10. Kapitel: Methoden zur Berechnung diskreter Logarithmen

Wir haben einige Public-Key-Verfahren kennengelernt, deren Sicherheit wesentlich darauf beruht, dass diskrete Logarithmen bei entsprechender Parameterwahl im Allgemeinen praktisch nicht zu berechnen sind: Diffie-Hellman-Schlüsselaustausch, ElGamal-Verschlüsselung, Massey-Omura-Nachrichtübertragung, ElGamal-Signatur, DSA.

1. Abschnitt: Einführung

Bemerkung: Ist p eine Primzahl, $g \in \mathbb{N}$ mit $\mathrm{ggT}(p, g) = 1$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(p, a) = 1$, so ist ein diskreter Logarithmus von a zur Basis g modulo p eine Lösung x der Gleichung

$$g^x \equiv a \bmod p.$$

Ein diskreter Logarithmus muss nicht existieren. Das folgende Lemma zeigt ein Existenzkriterium, wenn man die Ordnung von g modulo p kennt:

Lemma

Sei p eine Primzahl, $g \in \mathbb{Z}$ mit $\mathrm{ggT}(p, g) = 1$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(p, a) = 1$. Folgende Aussagen sind äquivalent:

(A) *Es gibt ein $x \in \mathbb{N}_0$ mit $g^x \equiv a \bmod p$.*
(B) $\mathrm{ord}_p(a) \mid \mathrm{ord}_p(g)$.
(C) $a^{\mathrm{ord}_p(g)} \equiv 1 \bmod p$.

(Kennt man also $\mathrm{ord}_p(g)$, so liefert (C) ein einfaches Kriterium für die Existenz des diskreten Logarithmus von a zur Basis g modulo p.)

Wir haben bereits die naive Methode zur Berechnung diskreter Logarithmen kennengelernt:

Naive Logarithmenberechnung

Eingabe: Primzahl p, $g, a \in \mathbb{F}_p^*$
Ausgabe: n mit $g^n = a$ oder $\operatorname{ord}_p(g)$

1: **if** $a = 1$ **then**
2: **return** 0 (mit $g^0 = a$)
3: **end if**
4: $h \leftarrow 1$ ▷ h ist g^n
5: **for** $n = 1, \ldots, p-2$ **do**
6: $h \leftarrow hg$ ▷ Nun ist $h = g^n$
7: **if** $h = a$ **then**
8: **return** n mit $g^n = a$
9: **end if**
10: **if** $h = 1$ **then**
11: **return** diskreter Logarithmus existiert nicht, $\operatorname{ord}_p(g) = n$
12: **end if**
13: **end for**

In diesem Kapitel sollen weitere Methoden vorgestellt werden.

2. Abschnitt: Logarithmenberechnung mit der Pollard-ρ-Methode

Wir wollen die Gleichung $g^x \equiv a \bmod p$ lösen (mit einer Primzahl p und $g, a \in \mathbb{Z}$ mit $p \nmid g$ und $p \nmid a$).

Bei der Pollard-ρ-Methode gibt es zwei Schritte.

1. Schritt:

- Ziel ist es, Zahlen $s, t \in \mathbb{N}$ zu finden mit

$$a^s \equiv g^t \bmod p.$$

- Wir teilen $\{1, 2, \ldots, p-1\}$ in drei ungefähr gleich große (disjunkte) Mengen S_1, S_2, S_3 ein. Für die folgenden Beispiele haben wir

$$S_i = \{x \in \mathbb{Z} : 1 \leq x \leq p-1, x \equiv i \bmod 3\}$$

gewählt. (Alternativ könnte man auch beispielsweise $S_1 = \{x \in \mathbb{Z} : 1 \leq x < \frac{p}{3}\}$, $S_2 = \{x \in \mathbb{Z} : \frac{p}{3} \leq x < \frac{2p}{3}\}$, $S_3 = \{x \in \mathbb{Z} : \frac{2p}{3} \leq x < p\}$ wählen.)

- Wir definieren eine Folge $x_0, x_1, x_2, \cdots \in \{1, \ldots, p-1\}$ durch

$$x_0 = 1 \quad \text{und} \quad x_{i+1} = \begin{cases} ax_i \bmod p & \text{für } x_i \in S_1, \\ x_i^2 \bmod p & \text{für } x_i \in S_2, \\ gx_i \bmod p & \text{für } x_i \in S_3. \end{cases}$$

Schreibt man $x_i \equiv a^{e_i} g^{f_i} \bmod p$, so hat man

$$e_0 = 0 \quad \text{und} \quad e_{i+1} = \begin{cases} e_i + 1 \bmod p-1 & \text{für } x_i \in S_1, \\ 2e_i \bmod p-1 & \text{für } x_i \in S_2, \\ e_i & \text{für } x_i \in S_3, \end{cases}$$

$$f_0 = 0 \quad \text{und} \quad f_{i+1} = \begin{cases} f_i & \text{für } x_i \in S_1, \\ 2f_i \bmod p-1 & \text{für } x_i \in S_2, \\ f_i + 1 \bmod p-1 & \text{für } x_i \in S_3. \end{cases}$$

- Verhält sich die Folge $x_0, x_1, x_2, \ldots$ nun wie eine Zufallsfolge in $\{1, \ldots, p-1\}$, so haben wir im Kapitel über die Pollardsche ρ-Faktorisierungsmethode gesehen, dass es sehr wahrscheinlich ist, dass es ein $i \leq 3.1\sqrt{p}$ gibt mit $x_i = x_{2i}$. Ist aber $x_i = x_{2i}$, so haben wir

$$a^{e_i} g^{f_i} \equiv x_i \equiv x_{2i} \equiv a^{e_{2i}} g^{f_{2i}} \bmod p,$$

also

$$a^{e_i - e_{2i}} \equiv g^{f_{2i} - f_i} \bmod p.$$

Mit

$$s = (e_i - e_{2i}) \bmod (p-1) \quad \text{und} \quad t = (f_{2i} - f_i) \bmod (p-1)$$

gilt dann

$$a^s \equiv g^t \bmod p.$$

- Der Vorteil ist nun, dass man die Folge $x_0, x_1, x_2, \ldots$ nicht speichern muss. Man berechnet einfach sukzessiv für $i = 0, 1, 2, \ldots$

$$(x_i, e_i, f_i, x_{2i}, e_{2i}, f_{2i})$$

und testet dann, ob $x_i = x_{2i}$ gilt.

Beispiel: $p = 113$, $g = 3$, $a = 5$. Die Folge $(x_i, e_i, f_i, x_{2i}, e_{2i}, f_{2i})$ sieht dann so aus:

$$
\begin{aligned}
i = 0: &\quad (1, 0, 0, 1, 0, 0) \\
i = 1: &\quad (5, 1, 0, 25, 2, 0) \\
i = 2: &\quad (25, 2, 0, 36, 3, 1) \\
i = 3: &\quad (12, 3, 0, 98, 3, 3) \\
i = 4: &\quad (36, 3, 1, 108, 7, 6) \\
i = 5: &\quad (108, 3, 2, 112, 14, 14) \\
i = 6: &\quad (98, 3, 3, 98, 15, 15)
\end{aligned}
$$

Nach 6 Schritten erhält man dann $s = e_6 - e_{12} = 100 \bmod 112$ und $t = f_{12} - f_6 = 12 \bmod 112$, sodass

$$5^{100} \equiv 3^{12} \bmod 113$$

gilt.

