$$D_2 = \alpha D_1$$

Quelques algorithmes classiques pour le log discret

On suppose que l'on se place dans le cas générique d'un groupe cyclique $G = \langle D \rangle$ d'ordre N.

Quelques algorithmes classiques pour le log discret

On suppose que l'on se place dans le cas générique d'un groupe cyclique $G = \langle D \rangle$ d'ordre N.

- Algorithme naif : On essaye tous les $[i]\,D$. $O(N)$.

Quelques algorithmes classiques pour le log discret

On suppose que l'on se place dans le cas générique d'un groupe cyclique $G = \langle D \rangle$ d'ordre N.

- Algorithme naif : On essaye tous les $[i]\,D$. $O(N)$.
- Pohlig-Helmann, idée (simplifiée) : si $p|N$ on a bien :

$$\frac{N}{p} D = \alpha \left(\frac{N}{p} D_1 \right)$$

si on sait résoudre ce problème sur le sous groupe d'ordre p, alors on obtient $\alpha \mod p$. Puis par CRT on en déduit α.

Quelques algorithmes classiques pour le log discret

On suppose que l'on se place dans le cas générique d'un groupe cyclique $G = \langle D \rangle$ d'ordre N.

- Algorithme naif : On essaye tous les $[i]\,D$. $O(N)$.
- Pohlig-Helmann, idée (simplifiée) : si $p|N$ on a bien :

$$\frac{N}{p}D = \alpha\left(\frac{N}{p}D_1\right)$$

si on sait résoudre ce problème sur le sous groupe d'ordre p, alors on obtient $\alpha \mod p$. Puis par CRT on en déduit α.

- Pollard-ρ : On crée des relations de la forme $\alpha_i D + \beta_i D_1 = F_i$ jusqu'à trouver une collision $F_i = F_j$. $O\left(\sqrt{N}\right)$.

Descente de Weil

On considère le DLP sur une courbe elliptique ($g = 1$) sur le corps $\mathbb{F}_{2^{kn}}$. Cependant, dans certains cas, il peut être ramené via une stratégie due à (Gaudry-Hess-Smart) appelée **descente de Weil** au même problème sur une courbe $\mathcal{H}$ vérifiant :

- $g = 2^{m-1}$ ou $2^{m-1} - 1$
- $\mathbb{K} = \mathbb{F}_{2^k}$

Descente de Weil

On considère le DLP sur une courbe elliptique ($g = 1$) sur le corps $\mathbb{F}_{2^{kn}}$. Cependant, dans certains cas, il peut être ramené via une stratégie due à (Gaudry-Hess-Smart) appelée **descente de Weil** au même problème sur une courbe $\mathcal{H}$ vérifiant :

- $g = 2^{m-1}$ ou $2^{m-1} - 1$
- $\mathbb{K} = \mathbb{F}_{2^k}$

L'idée est de réécrire la variable X comme :

$$X = (x_1, ..., x_{kn})$$

et de considérer le système d'équations qui en résulte.
Dans la suite nous considérons le DLP sur une courbe de genre g sur $\mathbb{F}_{2^k} = \mathbb{F}_q$.

Friabilité

On définit une base de friabilité $\mathcal{B}$ comme étant l'ensemble des diviseur de degré 1 ($deg(a) = 1$). On dit que $D \in \mathcal{J}$ est **friable** si il existe $P_1, ..., P_g \in \mathcal{B}$ tels que :

$$D = \sum_{i=1}^{g} P_i$$

Friabilité

On définit une base de friabilité $\mathcal{B}$ comme étant l'ensemble des diviseur de degré 1 ($deg(a) = 1$). On dit que $D \in \mathcal{J}$ est **friable** si il existe $P_1, ..., P_g \in \mathcal{B}$ tels que :

$$D = \sum_{i=1}^{g} P_i$$

- $D = (a, b)$ est friable si $a = \prod(X - x_i)$, ce qui se teste facilement par :$deg\left(pgcd\left(X^q - X, a\right)\right) = g$

