KAPITEL 7

Kryptographische Anwendungen diskreter Logarithmen

1. Die multiplikative Ordnung einer ganzen Zahl a modulo einer natürlichen Zahl n

Ist $n \in \mathbb{N}$, $a \in \mathbb{Z}$ mit $\mathrm{ggT}(n,a) = 1$, so gilt $a^{\varphi(n)} \equiv 1 \bmod n$, also ist folgende Definition sinnvoll:

$$\mathrm{ord}_n(a) = \min\{k \in \mathbb{N} : a^k \equiv 1 \bmod n\}.$$

$\mathrm{ord}_n(a)$ wird als **Ordnung** von a modulo n bezeichnet.

Beispiel: Sei $n = 10$ und $a = 3$. Wegen

$$3^1 \equiv 3 \bmod 10, \quad 3^2 \equiv 9 \bmod 10, \quad 3^3 \equiv 7 \bmod 10, \equiv 3^4 \equiv 1 \bmod 10$$

gilt

$$\mathrm{ord}_{10}(3) = 4.$$

SATZ. *Sei $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(n,a) = 1$. Für $k, k_1, k_2 \in \mathbb{N}_0$ gilt dann:*

(1) $a^k \equiv 1 \bmod n \iff \mathrm{ord}_n(a) \mid k$.
(2) $\{k \in \mathbb{N} : a^k \equiv 1 \bmod n\} = \mathbb{N} \cdot \mathrm{ord}_n(a) = \{\ell \cdot \mathrm{ord}_n(a) : \ell \in \mathbb{N}\}$.
(3) $\mathrm{ord}_n(a) \mid \varphi(n)$.
(4) $a^{k_1} \equiv a^{k_2} \bmod n \iff k_1 \equiv k_2 \bmod \mathrm{ord}_n(a)$.

Beweis:

(1) $\Longrightarrow$ Sei $a^k \equiv 1 \bmod n$. Wir dividieren k durch $\mathrm{ord}_n(a)$ und erhalten eine Darstellung

$$k = q \cdot \mathrm{ord}_n(a) + r \text{ mit } q, r \in \mathbb{N}_0, 0 \le r \le \mathrm{ord}_n(a) - 1.$$

Es folgt

$$1 \equiv a^k \equiv a^{q \cdot \mathrm{ord}_n(a) + r} \equiv \left(a^{\mathrm{ord}_n(a)}\right)^q \cdot a^r \equiv a^r \bmod n.$$

Nach Definition von $\mathrm{ord}_n(a)$ muss $r = 0$ und damit $\mathrm{ord}_n(a) \mid k$ gelten.
$\Longleftarrow$ Es gelte $\mathrm{ord}_n(a) \mid k$. Dann gibt es ein $q \in \mathbb{N}_0$ mit $k = q \cdot \mathrm{ord}_n(a)$. Es folgt

$$a^k \equiv a^{q \cdot \mathrm{ord}_n(a)} \equiv \left(a^{\mathrm{ord}_n(a)}\right)^q \equiv 1^q \equiv 1 \bmod n.$$

(2) Dies ist nur eine Umformulierung von (1).
(3) Nach dem Satz von Euler gilt $a^{\varphi(n)} \equiv 1 \bmod n$. Die Behauptung folgt dann aus (1).
(4) O.E. können wir $k_1 < k_2$ annehmen. Dann folgt mit $\mathrm{ggT}(n,a) = 1$:

$$\begin{aligned} a^{k_1} \equiv a^{k_2} \bmod n &\iff n \mid a^{k_2} - a^{k_1} \iff n \mid a^{k_1} \cdot (a^{k_2 - k_1} - 1) \iff \\ &\iff n \mid a^{k_2 - k_1} - 1 \iff a^{k_2 - k_1} \equiv 1 \bmod n \iff \\ &\iff \mathrm{ord}_k(a) \mid k_2 - k_1 \iff k_1 \equiv k_2 \bmod \mathrm{ord}_n(a). \end{aligned}$$

Dies war zu zeigen. ■

Bemerkung: Die Aussage (4) des Satzes besagt, dass wir im Exponenten modulo $\mathrm{ord}_n(a)$ rechnen dürfen, wenn wir in der Basis modulo n rechnen und $\mathrm{ggT}(n,a) = 1$ gilt.

Der folgende Satz enthält weitere Eigenschaften der Ordnung:

SATZ. *Sei $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(n,a) = 1$.*

(1) *Für $k \in \mathbb{N}$ gilt*

$$\operatorname{ord}_n(a^k) = \frac{\operatorname{ord}_n(a)}{\operatorname{ggT}(\operatorname{ord}_n(a), k)}.$$

(2) *Ist $d \in \mathbb{N}$ mit $d \mid \operatorname{ord}_n(a)$, so hat*

$$a^{\frac{\operatorname{ord}_n(a)}{d}}$$

Ordnung d.

