

Kryptographie und Komplexität

Einheit 5.3

Angriffe auf ElGamal Systeme



1. Berechnung diskreter Logarithmen
2. Andere Angriffe
3. Richtlinien für die Schlüsselauswahl

DAS PROBLEM DES DISKRETEN LOGARITHMUS

- **Allgemeine Formulierung**

- Gegeben multiplikative Gruppe $(G, \cdot)$, Element g der Ordnung n und $y \in \langle g \rangle$. Bestimme die eindeutige Zahl $x < n$ mit $y = g^x$

- **Über $\mathbb{Z}_p$ ähnlich schwer wie Faktorisierung**

- Viele Faktorisierungsalgorithmen lassen sich übertragen
- Kryptosysteme werden durch Verwendung des DL Problems alleine nicht sicherer als durch das Faktorisierungsproblem

- **Es gibt viele DL-Algorithmen**

- Aufzählungsverfahren, Shanks, Pollard ρ , Pohlig-Hellman, Index-Calculus, Zahlkörpersiebe
- Zunehmende Kompliziertheit senkt die Komplexität

- **Schwieriger über anderen Gruppen**

- Die besten DL-Algorithmen sind nur zahlentheoretisch verwendbar

AUFZÄHLUNG

- **Einfacher, leicht zu programmierender Ansatz**

- Lösungskandidaten $x = 1, 2, 3, \dots$ werden der Reihe nach überprüft
- Die Bestimmung von $x = \log_g y$ benötigt $x-1$ Multiplikationen
- Hochgradig ineffizient

Laufzeit $\mathcal{O}(n \cdot \|n\|^2) = \mathcal{O}(2^{\|n\|})$

- **Nur für Zahlen mit kleinen Logarithmen**

- Suche nach Logarithmen muß auf Schranke B begrenzt werden
- Schranke jenseits von 10^7 wenig sinnvoll

Anders als bei Faktorisierung sagt dies nichts über die Größe von n oder y aus

- **Keine Optimierungen wie bei Faktorisierung**

- Jede Zahl könnte der geeignete Logarithmus sein
- Man kann wenig über Struktur von x sagen (z.B. ungerade)
- Außer n gibt es keine obere Grenze für x

- **Beispiel für $(\mathbb{Z}_p, \cdot)$ mit $p = 944137$**

- $\log_2 3 = 467306$, $\log_2 4 = 2$, $\log_2 5 = 271379$, ...

SHANKS BABYSTEP-GIANTSTEP-ALGORITHMUS

- **Idee: Beschleunigung durch Zwischenspeicherung**

- Zerlege das gesuchte $x = \log_g y$ in $x = q \cdot m + r$
- Für diese Zerlegung gilt $g^{q \cdot m + r} = y$ also $(g^m)^q = y \cdot g^{-r}$
- Teste diese Gleichung für alle Paare (q, r) um x zu bestimmen
- Die Werte $y \cdot g^{-r}$ und $(g^m)^q$ können separat berechnet werden
- Für $m = \lfloor \sqrt{n} \rfloor$ müssen nur $2 \cdot \lfloor \sqrt{n} \rfloor$ Werte berechnet werden

- **Algorithmus**

- **Babystep**: Speichere für $r < m$ die Werte $(y \cdot g^{-r}, r)$ in einer Tabelle B
- **Giantstep**: Für $q = 0, 1, 2, \dots$ prüfe, ob $(g^m)^q$ in B vorkommt
- Im Erfolgsfall gebe $x = q \cdot m + r$ als Lösung für $\log_g y$ aus

- **Komplexität**

- Je eine Multiplikation für Berechnung eines $y \cdot g^{-r}$ bzw. eines $(g^m)^q$
- Bei Hashtabellen konstante Zeit für Test, ob $(g^m)^q$ in B vorkommt
- Maximal m Baby- und Giantsteps erforderlich
- **Laufzeit** $\mathcal{O}(\sqrt{n}) = \mathcal{O}(2^{\lceil n \rceil / 2})$ **Speicherbedarf** $\mathcal{O}(\sqrt{n})$

SHANKS ALGORITHMUS AM BEISPIEL

Bestimme $\log_3 5$ in $(\mathbb{Z}_{3137}, \cdot)$

- **Berechne Initialwerte**

- $m = \lfloor \sqrt{n} \rfloor = 56$, $g^{-1} = 3^{-1} \bmod 3137 = 1046$, $g^m = 893$