Friabilité

On définit une base de friabilité $\mathcal{B}$ comme étant l'ensemble des diviseur de degré 1 ($deg(a) = 1$). On dit que $D \in \mathcal{J}$ est **friable** si il existe $P_1, ..., P_g \in \mathcal{B}$ tels que :

$$D = \sum_{i=1}^{g} P_i$$

- $D = (a, b)$ est friable si $a = \prod(X - x_i)$, ce qui se teste facilement par :$deg\left(pgcd\left(X^q - X, a\right)\right) = g$
- Dans ce cas la décomposition de D est donnée par des éléments de $\mathcal{B}$ de la forme $P_i = ((X - x_i), b_i)$

algorithme

- On suppose que $\mathcal{J} = \langle P \rangle$ et on cherche à calculer le DLP de chaque $P_i \in \mathcal{B}$. Pour cela on cherche des n_i tels que :

$$[n_i]\, P = \sum n_{ij} P_j$$

algorithme

- On suppose que $\mathcal{J} = \langle P \rangle$ et on cherche à calculer le DLP de chaque $P_i \in \mathcal{B}$. Pour cela on cherche des n_i tels que :

$$[n_i]\, P = \sum n_{ij} P_j$$

- On cherche ensuite un vecteur du noyau de la matrice (n_{ij}).

En effet si on a un tel vecteur on le renormalise par rapport à la composante correspondant à P (on suppose que $P \in \mathcal{B}$).

Élimination de colonnes

- Pour les tailles crypto, la matrice est souvent trop grosse pour être traitée, on cherche donc à réduire ses dimensions.

Élimination de colonnes

- Pour les tailles crypto, la matrice est souvent trop grosse pour être traitée, on cherche donc à réduire ses dimensions.
- Les colonnes vides doivent être supprimées (trivial)

Élimination de colonnes

- Pour les tailles crypto, la matrice est souvent trop grosse pour être traitée, on cherche donc à réduire ses dimensions.
- Les colonnes vides doivent être supprimées (trivial)
- Les colonnes de poid 1 aussi, ainsi que la ligne correspondante en la sauvegardant : on pourra ainsi déduire le log discret du diviseur associé après la recherche du vecteur du noyau.

Élimination de colonnes

- Pour les tailles crypto, la matrice est souvent trop grosse pour être traitée, on cherche donc à réduire ses dimensions.
- Les colonnes vides doivent être supprimées (trivial)
- Les colonnes de poid 1 aussi, ainsi que la ligne correspondante en la sauvegardant : on pourra ainsi déduire le log discret du diviseur associé après la recherche du vecteur du noyau.
- Les colonnes de poid supérieur sont traitée en ordre croissant via une élimination de Gauss.

Problème : On a une densification de la matrice ainsi qu'une augmentation générale de la valeur des coefficients.

Élimination structurée

On cherche à limiter la densification et l'augmentation des coefficients lors du traitement de la ligne j_0. On crée un graphe de la manière suivante :

Élimination structurée

On cherche à limiter la densification et l'augmentation des coefficients lors du traitement de la ligne j_0. On crée un graphe de la manière suivante :

- On liste les lignes $L_1, ..., L_n$ apparaissant dans la colonne j_0. Ils sont les sommets.
- Pour chaque i_1, i_2 on trace une arrête entre les deux sommets de poid $|L_{i_1} - c_2 L_{i_2}|$.

Élimination structurée

On cherche à limiter la densification et l'augmentation des coefficients lors du traitement de la ligne j_0. On crée un graphe de la manière suivante :

- On liste les lignes $L_1, ..., L_n$ apparaissant dans la colonne j_0. Ils sont les sommets.
- Pour chaque i_1, i_2 on trace une arrête entre les deux sommets de poid $|L_{i_1} - c_2 L_{i_2}|$.

Une fois un tel graphe créé, on crée son arbre couvrant de poid minimal via l'algorithme glouton suivant :

Élimination structurée

On cherche à limiter la densification et l'augmentation des coefficients lors du traitement de la ligne j_0. On crée un graphe de la manière suivante :

- On liste les lignes $L_1, ..., L_n$ apparaissant dans la colonne j_0. Ils sont les sommets.
- Pour chaque i_1, i_2 on trace une arrête entre les deux sommets de poid $|L_{i_1} - c_2 L_{i_2}|$.