Logarithmenberechnung mit der ρ-Methode - Schritt 1

Eingabe: a, g, p

Ausgabe: s, t mit $a^s \equiv g^t \bmod p$

function $S(x, e, f)$
  if $x \equiv 1 \bmod 3$ **then**
    $x, e, f \leftarrow ax \bmod p, (e+1) \bmod (p-1), f$
  else if $x \equiv 2 \bmod 3$ **then**
    $x, e, f \leftarrow x^2 \bmod p, 2e \bmod (p-1), 2f \bmod (p-1)$
  else
    $x, e, f \leftarrow gx \bmod p, e, (f+1) \bmod (p-1)$
  end if
  return x, e, f
end function
$x, e, f, \tilde{x}, \tilde{e}, \tilde{f} \leftarrow 1, 0, 0, 1, 0, 0$
for $i = 1, \ldots, 10 \left\lfloor \sqrt{p} \right\rfloor$ **do**
  $x, e, f \leftarrow S(x, e, f)$
  $\tilde{x}, \tilde{e}, \tilde{f} \leftarrow S(\tilde{x}, \tilde{e}, \tilde{f})$
  $\tilde{x}, \tilde{e}, \tilde{f} \leftarrow S(\tilde{x}, \tilde{e}, \tilde{f})$
  if $x = \tilde{x}$ **then**
    $s, t \leftarrow (e - \tilde{e}) \bmod (p-1), (\tilde{f} - f) \bmod (p-1)$
    return s, t
  end if
end for

2. Schritt der Pollard-ρ-Methode:

- Wir nehmen nun an, dass wir eine Relation

$$a^s \equiv g^t \bmod p$$

  gefunden haben.
- Außerdem nehmen wir an, dass $\operatorname{ord}_p(g)$ bekannt ist. (Ist g eine Primitivwurzel modulo p, so ist $\operatorname{ord}_p(g) = p - 1$.)
- Existiert ein $x \in \mathbb{N}_0$ mit $a \equiv g^x \bmod p$, so erhalten wir folgende äquivalenten Umformungen

$$a^s \equiv g^t \bmod p \quad \Longleftrightarrow \quad g^{sx} \equiv g^t \bmod p \quad \Longleftrightarrow \quad sx \equiv t \bmod \operatorname{ord}_p(g).$$

  Die letzte Gleichung haben wir früher behandelt.
- Gilt $\operatorname{ggT}(s, \operatorname{ord}_p(g)) \nmid t$, so besitzt die Gleichung keine Lösung, es existiert also kein diskreter Logarithmus von a zur Basis g modulo p.
- Gilt nun $\operatorname{ggT}(s, \operatorname{ord}_p(g)) \mid t$, so gibt es genau $d = \operatorname{ggT}(s, \operatorname{ord}_p(g))$ Lösungen der Gleichung modulo $\operatorname{ord}_p(g)$, die wir explizit bestimmen können: $x_1, \ldots, x_d$. Nun testen wir, für welches x_i die Beziehung $g^{x_i} \equiv a \bmod p$ gilt, und wir sind fertig.

- Explizite Lösung im Fall $\mathrm{ggT}(s, \mathrm{ord}_p(g)) \mid t$: Sei $\tilde{s}$ das Inverse von $\frac{s}{\mathrm{ggT}(s,\mathrm{ord}_p(g))}$ modulo $\frac{\mathrm{ord}_p(g)}{\mathrm{ggT}(s,\mathrm{ord}_p(g))}$, d.h.

$$\tilde{s} \cdot \frac{s}{\mathrm{ggT}(s, \mathrm{ord}_p(g))} \equiv 1 \bmod \frac{\mathrm{ord}_p(g)}{\mathrm{ggT}(s, \mathrm{ord}_p(g))}.$$

Setzt man

$$x_0 = \tilde{s} \cdot \frac{t}{\mathrm{ggT}(s, \mathrm{ord}_p(g))} \bmod \frac{\mathrm{ord}_p(g)}{\mathrm{ggT}(s, \mathrm{ord}_p(g))},$$

so sind

$$x_i = x_0 + i \cdot \frac{\mathrm{ord}_p(g)}{\mathrm{ggT}(s, \mathrm{ord}_p(g))} \text{ für } i = 0, \ldots, \mathrm{ggT}(s, \mathrm{ord}_p(g)) - 1$$

die Lösungen der Gleichung $sx \equiv t \bmod \mathrm{ord}_p(x)$. Nun überprüft man, für welches x_i die Beziehung $g^{x_i} \equiv a \bmod p$ gilt.

Beispiel: Für $p = 113$, $g = 3$, $a = 5$ hatten wir die Beziehung

$$5^{100} \equiv 3^{12} \bmod 113$$

gefunden. $g = 3$ ist eine Primitivwurzel modulo p. Wir setzen an $5 \equiv 3^x \bmod 113$ und erhalten dann (für $x \in \{0, 1, \ldots, p-2\}$)

$$\begin{aligned}
(3^x)^{100} \equiv 3^{12} \bmod 113 \quad &\Longleftrightarrow \quad 3^{100x} \equiv 3^{12} \bmod 113 \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad 100x \equiv 12 \bmod 112 \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad 25x \equiv 3 \bmod 28 \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad -3x \equiv 3 \bmod 28 \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad x \equiv -1 \bmod 28 \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad x \in \{27, 55, 83, 111\}.
\end{aligned}$$

Nun gilt

$$3^{27} \equiv 108 \bmod p, \quad 3^{55} \equiv 75 \bmod p, \quad 3^{83} \equiv 5 \bmod p, \quad 3^{111} \equiv 38 \bmod p,$$

also ist 83 diskreter Logarithmus von 5 zur Basis 3 modulo 113.

Beispiele:

p	$10^{12}+547$	$10^{13}+259$	$10^{14}+5083$	$10^{15}+5719$
g	2	2	2	11
a	3	3	3	2
x	126279695832	4608949908312	78019496252700	811999217052726
s	478977044132	4104162820200	62388752267924	683074195741160
t	474549283964	279615485454	56811232574790	903354422956900
$\mathrm{ggT}(s, p-1)$	2	2	2	2
Schritte	782120	2847267	13131534	33179403
Schrittzahl/sqrt(p)	0.78	0.90	1.31	1.05

Beispiel: $p = 10^{12} + 39$, $g = 3$

a	Schrittzahl$/\sqrt{p}$
2	0.55
3	1.26
4	1.13
5	1.72
6	1.45
7	1.04
8	1.00
9	1.33
10	0.83

Bemerkungen:

- Das Verfahren ist nicht deterministisch, die Laufzeit ist $O(p^{\frac{1}{2}+\epsilon})$. Ein Vorteil ist, dass man praktisch keinen Speicherplatz braucht.
- Wie sollte man S_1, S_2, S_3 wählen?
- Das Verfahren lässt sich leicht auch für andere Gruppen G durchführen, man muss nur G in drei disjunkte Mengen S_1, S_2, S_3 aufteilen, die ungefähr gleich groß sind. Bei der Definition von x_{i+1} wird dann danach unterschieden, ob $x_i \in S_1$, $x_i \in S_2$ oder $x_i \in S_3$ gilt.

3. Abschnitt: Logarithmenberechnung mit der $(p-1)$-Methode

Für die folgende Logarithmenberechnungsmethode ($g^x \equiv a \bmod p$) braucht man die Faktorzerlegung der Ordnung von g modulo p. Ist g eine Primitivwurzel modulo p, so ist $\operatorname{ord}_p(g) = p-1$. Diese Methode ist auch unter dem Namen Silver-Pohlig-Hellman-Methode bekannt.

