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. 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 Zeilen 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 $p \to \infty$:

- $S = \{p \mid p \text{ prim und } p \le y\}$, $\#S \approx y/\log(y)$ (Faktorbasis, Größe nach Primzahlsatz).
- $\Pr_{p,y} = \Pr(z \text{ mit } 1 \le z < p \text{ faktorisiert über } S) \approx u^{-u}$ für $u = \log(p)/\log(y)$ und $\log(p)^{0.1} \le u \le \log(p)^{0.9}$ (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 (Wiedemann). Optimaler Wert für minimale Laufzeit $\mu = \sqrt{1/2}$, daher insgesamt $L_p(1/2, \sqrt{2})$.

$\Rightarrow$ Subexponentielle Laufzeit! Pollard 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 Theta~$(\sqrt{l_0})$, 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 $\mathbb{F}_q^\times$?

Elliptische Kurven

Für Index Calculus muß man „liften“ und faktorisieren können. Gibt es Gruppen, wo dies nicht geht bzw. wo Pollard rho die (vermutlich) effizienteste Methode für das DDH ist?

$\Rightarrow$ elliptische Kurven, hyperelliptische Kurven kleinen Geschlechts.

Sei $K = \mathbb{F}_q$ mit $q = p^r$ und $p > 3$.

Eine elliptische Kurve wird durch eine Gleichung gegeben:
$E : Y^2 = X^3 + aX + b$ mit $a, b \in K$ und $4a^3 + 27b^2 \neq 0$.

Menge der Punkte über K: $E(K) = \{(x,y) \in K^2 \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$.
$\mathcal{O}$ ist formal der Punkt „im unendlichen“.

Hasse-Weil: $\#E(K) = q + 1 - t$, wobei $|t| \leq 2\sqrt{q}$.

- Heuristisch: Hälfte der quadratischen Gleichungen in y nach Einsetzen für x hat zwei Nullstellen in k, die andere keine.

Elliptische Kurven

Man kann $E(K)$ in eine abelsche Gruppe mit neutralem Element $\mathcal{O}$ machen. Gruppengesetz wird üblicherweise additiv geschrieben. Es gibt spezielle Formeln, mit der die Punkte „addiert“ werden:

Sei $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(K)$.

Dann $P + Q = \begin{cases} \mathcal{O} \text{ für } x_P = x_Q \text{ und } y_P = -y_Q, \\ (x_{P+Q}, -y_Q - \lambda(x_{P+Q} - x_Q)) \text{ andernfalls,} \end{cases}$ wobei

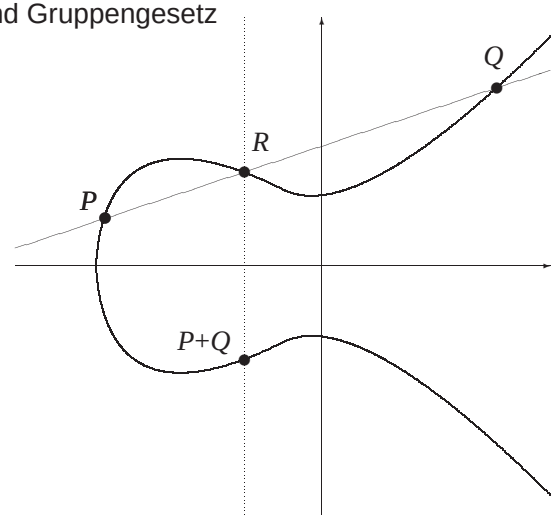
$x_{P+Q} = \lambda^2 - (x_P + x_Q)$ und $\lambda = \begin{cases} (y_Q - y_P)/(x_Q - x_P) \text{ für } x_P \neq x_Q, \\ (3x_P^2 + a)/(2y_P) \text{ andernfalls.} \end{cases}$

Das Nachrechnen der Gruppengesetze (insbesondere Assoziativität) ist recht umständlich bzw. benötigt mehr mathematische Theorie.

Das Gruppengesetz für elliptische Kurven über $K = \mathbb{R}$ kann geometrisch veranschaulicht werden.

Elliptische Kurven

Kurve und Gruppengesetz



Elliptische Kurven

Man geht davon aus, daß das effizienteste Verfahren für das DDH in einer Untergruppe G von großer Primzahlordnung der Punktgruppe $E(\mathbb{F}_q)$ einer elliptischen Kurve mit zufällig gewählten a, b das Pollard rho Verfahren ist.

