

---

# Fortgeschrittene Techniken der Kryptographie

Jürgen Fuß

Episode 6: Verfahren zur Berechnung von diskreten Logarithmen

HAGENBERG | LINZ | STEYR | WELS



UNIVERSITY  
OF APPLIED SCIENCES  
UPPER AUSTRIA

## Baby Step Giant Step

# Diskrete Logarithmen – Bruteforce

Es sei  $p := 389$ . Dann ist  $g := 5$  ein Element der Ordnung  $\omega = 97$  in  $\mathbb{Z}_p^*$ . 

$g^0 = 1$	$g^1 = 5$	$g^2 = 25$	$g^3 = 125$	$g^4 = 236$	...	$g^9 = 345$
$g^{10} = 169$	$g^{11} = 67$	$g^{12} = 335$	$g^{13} = 119$	$g^{14} = 206$	...	$g^{19} = 344$
$g^{20} = 164$	$g^{21} = 42$	$g^{22} = 210$	$g^{23} = 272$	$g^{24} = 193$	...	$g^{29} = 175$
$g^{30} = 97$	$g^{31} = 96$	$g^{32} = 91$	$g^{33} = 66$	$g^{34} = 330$		:
:					..	:
$g^{90} = 79$	...	...	...	...	...	$g^{99} = 25$

Um den diskreten Logarithmus zu finden, müssen im Worst Case  $\omega$  Potenzen von  $g$  berechnet und verglichen werden. Alternativ ließe sich auch ein **Lookup Table** mit den  $\omega$  verschiedenen Potenzen erstellen.

# Diskrete Logarithmen – Bruteforce mit weniger Speichern

Es sei  $p := 389$ . Dann ist  $g := 5$  ein Element der Ordnung  $\omega = 97$  in  $\mathbb{Z}_p^*$ .

Baby Steps

$g^0 = 1$	$g^1 = 5$	$g^2 = 25$	$g^3 = 125$	$g^4 = 236$	$\dots$	$g^9 = 345$
$g^{10} = 169$	$g^{11} = 67$	$g^{12} = 335$	$g^{13} = 119$	$g^{14} = 206$	$\dots$	$g^{19} = 344$
$g^{20} = 164$	$g^{21} = 42$	$g^{22} = 210$	$g^{23} = 272$	$g^{24} = 193$	$\dots$	$g^{29} = 175$
$g^{30} = 97$	$g^{31} = 96$	$g^{32} = 91$	$g^{33} = 66$	$g^{34} = 330$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$g^{90} = 79$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$g^{99} = 25$

Speichert man nur die Potenzen aus der ersten Zeile und der ersten Spalte, so sieht man

$$330 = 97 \cdot 236 = g^{30} \cdot g^4 = g^{34}$$

Das gesuchte Element ist das Produkt aus einem Element der ersten Zeile ( $g^4$ ) und einem Element aus der ersten Spalte ( $g^{30}$ ).

# Baby-Step-Giant-Step

$$A = g^{\alpha} \quad \text{Finde } \alpha.$$

330      5      ? 34

A =  $g^{\alpha}$ .

Wähle  $N := \lceil \sqrt{\omega} \rceil + 1$ . Dann ist  $N^2 > \omega$ . Seien  $q$  und  $r$  nun Quotient und Rest der Division  $\alpha/N$ , also  $\underline{\underline{\alpha}} = qN + r$ . Dann ist  $r < N$  und  $q < N$ .

Nun sollen  $q$  und  $r$  gefunden werden,  $q$  entspricht der gesuchten Zeile und  $r$  der Spalte in der Tabelle aller Potenzen von  $g$ .

Man beachte, dass nun

$$\begin{aligned} A &= g^{\alpha} = g^{qN+r} \\ A &= (g^N)^q \cdot g^r \\ A(g^{-N})^q &= g^r. \end{aligned}$$
$$\begin{aligned} g^{qN+r} &= g^{qN} \circ g^r \\ &= (g^N)^q \circ g^r \end{aligned}$$

# Baby Steps

$$A(g^{-N})^q = g^r$$

Zunächst berechnet man alle Potenzen auf der rechten Seite:

$$\mathcal{B} := \{(g^r, r) \mid r \in \{0, 1, \dots, N-1\}\}.$$

Diese Potenzen heißen **Baby Steps**, denn hier geht es in kleinen Schritten immer um den Faktor  $g$  weiter. Insbesondere lässt sich  $\underline{g^{r+1}} = \underline{g^r} \circ g$  mit nur einer Multiplikation aus  $g^r$  berechnen.