Satz

Sei p eine ungerade Primzahl, $g \in \mathbb{Z}$ mit $\operatorname{ggT}(p, g) = 1$ und

$$\operatorname{ord}_p(g) = u_1 \dots u_r$$

mit paarweise teilerfremden natürlichen Zahlen $u_1, \dots, u_r$. (Ist $\operatorname{ord}_p(g) = q_1^{e_1} \dots q_r^{e_r}$ die Primfaktorzerlegung von $\operatorname{ord}_p(g)$, so kann man $u_i = q_i^{e_i}$ wählen.) Sei $a \in \mathbb{Z}$ mit $\operatorname{ggT}(p, a) = 1$, sodass gilt

$$a^{\operatorname{ord}_p(g)} \equiv 1 \bmod p.$$

(Dies garantiert die Existenz eines diskreten Logarithmus von a zur Basis g modulo p.)

- *Für jedes $i = 1, \dots, r$ gibt es eine Zahl $x_i \in \mathbb{N}_0$ mit*

$$\left(g^{\frac{\operatorname{ord}_p(g)}{u_i}}\right)^{x_i} \equiv a^{\frac{\operatorname{ord}_p(g)}{u_i}} \bmod p \quad \text{und} \quad 0 \leq x_i \leq u_i - 1.$$

  x_i ist durch diese Bedingungen eindeutig bestimmt.

- *Für $x \in \mathbb{N}_0$ gilt*

$$g^x \equiv a \bmod p \quad \Longleftrightarrow \quad x \equiv \begin{cases} x_1 \bmod u_1, \\ \vdots \\ x_r \bmod u_r, \end{cases}$$

  kennt man also $x_1, \dots, x_r$, so findet man eine Lösung der Gleichung $g^x \equiv a \bmod p$ mit dem chinesischen Restsatz.

Beweis:

- Die Voraussetzung $a^{\operatorname{ord}_p(g)} \equiv 1 \bmod p$ garantiert die Existenz einer Zahl $\ell \in \mathbb{N}_0$ mit
$$g^{\ell} \equiv a \bmod p.$$

- Wir potenzieren $g^{\ell} \equiv a \bmod p$ mit $\frac{\operatorname{ord}_p(g)}{u_i}$:
$$g^{\frac{\operatorname{ord}_p(g)}{u_i} \cdot \ell} \equiv a^{\frac{\operatorname{ord}_p(g)}{u_i}} \bmod p.$$
Da $g^{\frac{\operatorname{ord}_p(g)}{u_i}}$ Ordnung u_i hat, folgt
$$(g^{\frac{\operatorname{ord}_p(g)}{u_i}})^{\ell \bmod u_i} = g^{\frac{\operatorname{ord}_p(g)}{u_i} \cdot \ell} \equiv a^{\frac{\operatorname{ord}_p(g)}{u_i}} \bmod p.$$
Also können wir $x_i = \ell \bmod u_i$ wählen. Wegen $\operatorname{ord}(g^{\frac{\operatorname{ord}_p(g)}{u_i}}) = u_i$ ist x_i modulo u_i eindeutig bestimmt.
- Wegen $x_i \equiv \ell \bmod u_i$ erhalten wir:
$$\begin{aligned} g^x \equiv a \bmod p \quad &\Longleftrightarrow \quad g^x \equiv g^{\ell} \bmod p \quad \Longleftrightarrow \quad x \equiv \ell \bmod \operatorname{ord}_p(g) \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad x \equiv \ell \bmod u_i \text{ für } i = 1, \ldots, r \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad x \equiv x_i \bmod u_i \text{ für } i = 1, \ldots, r, \end{aligned}$$
wie behauptet. ■

Logarithmenberechnung mit der $(p-1)$-Methode

Eingabe: p ungerade Primzahl, $g \in \mathbb{Z}$ mit $p \nmid g$, $a \in \mathbb{Z}$ mit $p \nmid a$, Primfaktorzerlegung von $\operatorname{ord}_p(g)$

$$\operatorname{ord}_p(g) = q_1^{e_1} \dots q_r^{e_r}.$$

(Es muss $a^{\operatorname{ord}_p(g)} \equiv 1 \bmod p$ gelten.)

Ausgabe: Ein $x \in \mathbb{N}_0$ mit $g^x \equiv a \bmod p$. (x ist ein diskreter Logarithmus von a zur Basis g modulo p.)

1: **for** $i = 1, \dots, r$ **do**
2: $u_i \leftarrow q_i^{e_i}$.
3: Bestimme ein $x_i \in \{0, 1, \dots, u_i - 1\}$ mit $(g^{\frac{\operatorname{ord}_p(g)}{u_i}})^{x_i} \equiv a^{\frac{\operatorname{ord}_p(g)}{u_i}} \bmod p$ (mit einer Methode zur Berechnung diskreter Logarithmen)
4: **end for**
5: Bestimme mit dem chinesischen Restsatz ein $x \in \mathbb{N}_0$ mit $x \equiv x_i \bmod u_i$ für $i = 1, \dots, r$. (Man kann dabei $0 \leq x \leq \operatorname{ord}_p(g)$ erreichen.)
6: **return** x

Beispiel: $p = 10^{10} + 19$ und $g = 2$. Dann ist

$$p - 1 = 2 \cdot 131 \cdot 521 \cdot 73259.$$

Wir wollen die Gleichung $2^x \equiv 3 \bmod p$ lösen. (2 ist Primitivwurzel.)
Nach dem eben geschilderten Verfahren erhalten wir folgende 4 Gleichungen:

$$\begin{aligned}
2^{\frac{p-1}{2}x_1} &\equiv 3^{\frac{p-1}{2}} \bmod p, \quad \text{d.h.} \quad 10000000018^{x_1} \equiv 1 \bmod p \\
2^{\frac{p-1}{131}x_1} &\equiv 3^{\frac{p-1}{131}} \bmod p, \quad \text{d.h.} \quad 6098757036^{x_2} \equiv 1552837932 \bmod p, \\
2^{\frac{p-1}{521}x_1} &\equiv 3^{\frac{p-1}{521}}, \bmod p \quad \text{d.h.} \quad 4211130545^{x_3} \equiv 2093263771 \bmod p \\
2^{\frac{p-1}{73259}x_1} &\equiv 3^{\frac{p-1}{73259}} \bmod p, \quad \text{d.h.} \quad 2205818057^{x_4} \equiv 6804285272 \bmod p.
\end{aligned}$$

Dabei kann man $0 \leq x_1 \leq 1$, $0 \leq x_2 \leq 130$, $0 \leq x_3 \leq 520$, $0 \leq x_4 \leq 73258$ wählen. Durch Probieren bzw. mit naiver Logarithmenberechnung findet man

$$x_1 = 0, \quad x_2 = 92, \quad x_3 = 223, \quad x_4 = 55292.$$

Dies liefert das Kongruenzgleichungssystem

$$x \equiv \begin{cases} 0 \bmod 2, \\ 92 \bmod 131, \\ 223 \bmod 521, \\ 55292 \bmod 73259. \end{cases}$$

Mit dem chinesischen Restsatz finden wir $x = 5181957398$, das $x \equiv x_i \bmod q_i^{e_i}$ für $1 \leq i \leq 4$ erfüllt und damit die Gleichung löst.