- **Erzeuge Babystep-Tabelle $B = [(y \cdot g^{-r}, r) \mid r < m]$**

- $B = [(5, 0); (2093, 1); (2789, 2); (3021, 3); (1007, 4); \dots (541, 55)]$

- Sortiert: $[(5, 0); (92, 17); (146, 35); (226, 26); (276, 16); (394, 43); (397, 12); (409, 41); (436, 10); (438, 34); (518, 48); (541, 55); (544, 39); (678, 25); (787, 8); (805, 32); (809, 6); (828, 15); (971, 30); (1007, 4); (1121, 27); (1177, 44); (1178, 13); (1182, 42); (1191, 11); (1227, 40); (1308, 9); (1314, 33); (1438, 45); (1525, 46); (1554, 47); (1589, 21); (1623, 54); (1630, 20); (1632, 38); (1732, 53); (1753, 19); (1759, 37); (2034, 24); (2059, 52); (2093, 1); (2122, 18); (2140, 36); (2264, 49); (2361, 7); (2415, 31); (2427, 5); (2465, 28); (2484, 14); (2621, 22); (2789, 2); (2846, 50); (2913, 29); (2965, 23); (3021, 3); (3040, 51)]$

- **Berechne Giantstep Werte $(g^m)^q$, $q = 0, 1, 2, \dots$**

- 1; 893; 651; 998; 306; 339; 1575; 1099; 2663; 213; 1989; 635; 2395; 2438; 56; 2953; 1949; 2559; 1451; 162; 364; 1941; 1689; 2517; 1589

- Treffer (1589, 21) für $q = 24$ liefert $x = 24 \cdot 56 + 21 = 1365$

POLLARD ρ ALGORITHMUS

• Modifikation der Pollard ρ Faktorisierung

- Suche $a \neq a', b \neq b' \in \mathbb{Z}_n$ mit $g^a \cdot y^b = g^{a'} \cdot y^{b'}$
- Ist $x = \log_g y$ in G so gilt $g^a \cdot g^{x \cdot b} = g^{a'} \cdot g^{x \cdot b'}$ also $g^{a+x \cdot b} = g^{a'+x \cdot b'}$
somit $a+x \cdot b \equiv a'+x \cdot b' \pmod{n}$ bzw. $(a-a') \equiv x \cdot (b'-b) \pmod{n}$
- Damit ist $x = (a-a')(b'-b)^{-1} \pmod{n}$ falls $\gcd(b'-b, n)=1$
- Bei Erzeugung einer Zufallsfolge mit $\mathcal{O}(\sqrt{n})$ Elementen findet man diese Kollision mit Wahrscheinlichkeit 50% (Geburtstagsparadox)

• Erzeuge und prüfe Zufallskandidaten

- Berechne Folge $(\beta_1, a_1, b_1), (\beta_2, a_2, b_2) \dots$ mit $\beta_i = g^{a_i} \cdot y^{b_i}$
und $(\beta_{i+1}, a_{i+1}, b_{i+1}) = f(\beta_i, a_i, b_i)$ für eine “Zufallsfunktion” f
- Gilt $\beta_i = \beta_j$ für ein $i < j$, dann auch $\beta_{i+1} = \beta_{j+1}$ also $\beta_{i+k} = \beta_{j+k}$
für alle k Folge der β_k läuft in eine Schleife, was aussieht wie ein ρ
- Hat die Schleife die Länge $l = j-i$, so gibt es ein $k \in \{i..j-1\}$,
das Vielfaches von l ist. Für dieses k gilt $\beta_k = \beta_{2k}$

• Einfaches Suchverfahren

- In Schritt k bestimme β_k, β_{2k} bis Werte gleich sind und berechne x

POLLARD ρ IM DETAIL

- **Wahl der “Zufallsfunktion” f**

- Teile Gruppe G in drei gleich große Teilgruppen G_1, G_2, G_3 auf
- Definiere $f(\beta, a, b) := \begin{cases} (\beta \cdot g, a +_n 1, b) & \text{falls } \beta \in G_1 \\ (\beta^2, 2 \cdot_n a, 2 \cdot_n b) & \text{falls } \beta \in G_2 \\ (\beta \cdot y, a, b +_n 1) & \text{falls } \beta \in G_3 \end{cases}$
- Per Konstruktion erhält f die Eigenschaft $\beta = g^a \cdot y^b$

