Untere Schranken für das DDH

Für konkret gegebene Gruppen gibt es keine (sinnvollen) unteren Schranken für die Laufzeit, ein DLP zu lösen. Für „allgemeine“ Gruppen gibt es dies aber.

Eine Numeration ist eine Injektion $\sigma : (\mathbb{Z}/\ell\mathbb{Z})^+ \to \{0,1\}^n$.

Ein generischer Algorithmus erwartet als Eingabe eine Liste von Gruppenelementen $L = (\sigma(x_1), \ldots, \sigma(x_k))$ und hat Zugang zu einem Orakel, welches die Werte $\sigma(x_i \pm x_j)$ berechnet. Solche neu berechneten Werte werden zu L hinzugefügt.

Alternativ kann man sich vorstellen, die Gruppe sei durch eine Klasse gegeben, und der generische Algorithmus kann nur die Methoden $+, -, =, <$ aufrufen (sonst nichts). $\Rightarrow$ Black-Box Gruppen.

Untere Schranken für das DDH

Thm: Sei ℓ prim. Das DDH kann durch einen generischen Algorithmus A bei zufälliger Numeration σ und $\leq m$ Orakelanfragen an σ mit von $1/2$ um maximal $\approx m^2/\ell$ abweichender Wahrscheinlichkeit gelöst werden.

Bew: A erhält $\sigma(1), \sigma(a), \sigma(b), w_s, w_{1-s}$ mit $w_0 = \sigma(ab)$, $w_1 = \sigma(c)$, $s \in \{0,1\}$ und $a, b, c \in \mathbb{Z}/\ell\mathbb{Z}$ zufällig. Die durch A berechneten Elemente sind von der Form $\sigma(F_j(a,b,c))$ für $F_j(t_1,t_2,t_3) = \lambda_{j,1}t_1 + \lambda_{j,2}t_2 + \lambda_{j,3}t_3 + \lambda_{j,4}t_1t_2 + \lambda_{j,5}$ und $\lambda_{j,i} \in \mathbb{Z}/\ell\mathbb{Z}$. A kann nur dann Information erhalten, wenn $\sigma(F_i(a,b,c)) = \sigma(F_j(a,b,c))$ für i, j mit $F_i \neq F_j$. Tritt dies nicht ein, hat A Erfolgswahrscheinlichkeit $1/2$. Wir suchen also nach der Wahrscheinlichkeit, daß ein $G_{i,j} = F_i - F_j$ eine Nullstelle (a,b,c) hat. Diese ist gleich Null für $G_{i,j}$ konstant, und gleich $1/\ell$ sonst. Es gibt $m(m-1)/2$ Polynome $G_{i,j}$, und somit eine Gesamtwahrscheinlichkeit $\leq m(m-1)/(2\ell) \leq m^2/\ell$. $\square$

Untere Schranken für das DDH

Der Satz gilt analog für das DLP, Erfolgswahrscheinlichkeit $\leq m^2/\ell$.

Um konstante Erfolgswahrscheinlichkeit zu haben, benötigt man also m = Omega($\sqrt{\ell}$) viele Orakelanfragen. Die Laufzeit ist demnach exponentiell in $\log(\ell)$.

Mit den Pollard Methoden ergibt sich im folgenden, daß $O(\sqrt{\ell})$ auch eine obere Schranke für die benötigte Zeit ist, folglich ist Theta($\sqrt{\ell}$) die genaue Komplexität für generische Algorithmen bzw. Black-Box Gruppen.

Shanks Baby Step Giant Step

Ist Anwendung von Meet-in-the-Middle Prinzip. Löst DLP deterministisch in Zeit $O(\sqrt{\ell})$ und Speicher $O(\sqrt{\ell})$.

Schreibe x mit $0 \leq x < \ell$ eindeutig als $x = jm + i$ mit $0 \leq i < m$ und $0 \leq j \leq \ell/m$. Schreibe $x \mapsto g^x$ als $(i,j) \mapsto g^{jm+i}$.