Bemerkungen:

- Berechnet man den diskreten Logarithmus von $a^{\frac{\operatorname{ord}_p(g)}{u_i}}$ zur Basis $g^{\frac{\operatorname{ord}_p(g)}{u_i}}$ modulo p mit der naiven Methode, so ist die Schrittzahl in etwa $O(\sum_{i=1}^{r} q_i^{e_i})$. Wir werden im nächsten Verfahren dies noch reduzieren auf $O(\sum_{i=1}^{r} e_i q_i)$. Man kann das Verfahren schneller machen, wenn man statt der naiven Methode eine bessere Logarithmenberechnungsmethode verwendet. Gut funktioniert das Verfahren, wenn $\operatorname{ord}_p(g)$ nur „kleine“ Primteiler hat.
- Kann man $\operatorname{ord}_p(g)$ nicht bestimmen, dann kann man das Verfahren auch nicht anwenden.

Beispiel: Wir betrachten die 78-stellige Primzahl

$p = 103089314192586000849956088883567437099862384829959097519276671552027932939001$

Man findet

$$\begin{aligned} p-1 \quad = \quad & 2^3 \cdot 3 \cdot 5^3 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59 \cdot 61 \cdot 67 \cdot 71 \cdot 73 \cdot 79 \cdot \\ & \cdot 83 \cdot 89 \cdot 97 \cdot 101 \cdot 103 \cdot 107 \cdot 109 \cdot 113 \cdot 127 \cdot 131 \cdot 137 \cdot 139 \cdot 149 \cdot 151 \cdot 157 \cdot 163 \cdot 167 \cdot \\ & \cdot 173 \cdot 179 \cdot 181 \cdot 191. \end{aligned}$$

Damit stellt man schnell fest, dass 193 die kleinste Primitivwurzel modulo p ist. Wir wollen den diskreten Logarithmus von $a = 2$ zur Basis $g = 193$ modulo p berechnen. Wir setzen

$$u_1 = 2^3, \quad u_2 = 3, \quad u_3 = 5^3, \quad u_4 = 7, \quad \ldots, \quad u_{43} = 191,$$

sodass also $p - 1 = u_1 u_2 u_3 \ldots u_{43}$ mit paarweise teilerfremden u_i gilt.
Nun bestimmen wir (mit naiver Logarithmenberechnung) Zahlen x_i mit

$$193^{\frac{p-1}{u_i} \cdot x_i} \equiv 2^{\frac{p-1}{u_i}} \bmod p \text{ und } \quad 0 \leq x_i \leq u_i - 1.$$

Man findet: $x_1, \ldots, x_{43}$:

$$\begin{aligned} & 6, 1, 11, 0, 9, 1, 4, 9, 14, 26, 12, 0, 21, 17, 1, 40, 27, 12, 1, 48, 22, 45, 31, 23, 96, 80, 19, 5, \\ & 34, 82, 89, 102, 43, 135, 39, 64, 22, 131, 36, 167, 16, 85, 92. \end{aligned}$$

Nun bestimmen wir mit dem chinesischen Restsatz x mit $x \equiv x_i \bmod u_i$ für $i = 1, \ldots, 43$ und erhalten

$x = 102742615102612239422105609955921982008376867256578190090301674188785288132886.$

Tatsächlich gilt nun $193^x \equiv 2 \bmod p$.

Beispiele:

p	Primfaktorzerlegung von $p-1$	Gleichung	Lösung
$10^{20}+441$	$2^3 \cdot 3^3 \cdot 5 \cdot 31 \cdot 20200847 \cdot 147858049$	$17^x = 2$	61849824164554360796
$10^{20}+441$	$2^3 \cdot 3^3 \cdot 5 \cdot 31 \cdot 20200847 \cdot 147858049$	$17^x = 3$	87884194200915358338
$10^{20}+441$	$2^3 \cdot 3^3 \cdot 5 \cdot 31 \cdot 20200847 \cdot 147858049$	$17^x = 5$	58895325358520474016
$10^{20}+207$	$2 \cdot 3^2 \cdot 811 \cdot 1531 \cdot 161521 \cdot 27701447$	$5^x = 2$	56125417435406247014
$10^{20}+207$	$2 \cdot 3^2 \cdot 811 \cdot 1531 \cdot 161521 \cdot 27701447$	$5^x = 3$	11472450610262189697
$10^{20}+757$	$2^2 \cdot 23 \cdot 337 \cdot 10271 \cdot 32173 \cdot 9760633$	$2^x = 3$	14739254935815201045
$10^{20}+757$	$2^2 \cdot 23 \cdot 337 \cdot 10271 \cdot 32173 \cdot 9760633$	$2^x = 5$	73924078420212120819
$10^{20}+801$	$2^5 \cdot 3^2 \cdot 5^2 \cdot 97 \cdot 26209 \cdot 32779 \cdot 166667$	$11^x = 2$	23453734185172248008
$10^{20}+801$	$2^5 \cdot 3^2 \cdot 5^2 \cdot 97 \cdot 26209 \cdot 32779 \cdot 166667$	$11^x = 3$	41775657877555128374
$10^{20}+9061$	$2^2 \cdot 5 \cdot 41 \cdot 43 \cdot 277 \cdot 751^2 \cdot 1483 \cdot 12241$	$2^x = 3$	27490110764324561483
$10^{20}+9061$	$2^2 \cdot 5 \cdot 41 \cdot 43 \cdot 277 \cdot 751^2 \cdot 1483 \cdot 12241$	$2^x = 5$	47656374274610060480
$10^{20}+12501$	$2^2 \cdot 3^2 \cdot 5^5 \cdot 7^2 \cdot 43 \cdot 163 \cdot 409 \cdot 1423 \cdot 4447$	$6^x = 2$	57498253311736065081
$10^{20}+12501$	$2^2 \cdot 3^2 \cdot 5^5 \cdot 7^2 \cdot 43 \cdot 163 \cdot 409 \cdot 1423 \cdot 4447$	$6^x = 5$	71851417937269344958

Bemerkung: Für das vorangegangene Verfahren braucht man eine Zerlegung

$$p - 1 = u_1 u_2 \dots u_r$$

mit paarweise teilerfremden Zahlen $u_1, \dots, u_r$. Das Verfahren funktioniert gut, wenn alle Zahlen u_i klein sind.

Beispiel: Die Zahl

$$p = 1 + 2^{499} \cdot 3^{411}$$

ist eine Primzahl mit 347 Stellen. Obwohl $p - 1$ nur die Primteiler 2 und 3 hat, lässt sich obiges Verfahren hier nicht sinnvoll anwenden.
Hier kann man die nachfolgende Vorgehensweise anwenden.

Mit nachfolgendem Lemma wird die Logarithmenberechnung zu einer Basis g mit Primzahlpotenzordnung q^e zurückgeführt auf die Berechnung von e Logarithmen zu einer Basis g_q mit Primzahlordnung q.

