KAPITEL 10

Methoden zur Berechnung diskreter Logarithmen

Das Kapitel muss noch überarbeitet werden.

1. Einführung

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

Bemerkung: Ist p eine Primzahl, $g \in \mathbb{N}$ mit $\mathrm{ggT}(p, g) = 1$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(p, a) = 1$, 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, $g \in \mathbb{N}$ 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. In diesem Kapitel sollen weitere Methoden vorgestellt werden.

2. Logarithmenberechnung mit der Pollardschen ρ-Methode

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

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

- 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.

2. Schritt:

- 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.

Algorithmus 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
$x, e, f, \widetilde{x}, \widetilde{e}, \widetilde{f} \leftarrow 1, 0, 0, 1, 0, 0$
for $i = 1, \ldots, 10\lfloor\sqrt{p}\rfloor$ **do**
  $x, e, f \leftarrow S(x, e, f)$
  $\widetilde{x}, \widetilde{e}, \widetilde{f} \leftarrow S(\widetilde{x}, \widetilde{e}, \widetilde{f})$
  $\widetilde{x}, \widetilde{e}, \widetilde{f} \leftarrow S(\widetilde{x}, \widetilde{e}, \widetilde{f})$
  if $x = \widetilde{x}$ **then**
    $s, t \leftarrow (e - \widetilde{e}) \bmod (p-1), (\widetilde{f} - f) \bmod (p-1)$
    return s, t
  end if
end for

- 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.

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

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

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:

(1) Das Verfahren ist nicht deterministisch, die Laufzeit ist $O(p^{\frac{1}{2}+\varepsilon})$. Ein Vorteil ist, dass man praktisch keinen Speicherplatz braucht.
(2) Wie sollte man S_1, S_2, S_3 wählen?
(3) 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. 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 $\mathrm{ord}_p(g) = p-1$. Diese Methode ist auch unter dem Namen Silver-Pohlig-Hellman-Methode bekannt.

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

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

die Primfaktorzerlegung von $\mathrm{ord}_p(g)$. Aus schreibtechnischen Gründen setzen wir $u_i = q_i^{e_i}$ und haben dann

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

mit paarweise teilerfremden natürlichen Zahlen $u_1, \dots, u_r$. Sei $a \in \mathbb{Z}$ mit $\mathrm{ggT}(p,a) = 1$, sodass gilt

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

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

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

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

x_i ist durch diese Bedingungen eindeutig bestimmt.

(2) *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 $x_1, \ldots, x_r$, so findet man eine Lösung der Gleichung $g^x \equiv a \bmod p$ mit dem chinesischen Restsatz.

Beweis:

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

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

(1) Dividiert man ℓ durch u_i, so erhält man Zahlen $v_i \in \mathbb{N}_0$ und $x_i \in \mathbb{N}_0$ mit

$$\ell = v_i u_i + x_i \text{ und } 0 \le x_i \le u_i - 1.$$

Dann gilt

$$a^{\frac{\operatorname{ord}_p(g)}{u_i}} \equiv g^{\frac{\operatorname{ord}_p(g)}{u_i}\ell} \equiv g^{\frac{\operatorname{ord}_p(g)}{u_i}(v_i u_i + x_i)} \equiv g^{\operatorname{ord}_p(g) v_i} \cdot g^{\frac{\operatorname{ord}_p(g)}{u_i} x_i} \equiv (g^{\frac{\operatorname{ord}_p(g)}{u_i}})^{x_i} \bmod p,$$

also erfüllt x_i die angegebenen Bedingungen. Wegen

$$\operatorname{ord}_p\left(g^{\frac{\operatorname{ord}_p(g)}{u_i}}\right) = u_i$$

ist x_i eindeutig bestimmt modulo u_i, was auch die zweite Behauptung zeigt.

(2) Wegen (1) gilt $x_i \equiv \ell \bmod u_i$, sodass wir erhalten:

$$\begin{aligned} g^x \equiv a \bmod p \quad &\Longleftrightarrow \quad g^x \equiv g^\ell \bmod p \quad \Longleftrightarrow \quad x \equiv \ell \bmod \operatorname{ord}_p(g) \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad x \equiv \ell \bmod u_i \text{ für } i = 1, \ldots, r \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad x \equiv x_i \bmod u_i \text{ für } i = 1, \ldots, r, \end{aligned}$$

wie behauptet. ■

Algorithmus 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 $\operatorname{ord}_p(g)$