Seien g, b gegeben und x mit $b = g^x$ gesucht.
Trenne i und j: Für $x = jm + i$ gilt $b/g^i = g^{jm}$.

Daher linke Seite für alle i ausrechnen und speichern, dann alle j für Kollision durchprobieren.

Zeit-optimiert für $m \approx \sqrt{\ell}$.

Pollard rho

Analog wie beim Kollisionsfinden für Hashfunktionen. Löst DLP probabilistisch in erwarteter Zeit $O(\sqrt{\ell})$ und Speicher $O(1)$ (im RO).

Idee: Wir brauchen einen Zufallsweg (random walk) in G von der Form $b^{u_i}g^{v_i}$ mit bekannten u_i, v_i, welche nur von $b^{u_{i-1}}g^{v_{i-1}}$ abhängen. Eine Kollision $b^{u_i}g^{v_i} = b^{u_j}g^{v_j}$ liefert dann $b^{u_i-u_j} = g^{v_j-v_i}$ und folglich $x = (v_j - v_i)/(u_i - u_j) \bmod \ell$, wenn $(u_i - u_j) \not\equiv 0 \bmod \ell$.

Definition von u_i, v_i ($u_0 = 0$, $v_0 = 0$):

- durch $u_i = H(b^{u_{i-1}}g^{v_{i-1}}||0)$ und $v_i = H(b^{u_{i-1}}g^{v_{i-1}}||1)$ für eine Hashfunktion H.
- oder wie folgt: Zerlege G in disjunkte Teilmengen S_1, S_2, S_3 und definiere $(u_i, v_i) = \begin{cases} (u_{i-1}, v_{i-1}+1) & \text{für } b^{u_{i-1}}g^{v_{i-1}} \in S_1 \\ (u_{i-1}+1, v_{i-1}) & \text{für } b^{u_{i-1}}g^{v_{i-1}} \in S_2 \\ (2u_{i-1}, 2v_{i-1}) & \text{für } b^{u_{i-1}}g^{v_{i-1}} \in S_3 \end{cases}$.

Pohlig-Hellman

Beruht auf dem Struktursatz für endlich erzeugte, abelsche Gruppen.

Löst DLP für G beliebig zyklisch, ℓ_0 größter Primfaktor von $\#G$, durch Shanks oder Pollard Methoden in Laufzeit $\text{poly}(\log(\#G))\sqrt{\ell_0}$ und Speicher $O(\sqrt{\ell_0})$ oder $O(1)$.

Idee 1 (chinesischer Restsatz):

- Sei $\#G = \prod_{i=0}^{r} \ell_i^{e_i}$, $n_i = 1 \bmod \ell_i^{e_i}$ und $n_i = 0 \bmod \ell_j^{e_j}$ für alle $j \neq i$, $\phi_i : G \to G$ mit $\phi(z) = z^{n_i}$. Wegen $\gcd\{n_0, \ldots, n_r\} = 1$ gilt $\cap_i \ker(\phi_i) = 1$.
- Die Ordnung von $G_i = \phi_i(G)$ ist $\ell_i^{e_i}$.
- DLP in jedem G_i lösen (falls lösbar) und mit chinesischem Restsatz zusammensetzen: Finde x_i mit $\phi_i(b) = \phi_i(g)^{x_i}$. Dann ist $x = \sum_i x_i n_i$ der DL.

Pohlig-Hellman

Idee 2 („Hensel Lifting“):

- Annahme $\#G = \ell^e$, ℓ prim und $g^{\ell^e} = 1$ mit e minimal.
- Dann $b = g^x$ mit $x = x_0 + x_1\ell + \ldots x_{e-1}\ell^{e-1}$ und $0 \leq x_i < \ell$.
- Löse induktiv DLPs $b^{\ell^{e-1-j}} / g^{\ell^{e-1-j}(x_0 + x_1\ell + \ldots x_{j-1}\ell^{j-1})} = g^{\ell^{e-1-j}x_j\ell^j}$ in x_j für $j = 0, \ldots, e-1$. Sind definiert in Gruppe $\langle g^{\ell^{e-1}} \rangle$ der Ordnung ℓ.