Die Baby Steps sind genau die Werte in der ersten Zeile der Tabelle aller Potenzen.

# Giant Steps

$$\frac{Af^q}{Af} = g^r$$

$$A(g^{-N})^q = g^r$$

Sodann prüft man für  $q \in \{0, 1, \dots, N - 1\}$ , ob für ein  $r \in \{0, 1, \dots, N - 1\}$  das Paar  $(A(g^{-N})^q, r)$  in  $\mathcal{B}$  vorkommt. Am besten berechnet man zunächst  $f := g^{-N}$  und dann  $Af^q$ . Auch hier erhält man  $Af^{q+1}$  aus  $Af^q \cdot f$  mit nur einer Multiplikation mit  $f$ . Für wachsendes  $q$  steigt hier das Ergebnis um den Faktor  $f = g^{-N}$ , darum spricht man hier von **Giant Steps** (ein Giant Step entspricht  $N$  Baby Steps).

Die Giant Steps sind genau die Kehrwerte der ersten Spalte in der Tabelle aller Potenzen. Ist ein passendes Paar  $(r, q)$  gefunden, so gilt

$$\alpha = Nq + r.$$

# Baby Step Giant Step: Aufwand

Für den Baby-Step-Giant-Step-Algorithmus braucht man  $N$  Multiplikationen, um die Baby Steps  $\mathcal{B}$  zu erstellen, und im Mittel  $\underline{N/2}$  Multiplikationen, bis der Index  $q$  gefunden ist, in Summe also im Mittel

$$1,5 \cdot N \approx 1,5 \cdot \underline{\sqrt{\omega}} \text{ Multiplikationen.}$$

Die Brute-Force-Suche hat im Vergleich dazu einen mittleren Aufwand von

$$\underline{\omega/2} \text{ Multiplikationen.}$$

Damit der Aufwand für BSGS bei mindestens  $2^n$  liegt muss  $\omega$  wenigstens  $2n$  Bit lang sein.

## Pohlig-Hellman

# Nichtprime Ordnungen

$$A = g^\alpha, \quad \underline{\text{ord}(g) = \omega} = \underline{\underline{m \cdot n}} \quad (\underline{\text{ggT}(m, n) = 1})$$

# Nichtprime Ordnungen

$$A = g^\alpha, \quad \text{ord}(g) = \underline{\omega} = \underline{m \cdot n} \quad (\text{ggT}(m, n) = 1)$$

$$\text{ord}(g^m) = \frac{\omega}{\text{ggT}(m, \omega)} = \frac{m \cancel{n}}{\cancel{m}} = \underline{n} < \omega \text{ und}$$

$$\text{ord}(g^n) = \frac{\omega}{\text{ggT}(n, \omega)} = \frac{n \cancel{m}}{\cancel{n}} = \underline{m} < \omega.$$

# Nichtprime Ordnungen

~~$A = g^\alpha$~~   $\xrightarrow{\text{mod } \omega}$   $\text{ord}(g) = \omega = m \cdot n$   $(\text{ggT}(m, n) = 1)$

$$\omega = 74 = \underline{\underline{2 \cdot 37}}$$

$$\text{ord}(g^m) = \frac{\omega}{\text{ggT}(m, \omega)} = \frac{m \cancel{n}}{\cancel{m}} = \underline{\underline{n}} < \omega \text{ und}$$

$$\text{ord}(g^n) = \frac{\omega}{\text{ggT}(n, \omega)} = \frac{m \cancel{n}}{\cancel{n}} = m < \omega.$$