Eine elliptische Kurve bietet somit maximal mögliche Sicherheit im Rahmen der gruppenbasierten Kryptographie.

Probleme/Fragen:

- Ordnung von $\#E(K)$ ($\Rightarrow$ Punkte zählen, Kurven konstruieren).
- Spezialfälle, in denen $E(K)$ unsicher ist.
- Optimierungen in Bandbreite und Rechnen (z.B. Punktkompression).

Im folgenden grober Überblick ...

Punkte zählen

Zum Rechnen und wegen Pohlig-Hellman möchten wir $E(K)$ kennen. Wissen nur $\#E(K) = q + 1 - t$, und $|t| \leq 2\sqrt{q}$.

Algorithmen zum Punktezählen:

- Schoof-Elkies-Atkin (SEA),
- Satoh, AGM (Mestre),
- Dwork-Spur Formel, Deformationen (Lauder-Wan),
- Monsky-Washnitzer Kohomologie (Kedlaya).

Diese Verfahren sind polynomiell in $\log(q)$ (SEA $O^{\sim}(\log(q)^4)$, die anderen $O^{\sim}(\log(q)^2)$ für $p = O(1)$).

Ist $\#E(K)$ berechnet, so kann man kleine Faktoren durch Probedivision herausdividieren und auf den Kofaktor dann einen Primzahltest (Miller-Rabin) anwenden.

Kurven konstruieren

Ein anderer Ansatz ist, elliptische Kurven so zu konstruieren, daß $\#E(K)$ a priori bekannt ist.

Subfield Kurven:

- Ist E über $\mathbb{F}_q$ definiert und $\#E(\mathbb{F}_q)$ bekannt, so kann man leicht $\#E(\mathbb{F}_{q^n})$ für alle n ausrechnen.

Komplexe Multiplikation:

- Mit weitergehender Mathematik kann man zu vorgegebener Punktanzahl direkt eine Kurve E konstruieren.

Etwas nachteilig ist hier - nur aus philosophischer Sicht -, daß die Kurven nicht zufällig gewählt werden. Dies könnte Möglichkeiten für spezielle Angriffe eröffnen (nichts wesentliches bekannt).

Unsichere Spezialfälle

Multiplikativer Transfer:

- auch Frey-Rück Reduktion (Menezes-Okamoto-Vanstone Angriff).
- Seien $\gcd\{\ell, q\} = 1$ und μ_ℓ die ℓ-ten Einheitswurzeln in $\mathbb{F}_{q^k}$ mit $\ell \mid (q^k - 1)$ und k minimal. $G = E(\mathbb{F}_q)[\ell]$ Untergruppe der Ordnung ℓ.
- Mit Hilfe der Tate-Paarung kann man einen Isomorphismus $E(\mathbb{F}_q)[\ell] \to \mu_\ell$ definieren, der in Zeit poly($k \log(q)$) berechnet werden kann.
- Man kann also ein DLP von $E(\mathbb{F}_q)[\ell]$ nach $\mathbb{F}_{q^k}^\times$ transferieren und dort subexponentiell lösen.
- Für von q unabhängiges, zufälliges ℓ ist k meist von der Größenordnung wie ℓ, somit der Angriff nicht durchführbar.
- Speziell für supersinguläre Kurven ($t = 0 \bmod p$) kann man jedoch immer $k \leq 6$ erreichen.
- Man sollte immer prüfen, ob zu q und ℓ der Exponent $k \geq 20$ ist.

Unsichere Spezialfälle

Additiver Transfer (für anomale Kurven, also $t = 1$ bzw. $\#E(\mathbb{F}_q) = q$):

- auch Rück oder SmartASS Angriff, additive Version der FR Reduktion. Hier $\ell = p$ und meistens $q = p$.
- Es gibt einen Isomorphismus $E(\mathbb{F}_q)[p] \to \mathbb{F}_p^+$, der in poly($\log(q)$) berechnet werden kann.
- Man kann also ein DLP aus $E(\mathbb{F}_q)[p]$ nach $\mathbb{F}_p^+$ transferieren und dort in Polynomzeit lösen. Durch Iterieren erhält man auch diskrete Logarithmen in $E(\mathbb{F}_q)$ (wie bei Pohlig-Hellman).

Anomale Kurven sind also besonders unsicher, während zum Beispiel supersinguläre Kurven nur eine reduzierte Sicherheit (subexponentiell) bieten und damit verwendbar bleiben. Siehe paarungsbasierte Kryptographie ...