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

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** $i = 1, \ldots, r$ **do**
2: $\quad u_i \leftarrow q_i^{e_i}$.
3: $\quad$ Bestimme ein $x_i \in \{0, 1, \ldots, u_i - 1\}$ 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 $i = 1, \ldots, r$. (Man kann dabei $0 \le x \le \operatorname{ord}_p(g)$ 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 und durch Probieren die zugehörigen Lösungen

$$\begin{aligned} 2^{\frac{p-1}{2} x_1} &\equiv 3^{\frac{p-1}{2}} \bmod p \qquad x_1 \equiv 0 \bmod 2, \\ 2^{\frac{p-1}{131} x_2} &\equiv 3^{\frac{p-1}{131}} \bmod p \qquad x_2 \equiv 92 \bmod 131, \\ 2^{\frac{p-1}{521} x_3} &\equiv 3^{\frac{p-1}{521}} \bmod p \qquad x_3 \equiv 223 \bmod 521, \\ 2^{\frac{p-1}{73259} x_4} &\equiv 3^{\frac{p-1}{73259}} \bmod p \qquad x_4 \equiv 55292 \bmod 73259. \end{aligned}$$

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:

(1) 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.
(2) Kann man $\operatorname{ord}_p(g)$ nicht bestimmen, dann man das Verfahren auch nicht anwenden.

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

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 $\operatorname{ggT}(p,g) = 1$ und $\operatorname{ord}_p(g) = q^e$ mit einer Primzahl q und $e \in \mathbb{N}$. Ist $a \in \mathbb{Z}$ mit $\operatorname{ggT}(p,a) = 1$ 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

$$(g^{q^{e-1}})^{x_0} = a^{q^{e-1}}, (g^{q^{e-1}})^{x_1} = \left(a \cdot (g^{-1})^{x_0}\right)^{q^{e-2}}, \quad \ldots \quad (g^{q^{e-1}})^{x_{e-1}} = \left(a \cdot (g^{-1})^{x_0 + x_1 q + \cdots + x_{e-3} q^{e-3} + x_{e-2} q^{e-2}}\right).$$

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:

(1) 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.

(2) Es gilt für $0 \le i \le e-1$

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

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.

(3) 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

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

so gilt

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

und

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

Damit erhält man sofort nachfolgenden Algorithmus.

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 $e \ge 2$ und q^e größer, so ist dies von großem Vorteil.

Beispiel: Wir betrachten folgende Situation:

$$\begin{aligned} p &= 702857981670509306221075879002735600669348806920438223719507534928032696914411 53, \\ g &= 605583059411022095567350578112877881821659500487969573336424154879476717760740 6, \\ a &= 169105058115399344398468379360382833774642593911657935895190296660759243721972 43. \end{aligned}$$

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}:

$$\operatorname{ord}_p(g) = 115792089237316195423570985008687907853269984665640564039457584007913129639936.$$

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:

$$x = 43493391421557323879794069153474181893360315816869636864652584353009783638222.$$

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

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

(1) Existiert eine Lösung x der Gleichung $g^x \equiv a \bmod p$, so kann man o.E. $0 \le x \le \operatorname{ord}_p(g) - 1$ annehmen.

(2) Wir wählen eine natürliche Zahl m mit $m^2 \ge \operatorname{ord}_p(g)$. Wenn wir $\operatorname{ord}_p(g)$ nicht kennen, wählen wir $m^2 \ge p - 1$.

(3) Jede ganze Zahl x mit $0 \le x \le m^2 - 1$ hat eine eindeutige Zerlegung

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

Dabei ist

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

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

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

(5) Umgekehrt: Finden wir i, j mit $0 \le i, j \le m - 1$ und

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

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

(6) Wir definieren bzw. berechnen jetzt zwei Listen:

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

und

$$\widetilde{G}_j = ((a \cdot (g^{-m})^j \bmod p), j) \quad \text{für} \quad j = 0, 1, \ldots, m - 1.$$

Die Elemente $\widetilde{B}_i$ werden als baby-steps, die $\widetilde{G}_j$ als giant-steps bezeichnet. Finden wir zwei Elemente $\widetilde{B}_i$ und $\widetilde{G}_j$ mit gleicher erster Komponente, so ist $x = i + mj$ der gesuchte diskrete Logarithmus.

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 $m^2 \ge \operatorname{ord}_p(g)$. Wir berechnen zunächst

$$\widetilde{B}_i = ((g^i \bmod p), i) \quad \text{für} \quad i = 0, \ldots, 9$$

und erhalten

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

Für

$$\widetilde{G}_j = ((a(g^{-m})^j \bmod p), j) \quad \text{für} \quad j = 0, \ldots, 9$$

finden wir

$$(3, 0), (94, 1), (50, 2), (18, 3), (59, 4), (98, 5), (7, 6), (51, 7), (83, 8), (42, 9).$$

Für $i = 9$ und $j = 6$ ergibt sich die gleiche erste Komponente 7. Daher ist $x = 9 + 6 \cdot 10 = 69$ der diskrete Logarithmus von 3 zur Basis 2 modulo 101.