$\alpha \text{ mod } m \cdot n \xrightarrow{\text{ggT}} \left\{ \begin{array}{l} \alpha \text{ mod } n \\ \alpha \text{ mod } m \end{array} \right.$

$$\begin{aligned} (g^m)^\alpha &= (\underbrace{g^\alpha}_N)^m = \underbrace{A^m}_M \\ (g^n)^\alpha &= (\underbrace{g^\alpha}_N)^n = \underbrace{A^n}_M \end{aligned}$$

$$\begin{array}{c} \xrightarrow{\text{mod } n} \\ \boxed{M^\alpha = A^m} \\ \xleftarrow{32} \\ \boxed{N^\alpha = A^n} \\ \xleftarrow{5^\alpha = 54} \\ \xleftarrow{\text{mod } m} \end{array}$$

# Chinesischer Restsatz to the Rescue

Der Exponent  $\alpha$  in der ersten Gleichung

$$(g^m)^\alpha = A^m$$

lässt sich einfacher finden, denn  $\text{ord}(g^m)$  ist kleiner als  $\text{ord}(g)$ , dasselbe gilt für die zweite Gleichung.

Allerdings können wir  $\alpha$  nur modulo der jeweiligen Ordnung bestimmen. Aus der linken Gleichung erhalten wir  $\alpha \bmod n$ , aus der rechten  $\alpha \bmod m$ .

Da aber  $\text{ggT}(m, n) = 1$  ist, lässt sich daraus mit dem chinesischen Restsatz  $\alpha \bmod m \cdot n$ , also  $\alpha \bmod \omega$  berechnen.

## Beispiel (1)

$$g^{\omega} \equiv 1$$

$$g^{35} \equiv 1$$

Das Element  $\underline{g = 10}$  in  $\mathbb{Z}_{71}^*$  hat die Ordnung  $\underline{\omega = 35}$ . In diesem Fall können wir  $\underline{m = 5}$  und  $\underline{n = 7}$  wählen. Wir suchen den diskreten Logarithmus  $\alpha$  von  $\underline{A = 38}$  zur Basis  $\underline{g}$ . Dazu berechnen wir

$$\begin{cases} \underline{g^m} = \underline{10^5} \mod 71 = \underline{32}, \\ \underline{A^m} = \underline{38^5} \mod 71 = \underline{20}, \\ \underline{g^n} = \underline{10^7} \mod 71 = \underline{5} \quad \text{und} \\ \underline{A^n} = \underline{38^7} \mod 71 = \underline{54}. \end{cases}$$

## Beispiel (2)

Anstatt der ursprünglichen Gleichung

$$10^\alpha = 38 \pmod{71} \quad (g^\alpha = A)$$

erhalten wir die neuen Gleichungen

$$\begin{array}{l} \overbrace{32^\alpha = 20 \pmod{71}}^{\alpha=6} \quad ((g^m)^\alpha = A^m) \text{ und} \\ 5^\alpha = 54 \pmod{71} \end{array}$$

$$\begin{array}{l} \overbrace{5^\alpha = 54 \pmod{71}}^{\alpha=3} \quad ((g^n)^\alpha = A^n). \end{array}$$

$$\begin{aligned} \alpha &\equiv 6 \pmod{7} \\ \alpha &\equiv 3 \pmod{5} \\ &\downarrow \text{CRS} \\ \alpha &\equiv 13 \pmod{35} \end{aligned}$$

Die Ordnung von 32 ist 7, die Ordnung von 5 ist 5. Aus der ersten Gleichung ergibt sich (mit Baby-Step-Giant-Step oder einfachem Durchprobieren)  $\alpha = 6 \pmod{7}$ . Aus der zweiten Gleichung ergibt sich  $\alpha = 3 \pmod{5}$ . Mit dem chinesischen Restsatz erhalten wir daraus  $\alpha = 13 \pmod{35}$ .

Der größte Aufwand liegt nach wie vor in der Bestimmung der diskreten Logarithmen, nun aber für Elemente mit kleineren Ordnungen  $m$  und  $n$ . Im Vergleich dazu ist der Aufwand für den chinesischen Restsatz vernachlässigbar.

Der Aufwand für den Pohlig-Hellman-Algorithmus hängt ausschließlich vom größten Primfaktor von  $\omega$  ab. In den allermeisten Fällen wird daher gleich ein Element mit großer primer Ordnung  $\omega$  gewählt.