Daher betrachten wir nur zyklische Gruppen mit „möglichst“ primer Ordnung.
Die Verwendung nicht zyklischer Gruppen macht ebenfalls keinen Sinn, da wir nur in der von g erzeugten Untergruppe arbeiten.

Generische Methoden für das DLP

Gruppenordnung $\ell = c\ell_0$, ℓ_0 größter Primfaktor von ℓ.

- Shanks: deterministisch, Laufzeit $O(\sqrt{\ell})$, Speicher $O(\sqrt{\ell})$.
- Pollard rho: probabilistisch, Laufzeit $O(\sqrt{\ell})$, Speicher $O(1)$.
- Pohlig-Hellman: deterministische Reduktion auf Shanks oder Pollard rho, Laufzeit $\text{poly}(\log(\#G))\sqrt{\ell_0}$, Speicher $O(\sqrt{\ell_0})$ oder $O(1)$.

Name „rho“ wegen des Aussehens des Zufallswegs ...

Die Methoden von Shanks, Pollard und Pohlig-Hellman funktionieren in jeder Gruppe gleichermaßen (also für Black-Box Gruppen), wobei für Pohlig-Hellman noch die Faktorisierung der Gruppenordnung bekannt sein muß.

Laufzeit exponentiell in Bitlänge $\log_2(\ell_0)$.

Methoden für das DDH

Die besten Algorithmen für das DDH in Black-Box Gruppen mit Primordnung sind die Algorithmen für das DLP (vgl. den Satz über die Schwierigkeit des DDH).

Besitzt die Gruppenordnung kleine Primfaktoren, ist das DDH im allgemeinen nicht schwer, es gibt einen Algorithmus, der die richtige Entscheidung mit Wahrscheinlichkeit signifikant $> 1/2$ fällt.

Daher immer mit einer Untergruppe von großer, primer Ordnung arbeiten.

Index Calculus

Das DLP in $(\mathbb{Z}/\ell\mathbb{Z}, +)$ ist leicht, weil wir zusätzlich die Multiplikation verwenden können. Dies können wir in einer Black-Box Gruppe nicht.

Index Calculus Algorithmen basieren auf der Tatsache, daß gewisse Gruppen Faktorgruppen von Ringen (oder auch Gruppen) mit Primfaktorisierung und endlich vielen Primelementen beschränkter Größe sind.

Die Laufzeit dieser Algorithmen ist wesentlich besser als die der generischen Algorithmen (subexponentiell versus exponentiell).

Die unterliegende Technik von Index Calculus Algorithmen findet auch bei der Faktorisierung ganzer Zahlen Anwendung.

Index Calculus

Betrachte $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ und $G \subseteq \mathbb{F}_p^\times$ der Primordnung ℓ mit $\ell \| p-1$.
In $\mathbb{Z}$ gibt es Primfaktorisierung und nur endliche viele Primelemente beschränkter Größe.

Seien $g, b \in G$ mit $b = g^x$ und x gesucht.

- Sei $S = \{p_1, \ldots, p_s\}$ eine Menge von Primzahlen.
- Bestimme zufällige Werte $b^{u_i} g^{v_i}$, lifte sie nach $[0, p-1] \cap \mathbb{Z}$ und „faktorisiere“ sie über S. Geht dies, erhalten wir $b^{u_i} g^{v_i} = \prod_{j=1}^{s} p_j^{e_{i,j}}$. Wiederhole dies mindestens $s+1$ mal.
- Durch Anwendung von „$\log_g$“ erhalten wir die linearen Relationen $u_i x + v_i = \sum_{j=1}^{s} e_{i,j} \log_g(p_j) \bmod \ell$. Bei genügend vielen Relationen können wir z_i nicht alle Null mit $\sum_i z_i e_{i,j} = 0$ für alle j ausrechnen.

Dann gilt $\sum_i z_i(u_i x + v_i) = 0 \bmod \ell$, folglich $x = -(\sum_i z_i v_i)/(\sum_i z_i u_i) \bmod \ell$.