Fortsetzung der Überlegungen:

(7) 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.

(8) Eine Möglichkeit wäre, Hashtabellen zu benutzen, die in Python3 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 $j = 0,1,2,\ldots,m-1$

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

- 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 wegen 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.

Bemerkungen:

(1) 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 $O(p^{\frac{1}{2}+\varepsilon})$ berechnen kann.

(2) Die baby-step-giant-step-Methode wird auch an anderen Stellen in der Zahlentheorie verwendet. Die Idee geht auf Shanks zurück.

Beispiele:

(1) $p = 211$, $g = 2$, $a = 3$, $m = 15$

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

$\widetilde{g} = 67$

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

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

(2) $p = 241$, $g = 7$, $a = 2$, $m = 16$

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

$\widetilde{g} = 183$

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

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

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

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

Eingabe: p, g, a, eventuell eine obere Schranke $m_{\mathrm{ord}_p(g)}$ für $\mathrm{ord}_p(g)$
Ausgabe: x mit $g^x \equiv a \bmod p$

Ist $m_{\mathrm{ord}_p(g)}$ nicht gegeben, setze $m_{\mathrm{ord}_p(g)} \leftarrow p-1$
$m \leftarrow \lceil \sqrt{m_{\mathrm{ord}_p(g)}} \rceil$
Lege eine Hashtabelle/dictionary B an mit $B(1) = 0$.
$g_i, i \leftarrow 1, 0$
while $i < m-1$ **do**
    $g_i, i \leftarrow (g_i g) \bmod p, i+1$
    $B(g_i) = i$
end while
$\widetilde{g}, h_j \leftarrow g^{p-1-m} \bmod p, a$
for $j = 0, \ldots, m-1$ **do**
    if h_j ist im Definitionsbereich von B **then**
        $i = B(h_j)$
        $x \leftarrow i + mj$
        return x
    end if
    $h_j \leftarrow (h_j \widetilde{g}) \bmod p$
end for
return Der diskrete Logarithmus existiert nicht oder $m_{\mathrm{ord}_p(g)}$ wurde zu klein gewählt.

(3) $p = 241$, $g = 2$, $a = 7$, $m = 16$

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

$\widetilde{g} = 15$

$$h_0 = 7, \quad h_1 = 105, \quad h_2 = 129, \quad h_3 = 7, \quad h_4 = 105, \quad h_5 = 129,$$
$$h_6 = 7, \quad h_7 = 105, \quad h_8 = 129, \quad h_9 = 7, \quad h_{10} = 105, \quad h_{11} = 129,$$
$$h_{12} = 7, \quad h_{13} = 105, \quad h_{14} = 129, \quad h_{15} = 7.$$

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

5. Die Index-Calculus-Methode

Sei p eine ungerade Primzahl und g eine Primitivwurzel modulo p. Wir wollen die Gleichung $g^x = a$ in $\mathbb{F}_p^*$ 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:

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

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

(4) Ist $\widetilde{x} > 1$, fängt man von vorne an. Ist $\widetilde{x} = 1$, 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

(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

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

Dies ist ein lineares Gleichungssystem über dem Ring $\mathbb{Z}/(p-1)$ mit den Unbekannten $\ell_1, \ldots, \ell_n$. Wie löst man ein solches Gleichungssystem? Wir geben eine Möglichkeit an mit einem modifizierten Gaußschen Eliminationsverfahren: Wir schreiben a_{ij} und b_i in eine große Matrix und wenden folgende Zeilenoperationen an:

- Wir lassen k von 1 bis n, dann jeweils i von $k+1$ bis m laufen und tun dann folgendes, falls $a_{ik} \neq 0$:
- Wir berechnen mit dem euklidischen Algorithmus $d = \mathrm{ggT}(a_{ij}, a_{kk})$ und $d = sa_{ik} + ta_{kk}$ mit $s, t \in \mathbb{Z}$. Nun machen wir folgende Zeilenumformungen

$$\begin{aligned} \text{neue Zeile } k &= s(\text{Zeile } i) + t(\text{Zeile } k), \\ \text{neue Zeile } i &= \frac{a_{kk}}{d}(\text{Zeile } i) - \frac{a_{ik}}{d}(\text{Zeile } k). \end{aligned}$$

  Diese Zeilentransformation ist umkehrbar, da die Transformationsmatrix Determinante 1 hat, ändert also nichts an der Lösungsmenge. Für die neue Matrix gilt dann $a_{kk} = d$ und $a_{ik} = 0$.
- Wenn wir so alle $k = 1, \ldots, n$ und $i = k+1, \ldots, m$ durchlaufen haben, haben wir eine Dreiecksmatrix.
- Nun gilt für $i = 1, \ldots, n$

$$b_i \equiv a_{ii}\ell_i + \sum_{j=i+1} a_{ij}\ell_j \quad \text{bzw.} \quad a_{ii}\ell_i \equiv b_i - \sum_{j=i+1} a_{ij}\ell_j.$$