- **Konstruktion der (β_i, a_i, b_i)**

- Definiere $(\beta_0, a_0, b_0) = (1, 0, 0)$ und $(\beta_{i+1}, a_{i+1}, b_{i+1}) = f(\beta_i, a_i, b_i)$
- Dann gilt $\beta_i = g^{a_i} \cdot y^{b_i}$ für alle i

- **Suchverfahren**

- In Schritt k bestimme (β_k, a_k, b_k) und $(\beta_{2k}, a_{2k}, b_{2k})$ bis $\beta_k = \beta_{2k}$
- Ist $\gcd(b_{2k} - b_k, n) = 1$ so berechne $x = (a_k -_n a_{2k})(b_{2k} -_n b_k)^{-1} \bmod n$
- Ansonsten breche ohne Erfolg ab

Laufzeit $\mathcal{O}(\sqrt{n}) = \mathcal{O}(2^{\lceil n \rceil / 2})$

konstanter Speicherbedarf

POLLARD ρ : ABLAUFBEISPIEL

• Trace der Berechnung von $\log_{89} 618$ in $\mathbb{Z}_{809}$

- Ordnung von $g = 89$ in $(\mathbb{Z}_{809}^*, \cdot)$ ist $n = 101$
- Wähle $G_{1/2/3} = \{g \leq 808 \mid g \equiv 1/0/2 \pmod{3}\}$

Schleife 1.	$\beta = 618$	$a = 0$	$b = 1$	$\beta' = 76$	$a' = 0$	$b' = 2$
Schleife 2.	$\beta = 76$	$a = 0$	$b = 2$	$\beta' = 113$	$a' = 0$	$b' = 4$
Schleife 3.	$\beta = 46$	$a = 0$	$b = 3$	$\beta' = 488$	$a' = 1$	$b' = 5$
Schleife 4.	$\beta = 113$	$a = 0$	$b = 4$	$\beta' = 605$	$a' = 4$	$b' = 10$
Schleife 5.	$\beta = 349$	$a = 1$	$b = 4$	$\beta' = 422$	$a' = 5$	$b' = 11$
Schleife 6.	$\beta = 488$	$a = 1$	$b = 5$	$\beta' = 683$	$a' = 7$	$b' = 11$
Schleife 7.	$\beta = 555$	$a = 2$	$b = 5$	$\beta' = 451$	$a' = 8$	$b' = 12$
Schleife 8.	$\beta = 605$	$a = 4$	$b = 10$	$\beta' = 344$	$a' = 9$	$b' = 13$
Schleife 9.	$\beta = 451$	$a = 5$	$b = 10$	$\beta' = 112$	$a' = 11$	$b' = 13$
Schleife 10.	$\beta = 422$	$a = 5$	$b = 11$	$\beta' = 422$	$a' = 11$	$b' = 15$

- Resultat: $x = (a -_n a')(b' -_n b)^{-1} = 95 \cdot_n 4^{-1} = 95 \cdot_n 76 = 49$

• Erweiterung des Algorithmus

- Originalverfahren bricht ab, wenn $\gcd(b_{2k} - b_k, n) \neq 1$
- Kongruenz $(a_k - a_{2k}) \equiv x \cdot (b_{2k} - b_k) \pmod{n}$ hat d mögliche Lösungen x_i wenn $d = \gcd(b_{2k} - b_k, n) > 1$
- Für kleine d kann man für alle x_i prüfen, ob $g^{x_i} = y$ ist

DER POHLIG-HELLMAN ALGORITHMUS (I)

Wenn die Faktorisierung von $n = |G|$ bekannt ist

- **Bestimme Logarithmen in Untergruppen von G**

- Sei $n = |G| = \prod_{p|n} p^{e_p}$ und $x = \log_g y$
- Für jedes p setze $n_p = n/p^{e_p}$, $g_p = g^{n_p}$, $y_p = y^{n_p}$
- Dann ist p^{e_p} die Ordnung von g_p und es gilt $g_p^x = g^{n_p \cdot x} = y^{n_p} = y_p$
also existiert der diskrete Logarithmus $x_p = \log_{g_p} y_p$
- Berechne alle x_p mit dem Shanks- oder dem Pollard ρ Algorithmus