Beweis:

(1) Für $\ell \in \mathbb{N}$ gilt die Äquivalenz:

$$\begin{aligned}
\operatorname{ord}_n(a^k) \mid \ell \quad &\iff \quad (a^k)^\ell \equiv 1 \bmod n \quad \iff \quad a^{k\ell} \equiv 1 \bmod n \quad \iff \\
&\iff \quad \operatorname{ord}_n(a) \mid k\ell \quad \iff \\
&\iff \quad \frac{\operatorname{ord}_n(a)}{\operatorname{ggT}(\operatorname{ord}_n(a), k)} \mid \frac{k}{\operatorname{ggT}(\operatorname{ord}_n(a), k)} \cdot \ell \quad \iff \\
&\iff \quad \frac{\operatorname{ord}_n(a)}{\operatorname{ggT}(\operatorname{ord}_n(a), k)} \mid \ell.
\end{aligned}$$

Daraus folgt die Behauptung.

(2) Dies folgt sofort aus (1) mit $k = \frac{\operatorname{ord}_n(a)}{d}$. ■

Exkurs: Zur Funktion $\omega_{N,m} : E_N \to Q_N$

Für eine RSA-Zahl N hatten wir definieren

$$E_N = \{a \in \{0, 1, \ldots, N-1\} : \operatorname{ggT}(N, a) = 1\} \quad \text{und} \quad Q_N = \{a \in \{0, 1, \ldots, N-1\} : a^2 \equiv 1 \bmod N\}.$$

E_N ist also ein Repräsentantensystem der Einheitengruppe $(\mathbb{Z}/N\mathbb{Z})^*$ und Q_N sind die Wurzeln aus 1 modulo N.
Ist $m \in \mathbb{N}$ gegeben mit

$$a^m \equiv 1 \bmod N \text{ für alle } a \in E_N,$$

zerlegt man $m = 2^\ell u$ mit $u \equiv 1 \bmod 2$, so wurde $\omega_{N,m} : E_N \to Q_N$ durch

$$\omega_{N,m}(a) = \begin{cases} 1, & \text{falls } a^u \equiv 1 \bmod N, \\ (a^{2^i u} \bmod N), & \text{falls } a^{2^i u} \not\equiv 1 \bmod N \text{ und } a^{2^{i+1} u} \equiv 1 \bmod N \text{ für ein } i \in \{0, 1, \ldots, \ell - 1\} \end{cases}$$

definiert.

Wir wollen zeigen, dass gilt

$$\omega_{N,m}(a) = \begin{cases} 1, & \text{falls } \operatorname{ord}_N(a) \equiv 1 \bmod 2, \\ a^{\frac{\operatorname{ord}_N(a)}{2}} \bmod N, & \text{falls } \operatorname{ord}_N(a) \equiv 0 \bmod 2. \end{cases}$$

Beweis:

- **Fall** $\operatorname{ord}_N(a)$ **ungerade:** Es ist $a^m \equiv 1 \bmod N$, woraus $\operatorname{ord}_N(a) \mid m$, also $\operatorname{ord}_N(a) \mid 2^\ell u$ folgt. Da $\operatorname{ord}_N(a)$ als ungerade vorausgesetzt war, ergibt sich $\operatorname{ord}_N(a) \mid u$ und damit $a^u \equiv 1 \bmod N$, also $\omega_{N,m}(a) = 1$, wie behauptet.
- **Fall** $\operatorname{ord}_N(a)$ **gerade:** Wäre $a^u \equiv 1 \bmod N$, so würde $\operatorname{ord}_N(a) \mid u$ folgen, was nicht sein kann, da $\operatorname{ord}_N(a)$ gerade ist. Also sind wir im 2. Fall, d.h.

  $$\omega_{N,m}(a) = (a^{2^i u} \bmod N) \quad \text{und} \quad a^{2^i u} \not\equiv 1 \bmod N \quad \text{und} \quad a^{2^{i+1} u} \equiv 1 \bmod N.$$