Lemma

Sei p eine Primzahl und $g \in \mathbb{N}$ mit $\mathrm{ggT}(p, g) = 1$ und $\mathrm{ord}_p(g) = q^e$ mit einer Primzahl q und $e \in \mathbb{N}$. Ist $a \in \mathbb{Z}$ mit $\mathrm{ggT}(p, a) = 1$ und $a^{q^e} \equiv 1 \bmod p$, so gibt es ein $x \in \mathbb{Z}$ mit $0 \leq x \leq q^e - 1$, sodass $g^x \equiv a \bmod p$ gilt. Es gibt $x_i \in \{0, 1, \ldots, q-1\}$ mit

$$x = x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}.$$

Es gilt

$$(g^{q^{e-1}})^{x_i} = \left(a \cdot (g^{-1})^{\sum_{j=0}^{i-1} x_j q^j}\right)^{q^{e-1-i}} \quad \text{für } 0 \leq i \leq e-1,$$

also

$$(g^{q^{e-1}})^{x_0} = a^{q^{e-1}}, (g^{q^{e-1}})^{x_1} = \left(a \cdot (g^{-1})^{x_0}\right)^{q^{e-2}}, \quad \ldots \quad (g^{q^{e-1}})^{x_{e-1}} = \left(a \cdot (g^{-1})^{x_0 + x_1 q + \cdots + x}\right.$$

Durch diese Gleichungen sind $x_0, x_1, \ldots, x_{e-1}$ eindeutig bestimmt, sie sind diskrete Logarithmen zur Basis $g^{q^{e-1}}$ modulo p, wobei $g^{q^{e-1}}$ die Ordnung q modulo p hat.

Beweis:

- Die Bedingung $a^{\operatorname{ord}(g)} \equiv 1 \bmod p$ sichert die Existenz einer Zahl x mit $g^x \equiv a \bmod p$. Wegen $\operatorname{ord}_p(g) = q^e$ ist x modulo q^e eindeutig bestimmt. Wir können also $0 \leq x \leq q^e - 1$ annehmen und die q-adische Entwicklung von x betrachten:

$$x = x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1} \quad \text{mit} \quad 0 \leq x_i \leq q - 1.$$

  Natürlich sind dann auch $x_0, x_1, \ldots, x_{e-1}$ eindeutig bestimmt.
- Es gilt für $0 \leq i \leq e - 1$

$$a \equiv g^x \equiv g^{\sum_{j=0}^{e-1} x_j q^j} \equiv g^{x_i q^i} \cdot g^{\sum_{j=0}^{i-1} x_j q^j} \cdot g^{\sum_{j=i+1}^{e-1} x_j q^j} \bmod p,$$

  daher gilt

$$g^{x_i q^i} \equiv a \cdot g^{-\sum_{j=0}^{i-1} x_j q^j} \cdot g^{-\sum_{j=i+1}^{e-1} x_j q^j} \bmod p.$$

  Potenzieren mit q^{e-i-1} liefert wegen $g^{q^e} \equiv 1 \bmod p$

$$(g^{q^{e-1}})^{x_i} \equiv \left(a \cdot g^{-\sum_{j=0}^{i-1} x_j q^j} \right)^{q^{e-i-1}} \bmod p.$$

  Dies sind genau die behaupteten Gleichungen.
- Nun gilt

$$\operatorname{ord}_p(g^{q^{e-1}}) = q,$$

  weswegen $x_0, x_1, \ldots, x_{e-1}$ nacheinander durch obige Gleichungen eindeutig bestimmt sind. ■

Bemerkung: Schreiben wir mit den Bezeichnungen des Lemmas

$$g_q \equiv g^{q^{e-1}} \bmod p \quad \text{und} \quad \tilde{x}_i = \sum_{j=0}^{i-1} x_j q^j,$$

so gilt

$$g_q^{x_i} \equiv \left(a \cdot (g^{-1})^{\tilde{x}_i}\right)^{q^{e-1-i}} \bmod p \text{ für } 0 \leq i \leq e-1$$

und

$$\tilde{x}_0 = 0, \quad \tilde{x}_{i+1} = \tilde{x}_i + x_i q^i, \quad \tilde{x}_e = x.$$

Damit erhält man sofort nachfolgenden Algorithmus.

Logarithmenberechnung im Fall $\operatorname{ord}_p(g) = q^e$

Eingabe: Primzahl p, $g \in \mathbb{N}$, Primzahl q, $e \in \mathbb{N}$ mit $\operatorname{ord}_p(g) = q^e$, $a \in \mathbb{Z}$ mit $a^{q^e} \equiv 1 \bmod p$.

Ausgabe: $x \in \{0, 1, \ldots, q^e - 1\}$ mit $g^x \equiv a \bmod p$.

1: $g_q \leftarrow g^{q^{e-1}} \bmod p$, $g_{\text{inv}} \leftarrow g^{-1} \bmod p$ (oder $g_{\text{inv}} \leftarrow g^{q^e-1} \bmod p$).
2: $x \leftarrow 0$.
3: **for** $i = 0, 1, \ldots, e-1$ **do**
4: Bestimme $x_i \in \{0, 1, \ldots, q-1\}$ mit $g_q^{x_i} \equiv (a \cdot g_{\text{inv}}^x)^{q^{e-1-i}} \bmod p$. ($x_i$ ist diskreter Logarithmus zur Basis g_q.)
5: $x \leftarrow x + x_i q^i$.
6: **end for**
7: **return** x (diskreter Logarithmus von a zur Basis g modulo p).

Bemerkung: Mit diesem Lemma bzw. Algorithmus reduziert man also die Berechnung eines diskreten Logarithmus zur Basis g, wo g Primzahlpotenz q^e hat, auf die Berechnung von e diskreten Logarithmen zur Basis g_q, wo g_q Primzahlordnung q hat. Ist $e \geq 2$ und q^e größer, so ist dies von großem Vorteil.

Beispiel: Wir betrachten folgende Situation:

$$\begin{aligned} p &= 702857981670509306221075879002735600669348806920438223719507\\ g &= 605583059411022095567350578112877881821659500487969573336424\\ a &= 169105058115399344398468379360382833774642593911657935895190\end{aligned}$$

Dabei ist p eine 80-stellig Primzahl mit

$$p - 1 = 2^{256} \cdot 607.$$

Außerdem ist 3 eine Primitivwurzel modulo p und

$$g \equiv 3^{\frac{p-1}{2^{256}}} \equiv 3^{607} \bmod p$$

ein Element der Ordnung 2^{256}:

$$\operatorname{ord}_p(g) = 115792089237316195423570985008687907853269984665640564039$$

Man testet, dass $a^{2^{256}} \equiv 1 \bmod p$ gilt. Mit obigem Algorithmus findet man sofort den diskreten Logarithmus x von a zur Basis g modulo p:

$$x = 434933914215573238797940691534741818933603158168696368646525843$$

4. Abschnitt: Berechnung diskreter Logarithmen mit der baby-step-giant-step-Methode nach Shanks

Überlegungen: Sei p eine Primzahl und $g, a \in \mathbb{Z}$ mit $\mathrm{ggT}(p, g) = \mathrm{ggT}(p, a) = 1$. Wir suchen nach einer Lösung der Gleichung $g^x \equiv a \bmod p$.