- **Berechne $x = \log_g y$ aus den Komponenten**

Satz: **Gilt $x \equiv \log_{g_p} y_p \pmod{p^{e_p}}$ für alle p , dann ist $x = \log_g y$**

Beweis: Es gilt $(g^{-x} \cdot y)^{n_p} = (g^{n_p})^{-x} \cdot y^{n_p} = g_p^{-x_p} \cdot y_p = 1$

für alle Primteiler p von n . Also ist die Ordnung von $g^{-x} \cdot y$ (in G) ein Teiler von allen n_p . Da 1 der größte gemeinsame Teiler aller n_p ist, gilt $g^{-x} \cdot y \equiv 1 \pmod{n}$ also $g^x \equiv y \pmod{n}$

- Berechne $x = \log_g y$ aus allen x_p mit dem chinesischem Restsatz

Gesamtlaufzeit $\mathcal{O}(\sum_{p|n} \sqrt{p^{e_p}})$

konstanter Speicherbedarf

DER POHLIG-HELLMAN ALGORITHMUS (II)

- **Vereinfache Darstellung von $x_p = \log_{g_p} y_p$**

Der Übersichtlichkeit halber wird der Index p im folgenden fallen gelassen

- Wegen $x < p^e$ gilt $x = \sum_{i < e} x_i p^i$ für bestimmte Koeffizienten x_i
- Folglich ist $y^{p^{e-1}} = g^{x \cdot p^{e-1}} = (g^{p^{e-1}})^{x_0} \cdot (g^{p^e})^{\sum_{0 < i < e} x_i p^{i-1}} = (g^{p^{e-1}})^{x_0}$ da p^e die Ordnung von g ist
- Da $g_* = g^{p^{e-1}}$ die Ordnung p hat, kann $x_0 = \log_{g_*} y^{p^{e-1}}$ in einer Gruppe der Ordnung p in der **Laufzeit** $\mathcal{O}(\sqrt{p})$ berechnet werden

- **Berechne Koeffizienten von x_p**

- Berechne x_0 mit dem Shanks- oder dem Pollard ρ Algorithmus
- Sind $x_0, \dots, x_{k-1}$ bereits bestimmt, so gilt
$$\begin{aligned} y_k &= (y \cdot g^{-\sum_{i < k} x_i p^i})^{p^{e-k-1}} = (g^{\sum_{k \leq i < e} x_i p^i})^{p^{e-k-1}} = (g^{\sum_{i < e-k} x_i p^i})^{p^{e-1}} \\ &= (g^{p^{e-1}})^{x_k} \cdot (g^{p^e})^{\sum_{0 < i < e-k} x_i p^{i-1}} = (g^{p^{e-1}})^{x_k}, \text{ also } x_k = \log_{g_*} y_k \end{aligned}$$
Berechne x_k mit dem Shanks- oder dem Pollard ρ Algorithmus

- **Gesamtkomplexität**

$$\mathcal{O}(\sum_{p|n} e_p \cdot \sqrt{p})$$

- Berechnung der Koeffizienten von x_p benötigt **Laufzeit** $\mathcal{O}(e_p \cdot \sqrt{p})$
- Aufwand für chinesischen Restsatz vernachlässigbar

POHLIG-HELLMAN ALGORITHMUS FÜR $x = \log_5 3$ IN $\mathbb{Z}_{2017}$

• Reduktion auf Primzahlpotenzen

- Die Gruppenordnung ist $n = 2016 = 2^5 \cdot 3^2 \cdot 7$
- Zu berechnen sind $x_2 = \log_{5^{63}} 3^{63} = \log_{500} 913$,
 $x_3 = \log_{5^{224}} 3^{224} = \log_{576} 1933$ und $x_7 = \log_{5^{288}} 3^{288} = \log_{1879} 1879$