Une fois un tel graphe créé, on crée son arbre couvrant de poid minimal via l'algorithme glouton suivant :

- Tant qu'il reste des sommets à traiter, prendre le sommet qui a une arête avec l'arbre de poid le plus faible.
- Ajouter ce sommet à l'arbre

Élimination structurée (suite)

Une fois l'arbre couvrant de poid minimal obtenu via la méthode précédente on procède comme suit :

Élimination structurée (suite)

Une fois l'arbre couvrant de poid minimal obtenu via la méthode précédente on procède comme suit :

- On recherche une feuille L_{i_1} de l'arbre.
- On considère son père L_{i_2}.
- On effectue l'opération $L_{i_2} \longleftarrow L_{i_2} - c_2 L_{i_1}$
- On détache L_{i_1} de l'arbre.
- On supprime la racine de l'arbre.
- On supprime la colonne.

Élimination structurée (suite)

Une fois l'arbre couvrant de poid minimal obtenu via la méthode précédente on procède comme suit :

- On recherche une feuille L_{i_1} de l'arbre.
- On considère son père L_{i_2}.
- On effectue l'opération $L_{i_2} \longleftarrow L_{i_2} - c_2 L_{i_1}$
- On détache L_{i_1} de l'arbre.
- On supprime la racine de l'arbre.
- On supprime la colonne.

Rq : Le critère retenu jusqu'à présent pour étiqueter les arêtes était le poid de la ligne créée par soustraction. Ce critère n'est pas forcément le plus pertinent. On peut par exemple pénaliser plus fortement l'augmentation globale des coefficients.

Large primes

Plusieurs variantes existes pour équilibrer l'effort entre le crible et l'algèbre linéaire :

Large primes

Plusieurs variantes existes pour équilibrer l'effort entre le crible et l'algèbre linéaire :

- (Harley) Réduire la base de facteurs $\mathcal{B}$ à un sous ensemble. Les relations sont plus dures à trouver, mais les dimensions de la matrice sont plus intéressantes.

Large primes

Plusieurs variantes existes pour équilibrer l'effort entre le crible et l'algèbre linéaire :

- (Harley) Réduire la base de facteurs $\mathcal{B}$ à un sous ensemble. Les relations sont plus dures à trouver, mais les dimensions de la matrice sont plus intéressantes.
- (Thériault) Écrire $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ et autoriser seulement un élément dans $\mathcal{B}_2$:

$$[n]\,P = \underbrace{P_1 + \ldots + P_n}_{\in \mathcal{B}_1} + \underbrace{Q}_{\in \mathcal{B}_2}$$

La soustraction de deux lignes contenant Q donne une ligne dont les éléments sont dans $\mathcal{B}_1$.

Large primes (suite)

- (Gaudry,Thomé) On peut autoriser deux large primes :

$$[n]\,P = \underbrace{P_1 + \ldots + P_n}_{\in \mathcal{B}_1} + \underbrace{Q_1 + Q_2}_{\in \mathcal{B}_2}$$

Large primes (suite)

- (Gaudry,Thomé) On peut autoriser deux large primes :

$$[n]\,P = \underbrace{P_1 + \ldots + P_n}_{\in \mathcal{B}_1} + \underbrace{Q_1 + Q_2}_{\in \mathcal{B}_2}$$

On indexe alors une ligne par les deux indices des large primes intervenant (ex plus haut (1,2)) et on forme des cycles :

$$(i_1, i_2), (i_2, i_3), \ldots, (i_k, i_1)$$

Large primes (suite)

- (Gaudry,Thomé) On peut autoriser deux large primes :

$$[n]\,P = \underbrace{P_1 + \ldots + P_n}_{\in \mathcal{B}_1} + \underbrace{Q_1 + Q_2}_{\in \mathcal{B}_2}$$

On indexe alors une ligne par les deux indices des large primes intervenant (ex plus haut (1,2)) et on forme des cycles :

$$(i_1, i_2), (i_2, i_3), \ldots, (i_k, i_1)$$

En menant l'élimination dans l'ordre du cycle on obtient une ligne avec seulement des $P_i \in \mathcal{B}_1$.

- On peut autoriser plus de large primes. Pas d'élimination particulière mais une bande dense.