- Existiert eine Lösung x der Gleichung $g^x \equiv a \bmod p$, so kann man o.E. $0 \leq x \leq \mathrm{ord}_p(g) - 1$ annehmen.
- Wir wählen eine natürliche Zahl m mit $m^2 \geq \mathrm{ord}_p(g)$. Wenn wir $\mathrm{ord}_p(g)$ nicht kennen, wählen wir $m^2 \geq p - 1$.
- Jede ganze Zahl x mit $0 \leq x \leq m^2 - 1$ hat eine eindeutige Zerlegung

$$x = i + mj \quad \text{mit} \quad 0 \leq i, j \leq m - 1.$$

Dabei ist

$$i = x \bmod m \quad \text{und} \quad j = \left\lfloor \frac{x}{m} \right\rfloor.$$

- Sei jetzt $g^x \equiv a \bmod p$ mit $0 \leq x \leq \operatorname{ord}_p(g) - 1$. Wir können zerlegen $x = i + mj$ mit $0 \leq i, j \leq m - 1$ und erhalten dann aus $a \equiv g^x \equiv g^{i+mj} \equiv g^i \cdot g^{mj} \bmod p$ die Gleichung

$$g^i \equiv a \cdot (g^{-m})^j \bmod p.$$

- Umgekehrt: Finden wir i, j mit $0 \leq i, j \leq m - 1$ und

$$(g^i \bmod p) = (a \cdot (g^{-m})^j \bmod p),$$

so löst $x = i + mj$ offensichtlich die Gleichung $g^x \equiv a \bmod p$.

- Wie kann man i, j mit

$$(g^i \bmod p) = (a \cdot (g^{-m})^j \bmod p)$$

finden? Probiert man naiv alle Paare $(i, j) \in \{0, 1, \ldots, m - 1\} \times \{0, 1, \ldots, m - 1\}$ durch, so braucht man dafür $O(m^2)$ Schritte, was wegen $m^2 \approx \operatorname{ord}_p(g)$ kein Vorteil gegenüber der naiven Methode ist.

- Eine Möglichkeit wäre, Hashtabellen zu benutzen, die in Python durch `dictionary` realisiert werden. Wir legen eine Hashtabelle B an und füllen sie mit

$$B(g^i \bmod p) = i \text{ für } i = 0, 1, \ldots, m-1.$$

Nun berechnen wir nacheinander für $j = 0, 1, 2, \ldots, m-1$

$$h_j = a(g^{-m})^j \bmod p,$$

was rekursiv durch

$$h_0 = a \quad \text{und} \quad h_{j+1} = g^{-m} h_j \bmod p$$

passieren kann.

  - Wir schauen nach, ob h_j im Definitionsbereich von B ist. (Die Laufzeit ist hier erstaunlicherweise $O(1)$.) Ist h_j im Definitionsbereich von B, so setzen wir

$$i = B(h_j)$$

und erhalten dann hieraus sofort den diskreten Logarithmus von a zu Basis g modulo p:

$$x = i + mj,$$

denn es gilt

$$(g^i \bmod p) = (a(g^{-m})^j) \bmod p.$$

  - Gibt es kein j, sodass h_j im Definitionsbereich von B liegt, so existiert kein $x \in \{0, 1, \ldots, m^2 - 1\}$ mit $g^x \equiv a \bmod p$.

Die Schrittzahl lässt sich durch $O(m)$ abschätzen.

Beispiel: Wir betrachten $p = 101$, $g = 2$ und $a = 3$. Es ist $\operatorname{ord}_p(g) = p - 1$. Die Zahl $m = 10$ erfüllt die Bedingung $m^2 \geq \operatorname{ord}_p(g)$. Wir berechnen zunächst

x	0	1	2	3	4	5	6	7	8	9
$B(x)$	1	2	4	8	16	32	64	27	54	7

Sei $\tilde{g} = g^{-m} \bmod p = 65$ und $h_j = a(g^{-m})^j \bmod p = 3 \cdot 65^j \bmod p$, also

$$h_0 = 3, \quad h_1 = 94, \quad h_2 = 50, \quad h_3 = 18, \quad h_4 = 59, \quad h_5 = 98, \quad h_6 = 7.$$

Es ist also $B(9) = 7 = h_6$. Daher ist $x = 9 + 6 \cdot 10 = 69$ der diskrete Logarithmus von 3 zur Basis 2 modulo 101.

Bemerkungen:

- Wie Beispiele zeigen, wird der Speicherplatz schnell zum Problem. Daher ist das Verfahren eher von theoretischem Interesse. Es zeigt, dass man diskrete Logarithmen mit einer Laufzeit von $O(p^{\frac{1}{2}+\epsilon})$ berechnen kann.
- Die baby-step-giant-step-Methode wird auch an anderen Stellen in der Zahlentheorie verwendet. Die Idee geht auf Shanks zurück.

Logarithmenberechnung mit der baby-step-giant-step-Methode

Eingabe: p, g, a, eventuell eine obere Schranke $m_{\operatorname{ord}_p(g)}$ für $\operatorname{ord}_p(g)$

Ausgabe: x mit $g^x \equiv a \bmod p$

1: Ist $m_{\operatorname{ord}_p(g)}$ nicht gegeben, setze $m_{\operatorname{ord}_p(g)} \leftarrow p-1$
2: $m \leftarrow \lceil \sqrt{m_{\operatorname{ord}_p(g)}} \rceil$
3: Lege eine Hashtabelle/dictionary B an mit $B(1) = 0$.
4: $g_i, i \leftarrow 1, 0$
5: **while** $i < m-1$ **do**
6: $\quad g_i, i \leftarrow (g_i g) \bmod p, i+1$
7: $\quad B(g_i) = i$
8: **end while**
9: $\tilde{g}, h_j \leftarrow g^{p-1-m} \bmod p, a$
10: **for** $j = 0, \ldots, m-1$ **do**
11: $\quad$ **if** h_j ist im Definitionsbereich von B **then**
12: $\quad\quad i = B(h_j)$
13: $\quad\quad x \leftarrow i + mj$
14: $\quad\quad$ **return** x
15: $\quad$ **end if**
16: $\quad h_j \leftarrow (h_j \tilde{g}) \bmod p$
17: **end for**
18: **return** Der diskrete Logarithmus existiert nicht oder $m_{\operatorname{ord}_p(g)}$ wurde zu klein gewählt.

Beispiele:

- $p = 211$, $g = 2$, $a = 3$, $m = 15$

x	1	2	4	8	16	32	64	128	45	90	180	149	87	174	137
$B(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$\tilde{g} = 67$

$$h_0 = 3, \quad h_1 = 201, \quad h_2 = 174$$

$B(174) = 13$, $i = 13$, $j = 2$, $x = i + mj = 43$

- $p = 241$, $g = 7$, $a = 2$, $m = 16$

x	1	7	49	102	232	178	41	46	81	85	113	68	235	199	188	111
$B(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$\tilde{g} = 183$

$h_0 = 2, \quad h_1 = 125, \quad h_2 = 221, \quad h_3 = 196, \quad h_4 = 200, \quad h_5 = 209,$

$h_6 = 169, \quad h_7 = 79, \quad h_8 = 238, \quad h_9 = 174, \quad h_{10} = 30, \quad h_{11} = 188$

$B(188) = 14$, $i = 14$, $j = 11$, $x = i + mj = 190$

- $p = 241$, $g = 2$, $a = 7$, $m = 16$

x	1	2	4	8	16	32	64	128	15	30	60	120	240	239	237	233
$B(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$\tilde{g} = 15$

$h_0 = 7, \quad h_1 = 105, \quad h_2 = 129, \quad h_3 = 7, \quad h_4 = 105, \quad h_5 = 129,$

$h_6 = 7, \quad h_7 = 105, \quad h_8 = 129, \quad h_9 = 7, \quad h_{10} = 105, \quad h_{11} = 129,$

$h_{12} = 7, \quad h_{13} = 105, \quad h_{14} = 129, \quad h_{15} = 7.$

Hier findet man keine Lösung, was wegen $\operatorname{ord}_{241}(2) = 24$ und $\operatorname{ord}_{241}(7) = 240$ auch klar ist.