• Berechnung von $x_2 = \sum_{i=0}^4 x_{2,i} 2^i$

- Es ist $g_* = 500^{16} \bmod 2017 = 2016$ und $913^{16} \bmod 2017 = 1$
- Damit ist $x_{2,0} = \log_{2016} 1 = 0$
- Es ist $y_{2,1} = (913 \cdot 500^0)^8 = 913^8 = 2016$ also $x_{2,1} = \log_{2016} 2016 = 1$
- Es ist $y_{2,2} = 1579^4 = 2016$ also $x_{2,2} = \log_{2016} 2016 = 1$
- Es ist $y_{2,3} = 1^2 = 1$ also $x_{2,3} = \log_{2016} 1 = 0$
- Es ist $y_{2,4} = 1$ also $x_{2,4} = \log_{2016} 1 = 0$
- Insgesamt ergibt sich $x_2 = 6$

• Berechnung von x

- Analog ergibt sich $x_3 = 4$ und $x_7 = 1$
- Lösung der simultanen Kongruenzen $x \equiv 6 \bmod 32$, $x \equiv 4 \bmod 9$ und $x \equiv 1 \bmod 7$ ergibt $x = 1030$

DIE INDEX-CALCULUS METHODE

Verwandt mit Siebverfahren für Faktorisierung

• Einfache Grundidee

- Wähle eine Faktorbasis $\mathcal{B} = \{p \text{ prim} \mid p \leq b\}$ für eine Zahl b
- Bestimme diskrete Logarithmen $x_p = \log_g p$ für alle Elemente von $\mathcal{B}$
- Suche Exponenten $a < n$ für die $y \cdot g^a$ b -glatt ist (d.h. $y \cdot g^a = \prod_{p \in \mathcal{B}} p^{e_p}$)
- Dann ist $y \cdot g^a \equiv \prod_{p \in \mathcal{B}} (g^{x_p})^{e_p} \equiv g^{\sum_{p \in \mathcal{B}} x_p \cdot e_p} \pmod{n}$
also $x = \log_g y \equiv \sum_{p \in \mathcal{B}} x_p \cdot e_p - a \pmod{\varphi(n)}$

• Bestimmung der $x_p = \log_g p$ für $p \in \mathcal{B}$

- Wähle zufällige $1 \leq z_i < n$ für die $g^{z_i} \pmod{n}$ zerlegbar in $\prod_{p \in \mathcal{B}} p^{e_{i,p}}$
- Dann ist $g^{z_i} \equiv \prod_{p \in \mathcal{B}} p^{e_{i,p}} \equiv \prod_{p \in \mathcal{B}} g^{x_p \cdot e_{i,p}} \pmod{n}$
also $z_i \equiv \sum_{p \in \mathcal{B}} x_p \cdot e_{i,p} \pmod{\varphi(n)}$
- Hat man $|\mathcal{B}|$ derartige **Relationen** $(e_{i,p})_{p \in \mathcal{B}}$ gefunden, so kann man die x_p mit Hilfe eines modifizierten Gaußalgorithmus bestimmen

• Gesamtkomplexität $\mathcal{O}(2^{(1+o(1)) \cdot \|n\|^{1/2} \cdot \log \|n\|^{1/2}})$

- Analyse ähnlich wie bei quadratischen Sieben für Faktorisierung

DIE INDEX-CALCULUS METHODE AM BEISPIEL

Logarithmen der Faktorbasis für $g = 2$ in $\mathbb{Z}_{2027}$

- **Relationen der Faktorbasis $\{2, 3, 5, 7, 11\}$**

- Zufällige Erzeugung von Zahlen z_i liefert

- $2^{1593} \bmod 2027 = 33 = 3 \cdot 11$
- $2^{983} \bmod 2027 = 385 = 5 \cdot 7 \cdot 11$
- $2^{1318} \bmod 2027 = 1408 = 2^7 \cdot 11$
- $2^{293} \bmod 2027 = 63 = 3^2 \cdot 7$
- $2^{1918} \bmod 2027 = 1600 = 2^6 \cdot 5^2$

- **Bestimme $x_p = \log_2 p$ für $p \in \{2, 3, 5, 7, 11\}$**

- $x_3 + x_{11} \equiv 1593 \bmod 2026$
- $x_5 + x_7 + x_{11} \equiv 983 \bmod 2026$
- $7x_2 + x_{11} \equiv 1318 \bmod 2026$
- $2x_3 + x_7 \equiv 293 \bmod 2026$
- $6x_2 + 2x_5 \equiv 1918 \bmod 2026$