Unsichere Spezialfälle

Weil Abstieg Techniken.

- Treffen im wesentlichen nur auf $q = 2^n$ und n mit kleinen Faktoren und/oder für spezielle elliptische Kurven zu.
- Konstruieren eine höher-geschlechtige Kurve, die aber über einem Teilkörper von $\mathbb{F}_q$ definiert ist.
- Das DLP kann in die Picardgruppe dieser Kurve transferiert werden. Hierin kann man einen Index-Calculus Angriff durchführen.
- Allgemeine Abhilfe: n prim wählen, zufällige Kurven verwenden.

Zusammenfassung Sicherheit

Kriterien für die Sicherheit einer elliptischen Kurve:

1. Punktanzahl $\#E(\mathbb{F}_q)$ enthält einen ausreichend großen Primfaktor ℓ, um gegen Pohlig-Hellman und Pollard rho zu schützen.
2. Es gilt nicht $\#E(\mathbb{F}_q) = q$, um vor Rück bzw. SmartASS Angriff zu schützen.
3. Das kleinste k mit $\ell \mid (q^k - 1)$ erfüllt $k \geq 20$, um gegen FR-Reduktion bzw. MOV-Angriff zu schützen.
4. Mit $q = p^n$ sollte n prim oder 1 sein, wegen Weil Abstieg Angriff.

Normalerweise wird nur $q = 2^n$ oder $q = p$ verwendet.

Für paarungsbasierte Kryptographie verzichtet man auf 3. (und 4.) und erhält zusätzliche Funktionalität bei reduziertem Sicherheits-Effizienz Verhältnis (z.B. supersingulär und $q = 3^n$, da dann $k = 6$ möglich).

Zusammenfassung Sicherheit

Will man „besonders sicher“ gehen, sollte man die elliptischen Kurven mit zufälligem a, b erzeugen, um keine „speziellen“ Eigenschaften zu erzeugen.

Wie kann man jemanden davon überzeugen, daß man eine Kurve zufällig erzeugt hat? Kurve beweisbar zufällig erzeugen ...

- Man nimmt z.B. a = SHA-1(Zahl-1) und b = SHA-1(Zahl-2), wobei die Zahlen zu variieren sind, bis die resultierende Kurve einen großen Primfaktor enthält.
- Man veröffentlicht dann die Zahlen-1,2 zur Berechnung von a, b.

In Standards werden die zu verwendenden Kurven häufig speziell vorgegeben (IPSec). Obiger „Sicherheitstest“ bei RSA nicht möglich.

Optimierungen

Man kann im Hinblick auf die Effizienz der erforderlichen Rechnungen in $E(\mathbb{F}_q)$ eine ganze Reihe von Optimierungen durchführen.

1. Gruppengesetz:
   - Verschiedene Koordinatensysteme für Punkte und optimierte Formel für das Gruppengesetz.

2. Punktvielfache:
   - Die Operation λP ist die teuerste Operation beim Verschlüsseln.
   - Man verwendet Varianten von Double-und-Add.
   - $-P$ kann besonders schnell ausgerechnet werden. Daher sollte man beim Double-und-Add auch Subtraktionen in Betracht ziehen.

Optimierungen

3. Punktkompression:
   - Für P ist y_P Nullstelle einer quadratischen Gleichung, die nur von E und x_P abhängt. Da es stets nur zwei solche Nullstellen gibt, genügt ein Bit zur Auswahl der Nullstelle.
   - Speicher- u. Kommunkationsbedarf für einen Punkt halbiert sich.

Gehen im folgenden exemplarisch etwas näher auf 2. und 3. ein.

Punktvielfache

Double-and-Add als Hornerschema:

- $m = \sum_{i=0}^{r} m_i 2^i$, $m_i \in \{0,1\}$ binäre Entwicklung.
- $[m]P = 2\big(\cdots\big(2\big(2[m_r]P + [m_{r-1}]P\big) + [m_{r-2}]P\big)\cdots + [m_1]P\big) + [m_0]P$.

Invertieren effizient, verwende daher allgemeiner $m_i \in \{0, \pm 1\}$.
$\Rightarrow$ Non-adjacent form (NAF), $m_i m_{i+1} = 0$ für alle i möglich.