Index Calculus

Erinnerung Komplexitätsfunktion für $x \to \infty$:
$L_x(u, v) = \exp((v + o(1)) \log(x)^u \log(\log(x))^{1-u})$.

$L_x(1, v) = x^{v+o(1)}$, also exponentiell in $\log(x)$.
$L_x(0, v) = \log(x)^{v+o(1)}$, also polynomiell in $\log(x)$.
Für $0 < u < 1$ spricht man von subexponentiellem Wachstum in $\log(x)$.

Man kann mit $L_x(u, v)$ also zwischen exponentieller und polynomieller Laufzeit mitteln.

- DLP Pollard rho in $\mathbb{F}_p^\times$: $L_p(1, 1/2)$.
- Faktorisieren ganzer Zahlen n: $L_N(1/2, 1)$, $L_N(1/3, (64/9)^{1/3})$.

Index Calculus

Grobe Laufzeitanalyse für $p \to \infty$:

- $S = \{p \mid p \text{ prim und } p \leq y\}$, $\#S \approx y/\log(y)$ (Faktorbasis, Größe nach Primzahlsatz).
- $\Pr_{p,y} = \Pr(z \text{ mit } 1 \leq z < p \text{ faktorisiert über } S) \approx u^{-u}$ für $u = \log(p)/\log(y)$ und $\log(p)^{0.1} \leq u \leq \log(p)^{0.9}$ (Glattheitswahrscheinlichkeit).
- Erwarteter Aufwand, $(e_{i,j})_{i,j}$ zu finden: $\#S \cdot \Pr_{p,y}^{-1} \approx (y/\log(y))u^u$,
- also $\approx \exp((1+o(1))\log(y) + (\log(p)/\log(y))\log(\log(p)/\log(y)))$.
- Wird minimiert für $\log(y) = (\mu + o(1))(\log(p)\log(\log(p)))^{1/2}$, nimmt Wert $L_p(1/2, \mu + 1/(2\mu))$ an.
- Matrixschritt noch $L_p(1/2, 2\mu)$ mit schneller linearer Algebra. Optimaler Wert für minimale Laufzeit $\mu = \sqrt{1/2}$, daher insgesamt $L_p(1/2, \sqrt{2})$.

$\Rightarrow$ Subexponentielle Laufzeit! Pollard in $\mathbb{F}_p^\times$ nur $L_p(1, 1/2)$.

Index Calculus

Index Calculus kann auch auf andere endliche Körper $\mathbb{F}_q$ mit $q = p^n$ verallgemeinert werden ($\mathbb{F}_q = \mathbb{F}_p[t]/(f(t))$, $\mathbb{F}_p[t]$ hat Primfaktorisierung).

Es gibt sogar Varianten, welche eine Laufzeit von $L_q(1/3, c)$ haben:

- Varianten des Zahlkörpersiebs und des Funktionenkörpersiebs.

Auswirkung auf Sicherheit daher ähnlich (ungünstig) wie bei RSA.

- s Bit Sicherheit bei $L_q(u, v)$ Angriff benötigt $\approx s^{1/u}/v$ Bit Körpergröße q (qualitativ).
- Verdoppelung von s führt also zur Verdoppelung der Bitlänge im generischen Fall und zur Verachtfachung für $\mathbb{F}_q^\times$ und RSA!

Methoden für das DLP und DDH

Gruppenordnung $\ell = c\ell_0$, ℓ_0 größter Primfaktor von ℓ.

Generische Methoden (Shanks BSGS, Pollard rho, ...):

- Laufzeit poly($\log(\#G)$)$\sqrt{l_0}$, also exponentiell in $\log(\ell_0)$.

Index Calculus Methoden für $\mathbb{F}_q^\times$ mit $\ell \mid (q-1)$:

- Laufzeit $L_q(1/3, c)$, c Konstante.

Daher Sicherheit im Verhältnis zur Effizienz bei $\mathbb{F}_q^\times$ nicht wesentlich besser als bei RSA ...

Gibt es bessere Gruppen als $\mathbb{F}_q^\times$?