## Der Index-Calculus-Algorithmus

$$\underline{A} \leftarrow g^{\alpha} \bmod p$$

# Diskrete Logarithmen in $\mathbb{Z}_p^*$

Dieses Verfahren zur Berechnung funktioniert **nur für die Gruppen  $\mathbb{Z}_p^*$** , ist aber wesentlich effizienter als die bisher betrachteten.

Es seien  $p \in \mathbb{P}$  und  $g \in \mathbb{Z}_p^*$  ein Element mit primer Ordnung  $\omega$ . Wir möchten diskrete Logarithmen zur Basis  $g$  modulo  $p$  berechnen.

$$A = \underline{g}^\alpha \pmod{p}$$

# Faktorbasis

$$A = g^\alpha \bmod p$$

$$\begin{aligned} A &= 3^2 \cdot 19^6 \\ &= (g^{\alpha_3})^2 \cdot (g^{\alpha_{19}})^6, \quad \text{circled } 2\alpha_3 + 4\alpha_{19} \end{aligned}$$

Zunächst wählen wir eine Menge  $\mathcal{F}$  von Primzahlen, die sogenannte **Faktorbasis**.

Diese Primzahlen müssen Potenzen von  $g$  sein. Dies lässt sich leicht überprüfen, indem man prüft, ob die Ordnung gleich  $\omega$  ist.

$$q \quad \text{ord}(q) = \omega?$$

$$\begin{aligned} \text{ord}(g) &= \omega \\ \text{ord}(g^\omega) &= \frac{\omega}{\text{ggT}(\omega, \varphi)} = \omega \end{aligned}$$

Eine Zahl  $z$  nennen wir  **$\mathcal{F}$ -glatt**, falls alle Primfaktoren von  $z$  in  $\mathcal{F}$  liegen.

[ Wir bestimmen zunächst die diskreten Logarithmen aller Primzahlen in der Faktorbasis, also für jedes  $q \in \mathcal{F}$  ein  $\alpha_q$  mit  $g^{\alpha_q} \bmod p = q$ .

$$q \quad g^{\alpha_q} \bmod p \quad \text{DL}$$

# Diskrete Logarithmen der Faktorbasiselemente (1)

Dazu berechnet man  $g^\zeta$  für viele zufällig gewählte Zahlen  $\zeta$  und merkt sich diejenigen, für die  $g^\zeta \bmod p$   $\mathcal{F}$ -glatt ist. Hat man viele solche Zahlen gefunden, kann man die diskreten Logarithmen der Faktorbasiselemente berechnen.

Dieses Beispiel illustriert die Methode für  $p = 2027$ ,  $g = 1000$  und die Faktorbasis  $\mathcal{F} = \{3, 13, 17, 19, 31\}$ . Die Ordnung von  $g$  ist  $\omega = 1013$ . Mit etwas Probieren finden wir z. B.

zeta $\rightarrow \zeta$	$g^\zeta \bmod p$	in $\mathcal{F}$
738	171	$3^2 \cdot 19$
474	1581	$3 \cdot 17 \cdot 31$
666	867	$3 \cdot 17^2$
336	1521	$3^2 \cdot 13^2$
168	39	$3 \cdot 13$
265	1989	$3^2 \cdot 13 \cdot 17$

$171 : 3 = 57 : 3 = 19$

## Diskrete Logarithmen der Faktorbasiselemente (2)

Wir bestimmen nun Zahlen  $\alpha_3, \alpha_{13}, \alpha_{17}, \alpha_{19}, \alpha_{31}$ , so dass

$$\underbrace{3 = g^{\alpha_3}}, \underbrace{13 = g^{\alpha_{13}}}, \underbrace{17 = g^{\alpha_{17}}}, \underbrace{19 = g^{\alpha_{19}}}, \underbrace{31 = g^{\alpha_{31}}} \pmod{2027}.$$