5. Abschnitt: Die Index-Calculus-Methode

Sei p eine ungerade Primzahl, g eine Primitivwurzel modulo p und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(p, a) = 1$. Wir wollen die Gleichung $g^x \equiv a \bmod p$ lösen. Dies erfolgt in 3 Schritten.

1. Schritt: Wir wählen eine natürliche Zahl n. Seien $q_1, q_2, \ldots, q_n$ die ersten n Primzahlen. Wir brauchen zunächst einige Relationen

$$\prod_{j=1}^{n} q_j^{a_{ij}} \equiv g^{b_i} \bmod p \text{ für } i = 1, 2, \ldots, m \text{ mit } m \geq n$$

und gehen dazu wie folgt vor:

- Wähle b mit $0 \leq b \leq p-2$ zufällig.
- Berechne $x = g^b \bmod p$. (Dann ist $1 \leq x \leq p-1$.)
- Faktorisiere aus x alle q_i, $i = 1, \ldots, n$ heraus, solange es geht:

$$x = q_1^{a_1} q_2^{a_2} \ldots q_m^{a_m} \cdot \tilde{x} \text{ mit } \tilde{x} \in \mathbb{N} \text{ und } \mathrm{ggT}(\tilde{x}, q_1 \ldots q_n) = 1.$$

- Ist $\tilde{x} > 1$, fängt man von vorne an, d.h. man wählt ein anderes b. Ist $\tilde{x} = 1$, haben wir eine gewünschte Relation

$$q_1^{a_1} q_2^{a_2} \ldots q_m^{a_m} \equiv g^b \bmod p$$

gefunden.

Beispiel: $p = 10009$, $g = 11$ und $n = 3$. Wir suchen also Relationen

$$11^{b_i} \equiv 2^{a_{i1}} \cdot 3^{a_{i2}} \cdot 5^{a_{i3}} \bmod p.$$

Durch zufällige Wahl von b finden wir:

b_i	a_{i1}	a_{i2}	a_{i3}	Anzahl der Versuche
5140	11	1	0	61
3438	2	1	1	88
6876	4	2	2	163
4374	2	3	0	8
3620	3	5	0	5
5626	1	0	2	77

(Anzahl der Versuche meint dabei die Anzahl der Versuche, bis man ein geeignetes b gefunden hat.)

2. Schritt: Wir haben also jetzt einige Relationen

$$\prod_{j=1}^{m} q_j^{a_{ij}} \equiv g^{b_i} \text{ für } i = 1, 2, \ldots, m \text{ mit } m \geq n.$$

Schreiben wir $q_i \equiv g^{\ell_i} \bmod p$ mit $0 \leq \ell_i \leq p-2$, so werden die Relationen zu

$$\sum_{j=1}^{n} a_{ij}\ell_j \equiv b_i \bmod p-1 \text{ für } i = 1, 2, \ldots, m.$$

(ℓ_i ist der diskrete Logarithmus von q_i zur Basis g modulo p.) Dies ist ein lineares Gleichungssystem über dem Ring $\mathbb{Z}/(p-1)\mathbb{Z}$ mit den Unbekannten $\ell_1, \ldots, \ell_n$. Wie löst man ein solches Gleichungssystem?

Wir skizzieren eine Möglichkeit, die so gemacht ist, dass wir SAGE verwenden können. Wir bilden die zum Gleichungssystem gehörige (erweiterte) Koeffizientenmatrix

$$M = \begin{pmatrix} a_{11} & \dots & a_{1n} & b_1 \\ a_{21} & \dots & a_{2n} & b_2 \\ \vdots & & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} & b_m \end{pmatrix}$$

und betrachten sie als Matrix mit Einträgen in $\mathbb{Z}$. (Das $\bmod(p-1)$ beachten wir nicht.) Elementare Zeilenumformungen über $\mathbb{Z}$ ändern die Lösungen nicht:

- Addition des ganzzahligen Vielfachen einer Zeile zu einer anderen Zeile.
- Vertauschen zweier Zeilen.
- Multiplikation einer Zeile mit -1.

Man führt jetzt eine Art Gauß-Verfahren durch. Man kommt dann zur sogenannten hermiteschen Normalform. (SAGE: `M.hermite_form()`)

Wenn man viele Relationen hat, kann man hoffen, auf folgende Form zu kommen:

$$\begin{pmatrix} 1 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & c_2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & c_n \\ 0 & 0 & \dots & 0 & c_{n+1} \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix}$$

Nun lesen wir die zugehörigen Gleichungen wieder modulo $p-1$ und erhalten

$$\ell_1 \equiv c_1 \bmod (p-1), \quad \dots \quad \ell_n \equiv c_n \bmod (p-1)$$

und

$$c_{n+1} \equiv 0 \bmod (p-1).$$

In diesem Fall kennen wir nun $\ell_1, \dots, \ell_n$, die diskreten Logarithmen von $q_1 = 2$, $q_2 = 3$, $\dots$, q_n zur Basis g modulo p.

Beispiel: Wir betrachten unser obiges Beispiel weiter, also $p = 10009$, $g = 11$, $n = 3$. Die gefundenen Relationen liefern folgende (erweiterte) Matrix

$$\begin{pmatrix} 11 & 1 & 0 & 5140 \\ 2 & 1 & 1 & 3438 \\ 4 & 2 & 2 & 6876 \\ 2 & 3 & 0 & 4374 \\ 3 & 5 & 0 & 3620 \\ 1 & 0 & 2 & 5626 \end{pmatrix}$$

Wir bestimmen mit SAGE die hermiteschen Normalform der Matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 1002 \\ 0 & 1 & 0 & 4126 \\ 0 & 0 & 1 & 7316 \\ 0 & 0 & 0 & 10008 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Wir lesen nun dies wieder als Gleichungssystem (modulo $p - 1$) und erhalten

$$\log_{11}(2) = \ell_1 = 1002, \quad \log_{11}(3) = \ell_2 = 4126, \quad \log_{11}(5) = \ell_3 = 7316.$$

Bemerkung zum Beispiel: Hätten wir uns mit 4 Relationen begnügt, so hätten wir die Matrix

$$\begin{pmatrix} 11 & 1 & 0 & 5140 \\ 2 & 1 & 1 & 3438 \\ 4 & 2 & 2 & 6876 \\ 2 & 3 & 0 & 4374 \end{pmatrix}$$

erhalten. Die hermitesche Normalform der Matrix ist:

$$\begin{pmatrix} 1 & 0 & 24 & 36474 \\ 0 & 1 & 15 & 23794 \\ 0 & 0 & 31 & 46652 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Die zugehörigen Gleichungen sind

$$\ell_1 + 24\ell_3 \equiv 36474 \bmod (p-1), \quad \ell_2 + 15\ell_3 \equiv 23794 \bmod (p-1),$$

$$31\ell_3 \equiv 46652 \bmod (p-1).$$

Da 31 invertierbar modulo $p - 1 = 10008 = 2^3 \cdot 3^2 \cdot 139$ ist, könnte man auch hier ℓ_3, dann damit ℓ_1, ℓ_2 berechnen.