Berechnung der NAF (im i-ten Schritt):

- Wenn $m \equiv 0 \bmod 2$, dann $m_i = 0$. Ansonsten wähle $m_i \in \{-1, 1\}$ mit $m - m_i \equiv 0 \bmod 4$ (dann $m_{i+1} = 0$).
- Setze $m = (m - m_i)/2$ und wiederhole für $i = i + 1$.

NAF höchstens ein Bit länger als binäre Entwicklung.
Durchschnittliche Dichte der binären Entwicklung $1/2$, der NAF $1/3$.
Binäre Entwicklung $(r/2)$ Adds + r Doubles, NAF $(r/3)A + rD$.

Punktvielfache

$E(\mathbb{F}_q)[\ell]$ zyklisch der Ordnung ℓ, $\phi \in \mathrm{End}(E)$ durch algebraische Formeln auf Koordinaten definiert, liefert Element von $\mathrm{End}(E(\mathbb{F}_q))$.

$\Rightarrow$ Es gibt $\lambda \in \mathbb{Z}$ mit $\phi(P) = [\lambda]P$ für alle $P \in E(\mathbb{F}_q)[\ell]$.

Schreibe $m = \sum_{i=0}^{r-1} m_i \lambda^i \bmod \ell$. Dann

$$\begin{aligned}[m]P &= [\sum_{i=0}^{r-1} m_i\lambda^i]P = \sum_{i=0}^{r-1} m_i[\lambda]^i(P) = \sum_{i=0}^{r-1} m_i\phi^i(P)\\ &= [m_0]P + [m_1]\phi(P) + \cdots + [m_{r-1}]\phi^{r-1}(P)\\ &= \phi\big(\cdots\big(\phi\big(\phi([m_{r-1}]P) + [m_{r-2}]P\big) + [m_{r-3}]P\big)\cdots + [m_1]P\big) + [m_0]P.\end{aligned}$$

Punktvielfache

Finde geeignete, effizient berechenbare ϕ:

- Frobeniusendomorphismus $(x_P, y_P) \mapsto (x_P^q, y_P^q)$ für Subfieldkurven über $\mathbb{F}_{q^n}$.
- Endomorphismen durch komplexe Multiplkation (siehe Konstruktion).

Dann zwei Hauptfälle:

- $r = 2$, $m_i = O(\sqrt{\ell}) \Rightarrow$ Benutze simultane Multiexponentiation.
- $r \approx \log_d(\ell)$, $|m_i| \lesssim d$, $d = O(1) \Rightarrow$ Benutze Horner Schema wie vorige Folie.

Liefert sogenannte ϕ-Entwicklungen. Weitere Methoden:

- ϕ-NAF, Sliding Window, ...

Punktkompression

Sei $P = (x_P, y_P)$ mit $y_P^2 = x_P^3 + ax_P + b$ in $\mathbb{F}_q$ (p ungerade).
Dann ist $Q = (x_P, -y_P)$ ebenfalls ein Punkt auf E.
Es gibt keine weiteren Punkte mit derselben x-Koordinate x_P.

Gegeben x_P, wie zwischen y_P und $-y_P$ mit Hilfe eines Bits unterscheiden? Verschiedene Möglichkeiten:

- Bit gibt an, ob y-Koordinate ein Quadrat oder nicht ist (Jacobi Symbol; für $q \equiv 3 \bmod 4$, da dann -1 kein Quadrat ist).
- Bit gibt an, ob lexikographisch größere oder kleinere y-Koordinate zu wählen ist.
- Praxis: Für $q = p$ wählt man als Bit das LSB(y_P) (least significant bit). Für $0 \le y_P < p$ wird $-y_P$ durch $-y_P + p$ repräsentiert. LSB(y_P) = 0 $\Leftrightarrow$ LSB($-y_P$) = 1, da p ungerade. Daher klappt's.

Speicher- und Übertragungsersparnis von ca. 50% (Patentgeschützt).

Parametervergleich

NIST Tabelle, Schlüsselgrößen bei ungefähr gleicher Sicherheit:

Block Chiffre Schlüsselgröße	Beispiel Block Chiffre	ECC Schlüsselgröße	RSA / $\mathbb{F}_q^\times$ Schlüsselgröße
80	SKIPJACK	163	1024
128	AES (klein)	283	3072
192	AES (mittel)	409	7680
256	AES (groß)	571	15360

ECC mit 517 praktikabel, RSA / $\mathbb{F}_q^\times$ mit 15360 nicht.