Die Primfaktorzerlegungen von oben ergeben das Gleichungssystem (modulo 2027)

$$g^{738} = 3^2 \cdot 19 = (g^{\alpha_3})^2 \cdot (g^{\alpha_{19}}) = g^{2\alpha_3 + \alpha_{19}}$$

$$g^{474} = 3 \cdot 17 \cdot 31 = (g^{\alpha_3}) \cdot (g^{\alpha_{17}}) \cdot (g^{\alpha_{31}}) = g^{\alpha_3 + \alpha_{17} + \alpha_{31}}$$

$$g^{666} = 3 \cdot 17^2 = (g^{\alpha_3}) \cdot (g^{\alpha_{17}})^2 = g^{\alpha_3 + 2\alpha_{17}}$$

$$g^{336} = 3^2 \cdot 13^2 = (g^{\alpha_3})^2 \cdot (g^{\alpha_{13}})^2 = g^{2\alpha_3 + 2\alpha_{13}}$$

$$g^{168} = 3 \cdot 13 = (g^{\alpha_3}) \cdot (g^{\alpha_{13}}) = g^{\alpha_3 + \alpha_{13}}$$

$$g^{265} = 3^2 \cdot 13 \cdot 17 = (g^{\alpha_3})^2 \cdot (g^{\alpha_{13}}) \cdot (g^{\alpha_{17}}) = g^{\alpha_3 + \alpha_{13} + \alpha_{17}}$$

$$\begin{aligned} 738 &= 2\alpha_3 + \alpha_{19} \pmod{w} \\ 474 &= \alpha_3 + \alpha_{17} + \alpha_{31} \end{aligned}$$

# Diskrete Logarithmen der Faktorbasiselemente (3)

Wir betrachten die Exponenten und es ergibt sich ein lineares Gleichungssystem (modulo  $\omega = 1013$ ):

$$738 = 2\alpha_3 + \alpha_{19}$$

$$474 = \alpha_3 + \alpha_{17} + \alpha_{31}$$

$$666 = \alpha_3 + 2\alpha_{17}$$

$$336 = 2\alpha_3 + 2\alpha_{13}$$

$$168 = \alpha_3 + \alpha_{13}$$

$$265 = \alpha_3 + \alpha_{13} + \alpha_{17}$$

```
> from galois import GF  
> k = GF(1013)  
> m = k( [[2,0,0,1,0,738],  
           [1,0,1,0,1,474],  
           [1,0,2,0,0,666],  
           [2,2,0,0,0,336],  
           [1,1,0,0,0,168],  
           [2,1,1,0,0,265]])
```

# Diskrete Logarithmen der Faktorbasiselemente (4)

```
> r = m.row_reduce(); r
GF([[ 1, 0, 0, 0, 0, 541],
 [ 0, 1, 0, 0, 0, 640],
 [ 0, 0, 1, 0, 0, 569],
 [ 0, 0, 0, 1, 0, 669],
 [ 0, 0, 0, 0, 1, 377],
 [ 0, 0, 0, 0, 0, 0]],

order=1013)
> (alpha3,alpha13,alpha17,alpha19,alpha31) = r[:-1,-1].tolist()
> pow( 1000, alpha19, 2027 )
19
```

## Diskrete Logarithmen der Faktorbasiselemente (5)

Nun können wir beliebige diskrete Logarithmen berechnen. Um den diskreten Logarithmus von 1469 zur Basis g zu berechnen, suchen wir (durch Probieren) nach einer Zahl  $\gamma$ , so dass  $A \cdot g^\gamma \mathcal{F}$ -glatt ist. Wir finden (modulo 2027)

$$1469 \cdot 1000^{696} = 3^2 \cdot 19$$

$$1469 \cdot 1000^{696} = 1000^{2 \cdot 541 + 669}$$

$$1469 = 1000^{1055} = 1000^{42}$$

Der diskrete Logarithmus von 1469 zur Basis 1000 modulo 2027 ist also 42.

# Index-Calculus-Algorithmus: Aufwand

Dass es mit dem Index-Calculus-Algorithmus eine Methode gibt, diskrete Logarithmen modulo  $p$  zu berechnen, äußert sich nicht zuletzt in den Schlüssellängen, also in der Größe der Primzahl  $p$ . Reines Durchprobieren ließe sich mit Primzahlen von 128 Bit Länge bereits erfolgreich verhindern. Geschicktes Probieren (Baby-Step-Giant-Step) lässt sich mit 256 Bit langen Primzahlen verhindern.

Tatsächlich empfiehlt sich heute die Verwendung von zumindest 3072 Bit langen Primzahlen, um auch den Index-Calculus-Algorithmus zu aufwendig werden zu lassen.