Beispiel: Wir betrachten $p = 1234567891$ mit $g = 3$ und $n = 4$. Wir finden folgende Relationen:

$$\begin{pmatrix} 10 & 5 & 1 & 1 & 782539446 \\ 2 & 6 & 0 & 3 & 902598867 \\ 4 & 7 & 3 & 0 & 996720001 \\ 5 & 0 & 5 & 1 & 923968584 \\ 0 & 11 & 2 & 2 & 779973373 \\ 3 & 11 & 1 & 2 & 905905412 \\ 21 & 1 & 0 & 1 & 234859377 \\ 8 & 7 & 0 & 1 & 94911162 \end{pmatrix}$$

Die hermitesche Normalform ist

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1150366377 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 856031312 \\ 0 & 0 & 0 & 1 & 768523259 \\ 0 & 0 & 0 & 0 & 1234567890 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Daraus lesen wir ab:

$$\log_3(2) = 1150366377, \quad \log_3(3) = 1, \quad \log_3(5) = 856031312, \quad \log_3(7) = 768523259.$$

3. Schritt: Wir können jetzt die Gleichung $g^x \equiv a \bmod p$ für verschiedene a's (leicht) lösen.
Wir wählen wie im 1. Schritt zufällig b, bis wir eine Relation

$$ag^b \equiv \prod_{j=1}^{n} q_j^{e_j} \bmod p$$

erhalten. Dann ist nämlich

$$ag^b \equiv g^{\sum_{j=1}^{n} e_j \ell_j} \bmod p$$

und damit

$$\log_g(a) = b - \sum_{j=1}^{n} e_j \ell_j \bmod (p-1).$$

Beispiel: Wir betrachten wieder $p = 10009$ mit $g = 11$, wobei wir jetzt schon

$$\log(2) = 1002, \quad \log(3) = 4126, \quad \log(5) = 7316$$

kennen. Wir berechnen einige Logarithmen:

a	nach wieviel Schritten	b	e_1	e_2	e_3	$\log(a)$
101	6	6373	0	1	1	5069
1001	145	3438	1	1	0	1690
10001	57	1238	0	0	4	8010

Bemerkungen:

- Eine wesentliche Vorarbeit in der Index-Calculus-Methode ist der 1. Schritt. Man muss n, die Anzahl der Basisprimzahlen, geeignet wählen.
  - Ist n zu klein, findet man schwer Relationen.
  - Ist n zu groß, muss man viele Relationen erzeugen, außerdem wird das Gleichungssystem im 2. Schritt schwierig.
- Hat man die ersten beiden Schritte erledigt, lassen sich diskrete Logarithmen bei festem p und g schnell finden. Dies ist anders als bei den anderen Verfahren.

Beispiel: $p = 10^{20} - 27 = 99999999999999999973$ ist eine 20-stellige Primzahl mit

$$p - 1 = 2^2 \cdot 3^2 \cdot 2777777777777777.$$

Kleinste Primitivwurzel modulo p ist $g = 7$. Wir haben $n = 100$ gewählt, also $q_1 = 2, \ldots, q_{100} = 541$. In ein paar Stunden hatten wir 200 Relationen. Die Bestimmung der hermiteschen Normalform mit SAGE ging ganz schnell. Dann konnten wir die Logarithmen der ersten 100 Primzahlen direkt ablesen:

q_i	ℓ_i	q_i	ℓ_i	q_i	ℓ_i	q_i	ℓ_i
2	97813729651897938099	101	38816803684993835495	233	45946453404202386269	383	41928176267284746217
3	79716573511464378732	103	75899545675585110251	239	30810937529061062143	389	55889276365799548055
5	99143756374821718659	107	82875437503002182038	241	15642080741805829904	397	97929661550394571289
7	1	109	3297183857295729343	251	36814249397101671748	401	45648216169628604450
11	85171810750252824683	113	44616056415023365167	257	1585576539952542120	409	82444187936494970551
13	55644183776730916627	127	236221137261058018	263	83590593663866337872	419	97133413317370576152
17	96691561306980875229	131	73185860385174843088	269	38010699860086585363	421	94996803927193055704
19	55739184273456481936	137	49407735163329994524	271	94476314312783833300	431	83766434423817079718
23	87074234668116865451	139	93769794006808996144	277	11601192250058823515	433	22277157572244754238
29	77545050555048968508	149	87438255261222732778	281	17645959163619489531	439	91245647295659168988
31	88242151415535381121	151	38168822190600906255	283	93970754695577483821	443	59556442806245068183
37	44122675618012964818	157	57225443322873802749	293	93673370812638660539	449	90229100330218247979
41	19760226622058060853	163	72495682565417102202	307	43574087353162929982	457	56039335326190136816
43	43786279493065638113	167	57606389293931982736	311	70147616037551849853	461	73026887718254562722
47	72774307713555535371	173	98219309882693480649	313	61881063747783795601	463	41609551435366400499
53	89787456290256070737	179	77134612496288529845	317	98809325488010602894	467	47348147607842680321
59	69800780233256409927	181	40654370701323587166	331	93563765867921028171	479	16422253961801807293
61	99604446958933693350	191	35164103231367296957	337	777499277234052123	487	31402511922453335709
67	75222888617350428740	193	28862311291728531752	347	53563598782030398291	491	93243295192635188362
71	45014310482982352679	197	72346437416474178719	349	5994436119122083793	499	68193684919182480508
73	59067872661957387651	199	4518648436168144236	353	63660235771380877067	503	55919002540331368733
79	81702549335666538777	211	22426029497797535457	359	37913712704366431430	509	70832965232573844604
83	37765146505223434350	223	13795490779108271707	367	4267248552283825916	521	87445214133162135723
89	26647581286393089427	227	40894658515160067495	373	55630269320712364984	523	52019643536217084589

Bemerkungen: (Die nachfolgenden Aussagen geben eventuell nicht den neuesten Stand wieder.)

- Läßt sich die Laufzeit eines Algorithmus (in Abhängigkeit von p) durch
$$L_p[\alpha, c] = O(e^{c(\ln p)^{\alpha}(\ln \ln p)^{1-\alpha}})$$
mit Konstanten $c > 0$ und $0 < \alpha < 1$ abschätzen, so sagt man, der Algorithmus hat subexponentielle Laufzeit. (Es gibt etwas voneinander abweichende Definitionen der Funktion $L_p[\alpha, c]$.) Für $\alpha = 0$ hat man
$$L_p[0, c] = O((\ln p)^c),$$
d.h. eine polynomiale Laufzeit, für $\alpha = 1$
$$L_p[1, c] = O(p^c),$$
also eine exponentielle Laufzeit. Für $0 < \alpha < 1$ gilt
$$L_p[\alpha, c] = o(p^{\epsilon})$$
für jedes $\epsilon > 0$, was auch den Namen subexponentiell erklärt.
- Man kann zeigen [McCurley, p.63], dass sich das Index-Calculus-Verfahren so implementieren läßt, dass die Laufzeit für die ersten beiden Schritte bei Wahl von $q_n \approx e^{\frac{1}{2}\sqrt{\ln p \ln \ln p}}$ durch
$$L_p[\frac{1}{2}, 2 + o(1)] = O(e^{(2+o(1))\sqrt{\ln p \ln \ln p}})$$
abgeschätzt werden kann.
- Das Index-Calculus-Verfahren hat also subexponentielle Laufzeit im Unterschied zu den anderen allgemeinen (hier vorgestellten) Verfahren. Das schnellste zur Zeit bekannte Verfahren ist eine Verallgemeinerung der Index-Calculus-Methode und verwendet das Zahlkörpersieb; es hat eine Laufzeit von $L_p[\frac{1}{3}, c]$.