Illustration : 2 large primes

On suppose que $\#\mathcal{B} = \frac{1}{2}q^r \ \ r \in]0,1[$

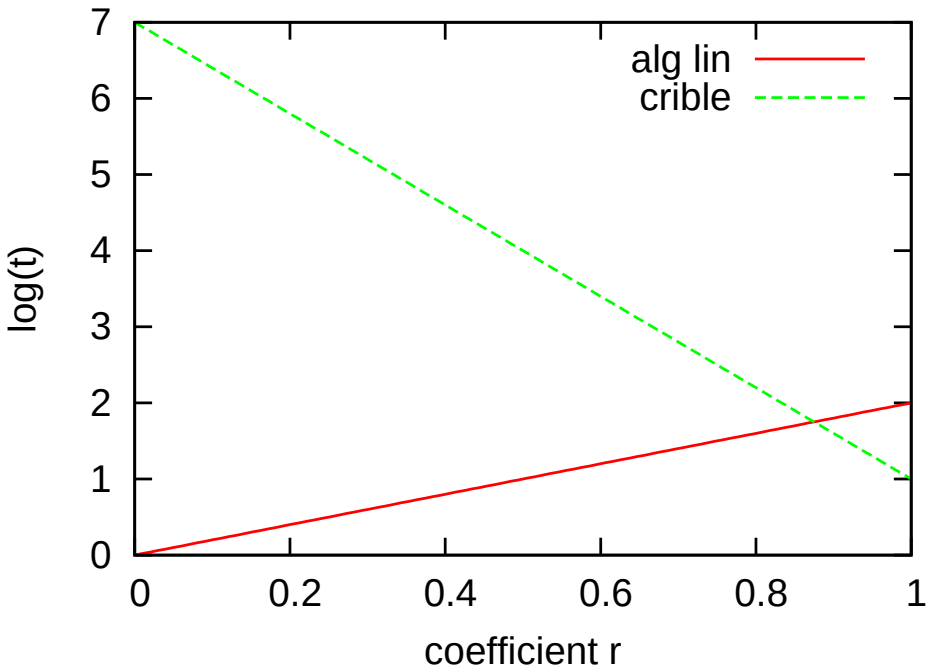

FIG.: Optimum de la taille de la base de facteurs

Complexité

Pour l'étude de l'algorithme de calcul de logarithme discret présenté il faut définir la fonction sous exponentielle :

$$L_N(\alpha, c) = \exp\left(c \log(N)^{\alpha} \log\log(N)^{1-\alpha}\right)$$

Complexité

Pour l'étude de l'algorithme de calcul de logarithme discret présenté il faut définir la fonction sous exponentielle :

$$L_N(\alpha, c) = \exp\left(c \log(N)^{\alpha} \log\log(N)^{1-\alpha}\right)$$

On voit bien que :

- $L_N(0, c) = \log(N)^c$
- $L_N(1, c) = N^c$

Complexité

Pour l'étude de l'algorithme de calcul de logarithme discret présenté il faut définir la fonction sous exponentielle :

$$L_N(\alpha, c) = \exp\left(c \log(N)^{\alpha} \log\log(N)^{1-\alpha}\right)$$

On voit bien que :

- $L_N(0, c) = \log(N)^c$
- $L_N(1, c) = N^c$

Les algorithmes à base de calcul d'index sur les courbes hyperelliptiques ont une complexité en :

$$L_{q^g}\left(\frac{1}{2}, C\right)$$

lorsque $g \to \infty$ et $g > \log q$.

Complexité(2)

Soit $q = \#\mathbb{K}$. Sans stratégie de large prime variation, à g fixé la complexité est en :

Complexité(2)

Soit $q = \#\mathbb{K}$. Sans stratégie de large prime variation, à g fixé la complexité est en :

- $O(q)$ pour la recherche des relations.

Complexité(2)

Soit $q = \#\mathbb{K}$. Sans stratégie de large prime variation, à g fixé la complexité est en :

- $O(q)$ pour la recherche des relations.
- $O(q^2)$ pour l'algèbre linéraire

Complexité(2)

Soit $q = \#\mathbb{K}$. Sans stratégie de large prime variation, à g fixé la complexité est en :

- $O(q)$ pour la recherche des relations.
- $O(q^2)$ pour l'algèbre linéraire

Dans le cas des 2 large primes on a :

- $O\left(q^{1+\frac{g-1}{g+1}}\right)$ pour la recherche des relations.