- Löse Kongruenzen mit modifiziertem Gaußalgorithmus

DIE INDEX-CALCULUS METHODE AM BEISPIEL (II)

- **Modifizierter Gaußalgorithmus**

- Löse Kongruenzen modulo 2 und 1013 (die Primteiler von 2026)
- Berechne x_p aus Einzellösungen mit Chinesischem Restsatz

- **Löse Kongruenzen modulo 2**

- $x_3 + x_{11} \equiv 1593 \pmod{2}$

$$x_5 + x_7 + x_{11} \equiv 983 \pmod{2}$$

- $x_2 + x_{11} \equiv 1318 \pmod{2}$

$$x_7 \equiv 293 \pmod{2}$$

- Wegen $g = 2$ ist $x_2 = 1$
- Ansonsten ergibt sich $x_5 \equiv x_7 \equiv x_{11} \equiv 1 \pmod{2}$ und $x_3 \equiv 0 \pmod{2}$

- **Löse Kongruenzen modulo 1013**

- $x_3 + x_{11} \equiv 580 \pmod{1013}$

$$x_5 + x_7 + x_{11} \equiv 983 \pmod{1013}$$

- $x_{11} \equiv 298 \pmod{1013}$

$$2x_3 + x_7 \equiv 293 \pmod{1013}$$

- $2x_5 \equiv 899 \pmod{1013}$

- Ergibt $x_{11} \equiv 298$, $x_5 \equiv 956$, $x_7 \equiv 742$, $x_3 \equiv 282 \pmod{1013}$

- **Lösung der simultanen Kongruenz**

- $x_2 = 1$, $x_3 = 282$, $x_5 = 1969$, $x_7 = 1755$, $x_{11} = 1311$

DAS INDEX-CALCULUS VERFAHREN AM BEISPIEL (III)

Berechnung von $\log_2 13$ in $\mathbb{Z}_{2027}$

- **Suche ein 11-glattes $13 \cdot 2^a$ für ein $a \in \{1, \dots, 2026\}$**
 - Zufallssuche ergibt $a = 1397$ und $13 \cdot 2^{1397} \equiv 110 \equiv 2 \cdot 5 \cdot 11 \pmod{2027}$
- **Berechne $x \equiv \sum_{p \in \mathcal{B}} x_p \cdot e_p - a \pmod{\varphi(n)}$**
 - Liefert $x \equiv x_2^1 + x_5^1 + x_{11}^1 - a \equiv 1 + 1969 + 1311 - 1397 \pmod{2026}$
 - Ergebnis ist $x = \log_2 13 = 1884$

Verallgemeinerung auf andere Gruppen

- Methode stützt sich auf zahlentheoretische Eigenschaften
- Andere Gruppen müssen Aufbau einer Faktorbasis und Konstruktion von Relationen aus Exponentenvektoren unterstützen
- Relationenfindung in $\mathbb{Z}_p^*$ baut auf Primfaktorzerlegung in $\mathbb{Z}$ auf
- **Verfahren zur Relationenfindung in anderen Gruppen nicht bekannt**

DL ALGORITHMEN IM RÜCKBLICK

- **Aufzählungsverfahren**

- Standardverfahren, gut für kleine Gruppenordnungen $\mathcal{O}(2^{\|n\|})$

- **Algorithmus Algorithmus von Shanks**

- Zerlegung der Suche in Baby- und Giantsteps $\text{Zeit/Platz } \mathcal{O}(2^{\|n\|/2})$

- **Pollard ρ**

- Systematische Suche nach Kollisionen $\mathcal{O}(2^{\|n\|/2})$

- **Pohlig-Hellman Verfahren**

- Reduktion auf Primfaktoren von n $\mathcal{O}(\sum_{p|n} e_p \cdot \sqrt{p})$

- **Index-Calculus Methode**

- Aufbau von Logarithmen einer Faktorbasis $\mathcal{O}(2^{(1+o(1)) \cdot \|n\|^{1/2} \cdot \log \|n\|^{1/2}})$
- **Zahlkörpersieb** als effizienteste Variante $\mathcal{O}(2^{1.92 \cdot \|n\|^{1/3} \cdot \log \|n\|^{2/3}})$
- Verallgemeinert sehr schlecht auf andere Gruppen