  Damit kann man rekursiv ℓ_n, ℓ_{n-1}, ... berechnen, wenn $\mathrm{ggT}(a_{ii}, p-1) = 1$ gilt.
- Gilt $\mathrm{ggT}(a_{11} \ldots a_{nn}, p-1) = 1$, so kann man also das Gleichungssystem eindeutig lösen. Andernfalls braucht man noch mehr Relationen, d.h. ein größeres m. Wir gehen also zurück zum 1. Schritt und erzeugen noch einige Relationen.

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

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

die wir gemäß obigem Schema umformen wollen. Für $k = 1$, $i = 2$ ist $d = \mathrm{ggT}(11, 2) = (-5) \cdot 2 + 1 \cdot 11$. Wir ersetzen also Zeile 1 durch $(-5)\cdot$ Zeile 2 $+1\cdot$ Zeile 1, Zeile 2 durch $11\cdot$ Zeile 2 $-2\cdot$ Zeile 1, was dann folgende Matrix ergibt:

$$\begin{pmatrix} 1 & -4 & -5 & -12050 \\ 0 & 9 & 11 & 27538 \\ 4 & 2 & 2 & 6876 \\ 2 & 3 & 0 & 4374 \end{pmatrix}$$

Man man auf diese Weise weiter, kommt man schließlich auf folgende Matrix

$$\begin{pmatrix} 1 & -4 & -5 & -12050 \\ 0 & 1 & 15 & 23794 \\ 0 & 0 & 31 & 46652 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Damit kann man dann sofort berechnen:

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

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

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

und damit

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

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:

(1) 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.
(2) 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.
(3) Wir haben zu dem Verfahren ein Programm dlic1.c geschrieben, mit dem die nachfolgenden Beispiele gerechnet wurden.
(4) In einem leicht veränderten Programm dlic2.c wurden die b's nicht zufällig gewählt, sondern der Reihe nach $b = 1, 2, \ldots$ (mit einer leichten Modifikation: Ist $x \equiv g^b \bmod p$, $x = q_1^{e_1} \ldots q_n^{e_n}$ und g ein Produkt aus q_i's, so liefert g^{b+i} keine neue Relation, solange $xg^i < p$ gilt.)

Beispiel: Wir wählen $p = 10^{13} + 37$, $g = 2$ und führen für verschiedene n's die ersten beiden Schritte durch, d.h. wir berechnen $\log_2(2), \ldots, \log_2(p_n)$ mit dlic1.c (links) bzw. dlic2.c (rechts):

n	benötigte Relationen/n	Zeit
50	1.12	28 sec
60	1.57	30 sec
70	1.83	31 sec
80	1.79	25 sec
90	1.44	20 sec
100	1.67	22 sec
200	2.94	68 sec

n	benötigte Relationen/n	Zeit
50	1.02	12 sec
60	1.02	10 sec
70	1.10	8 sec
80	1.45	11 sec
90	1.33	10 sec
100	1.60	12 sec
200	2.45	46 sec

Die ersten Logarithmen sind:

$$\log_2(3) = 1799874854241, \quad \log_2(5) = 9904820564071, \quad \log_2(7) = 6834622192483.$$

Beispiel: $p = 10^{18} + 3$, $g = 2$, $n = 100$. Nach 1656 sec und 154 Relationen mit dlic2.c (bzw. 2294 sec und 103 Relationen mit dlic1.c) war die Berechnung beendet. Beispielhaft sind hier die ersten Logarithmen:

$$\log_2(3) = 69083101039644759, \quad \log_2(5) = 364949061168869153, \quad \log_2(7) = 682412085515776013.$$

Beispiel: $p = 10^{20} + 39$, $g = 3$. Bei Wahl von $n = 600$ wurden 1515 Relationen und 5428 sec mit dlic2.c (bzw. 1579 Relationen und 6224 sec mit dlic1.c) benötigt. Die ersten Logarithmen sind:

a	$\log_3(a)$
2	86792332251783168850
3	1
5	41025259664069146848
7	10465869232100529600
11	97981806285588131496
13	6351838317046312990
17	22945125767903614204
19	27629884555029366917
23	92869052686674421326
29	58629941811305283781

Bemerkungen:

(1) 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

$$L_p[\alpha, c] = o(p^\varepsilon)$$

für jedes $\varepsilon > 0$, was auch den Namen subexponentiell erklärt.

(2) Man kann zeigen [**?**, 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.

(3) 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 $L_p[\frac{1}{3}, c]$.