Complexité(2)

Soit $q = \#\mathbb{K}$. Sans stratégie de large prime variation, à g fixé la complexité est en :

- $O(q)$ pour la recherche des relations.
- $O(q^2)$ pour l'algèbre linéraire

Dans le cas des 2 large primes on a :

- $O\left(q^{1+\frac{g-1}{g+1}}\right)$ pour la recherche des relations.
- $O\left(q^{2-\frac{2}{g+1}}\right)$ pour l'algèbre linéaire.

L'exemple

On dispose de la courbe elliptique :

$$E : y^2 + xy = x^3 + ax^2 + b$$

sur $\mathbb{F}_{2^{161}} = \mathbb{F}_2[z]/(z^{161} + z^{18} + 1)$.

L'exemple

On dispose de la courbe elliptique :

$$E : y^2 + xy = x^3 + ax^2 + b$$

sur $\mathbb{F}_{2^{161}} = \mathbb{F}_2[z]/(z^{161} + z^{18} + 1)$.

Via la descente de Weil on peut ramener le problème du logarithme discret sur une courbe

$$H : Y^2 + v(X)Y + u(x)$$

sur $\mathbb{F}_{2^{23}} = \mathbb{F}_2[z]/(z^{23} + z^5 + 1)$ avec $g = 8$ car $deg(u) = 17$ et $deg(v) = 8$

Phase de crible

La jacobienne de H est isomorphe à :

$$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$$

On a donc deux générateurs D_1 et D_2 et on cherche des $\alpha_i D_1 + \beta_i D_2$ friables.

Phase de crible

La jacobienne de H est isomorphe à :

$$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$$

On a donc deux générateurs D_1 et D_2 et on cherche des $\alpha_i D_1 + \beta_i D_2$ friables.

Sur 300 processeurs en parallèle on obtient environs 10^6 relations par jour. (machines des salles informatique des élèves à Polytechnique $\sim$ 2 GHZ)

Réduction de la matrice

On obtient une matrice avec :

- 4.10^6 colonnes.
- 8 entrées non nulles par lignes (toutes 1 ou -1)

Réduction de la matrice

On obtient une matrice avec :

- 4.10^6 colonnes.
- 8 entrées non nulles par lignes (toutes 1 ou -1)

Après une réduction jusqu'aux colonnes de poid 16 on obtient

- environs $2,6.10^6$ colonnes

Réduction de la matrice

On obtient une matrice avec :

- 4.10^6 colonnes.
- 8 entrées non nulles par lignes (toutes 1 ou -1)

Après une réduction jusqu'aux colonnes de poid 16 on obtient

- environs $2,6.10^6$ colonnes
- 650 entrées non nulles par ligne.

Réduction de la matrice

On obtient une matrice avec :

- 4.10^6 colonnes.
- 8 entrées non nulles par lignes (toutes 1 ou -1)

Après une réduction jusqu'aux colonnes de poid 16 on obtient

- environs $2,6.10^6$ colonnes
- 650 entrées non nulles par ligne.
- 50 entrées non 1 ou -1

Et les large primes ?

On se propose de réduire la taille de la base de facteurs afin de soulager l'algèbre linéaire. Étudions la difficulté de calcul pour que :

$$\#\mathcal{B}' = \mathcal{B}/2 = 2.10^6$$

Et les large primes ?

On se propose de réduire la taille de la base de facteurs afin de soulager l'algèbre linéaire. Étudions la difficulté de calcul pour que :

$$\#\mathcal{B}' = \mathcal{B}/2 = 2.10^6$$

- Réduction simple de la base de facteurs : 2^7 fois plus lent.

Et les large primes ?

On se propose de réduire la taille de la base de facteurs afin de soulager l'algèbre linéaire. Étudions la difficulté de calcul pour que :

$$\#\mathcal{B}' = \mathcal{B}/2 = 2.10^6$$

- Réduction simple de la base de facteurs : 2^7 fois plus lent.
- 1 large prime : 2^6 fois plus lent.

Et les large primes ?

On se propose de réduire la taille de la base de facteurs afin de soulager l'algèbre linéaire. Étudions la difficulté de calcul pour que :

$$\#\mathcal{B}' = \mathcal{B}/2 = 2.10^6$$

- Réduction simple de la base de facteurs : 2^7 fois plus lent.
- 1 large prime : 2^6 fois plus lent.
- 2 large primes : $2^{5,95}$ fois plus lent.

conclusion

Merci de votre attention