  Aus $a^{2^{i+1} u} \equiv 1 \bmod N$ folgt $\operatorname{ord}_N(a) \mid 2^{i+1} u$. Wir können also zerlegen $\operatorname{ord}_N(a) = 2^x \widetilde{u}$ mit $\widetilde{u}$ ungerade udn $\widetilde{u} \mid u$, außerdem $x \leq i + 1$. Aus $a^{2^i u} \not\equiv 1 \bmod N$ folgt $\operatorname{ord}_N(a) \nmid 2^i u$, also $2^x \widetilde{u} \nmid 2^i u$, und damit $x = i + 1$. Es ist also

  $$\operatorname{ord}_N(a) = 2^{i+1} \widetilde{u} \text{ mit } \widetilde{u} \mid u.$$

Nun gilt:

$$\begin{aligned} a^{2^i u} \equiv a^{\frac{\operatorname{ord}_N(a)}{2}} \bmod N \quad &\Longleftrightarrow \quad 2^i u \equiv \frac{\operatorname{ord}_N(a)}{2} \bmod \operatorname{ord}_N(a) \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad 2^i u \equiv 2^i \widetilde{u} \bmod 2^{i+1}\widetilde{u} \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad 2^i u \equiv 2^i \widetilde{u} \bmod 2^{i+1} \text{ und } 2^i u \equiv 2^i \widetilde{u} \bmod \widetilde{u} \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad u \equiv \widetilde{u} \bmod 2. \end{aligned}$$

Die letzte Aussage stimmt, woraus dann die Behauptung folgt. ■

Die bewiesene Aussage zeigt, dass die Funktion $\omega_{N,m}$ nicht von der Wahl von m abhängt.

2. Wie bestimmt man $\operatorname{ord}_n(a)$?

Ist n klein oder die Ordnung $\operatorname{ord}_n(a)$ vermutlich klein, kann man naiv die Ordnung bestimmen:

Algorithmus Bestimmung von $\operatorname{ord}_n(a)$ - naive Methode

Eingabe: $n \in \mathbb{N}$, $a \in \mathbb{Z}$ mit $\operatorname{ggT}(n,a) = 1$
Ausgabe: $\operatorname{ord}_n(a)$

```
for k = 1, ..., n do
    if a^k ≡ 1 mod n then          ▷ Wegen φ(n) ≤ n ist als obere Schranke n ausreichend
        return k (als ord_n(a))
    end if
end for
```

Das nachfolgende Verfahren beruht auf folgender Überlegung: Sei $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit $\operatorname{ggT}(n,a) = 1$.

- Wir nehmen an, wir kennen eine Zahl $m \in \mathbb{N}$ mit
$$a^m \equiv 1 \bmod n.$$
(Im Fall einer Primzahl $n = p$ hat $m = p - 1$ diese Eigenschaft.) Dann gilt
$$\operatorname{ord}_n(a) \mid m.$$
Die Teiler einer großen Zahl m zu bestimmen ist aber nicht harmlos.
- Wir nehmen an, wir können die Primfaktorzerlegung von m bestimmen:
$$m = q_1^{e_1} \dots q_r^{e_r}.$$
Wegen $\operatorname{ord}_n(a) \mid m$ entsteht $\operatorname{ord}_n(a)$ aus m durch Herausteilen von Primteilern q_i. Das nachfolgende Lemma beschreibt den mathematischen Sachverhalt.

LEMMA. *Sei $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit $\operatorname{ggT}(n,a) = 1$. Dazu sei eine Zahl $m \in \mathbb{N}$ gegeben mit folgenden Eigenschaften:*

- $a^m \equiv 1 \bmod n$.
- *m habe die Primfaktorzerlegung $m = q_1^{e_1} \dots q_r^{e_r}$.*

(1) *Sei $\widetilde{m} \in \mathbb{N}$ mit $\widetilde{m} \mid m$ und $a^{\widetilde{m}} \equiv 1 \bmod n$, also $\operatorname{ord}_n(a) \mid \widetilde{m}$. Sei weiter $i \in \{1, \dots, r\}$. Dann gibt es drei Fälle:*
   - *Gilt $q_i \nmid \widetilde{m}$, so gilt $v_{q_i}(\operatorname{ord}_n(a)) = v_{q_i}(\widetilde{m}) = 0$.*
   - *Gilt $q_i \mid \widetilde{m}$ und $a^{\frac{\widetilde{m}}{q_i}} \not\equiv 1 \bmod n$, so gilt $v_{q_i}(\operatorname{ord}_n(a)) = v_{q_i}(\widetilde{m})$.*
   - *Gilt $q_i \mid \widetilde{m}$ und $a^{\frac{\widetilde{m}}{q_i}} \equiv 1 \bmod n$, so gilt $\operatorname{ord}_n(a) \mid \frac{\widetilde{m}}{q_i}$.*

(2) *Ist $\widetilde{m} \in \mathbb{N}$ mit $\widetilde{m} \mid m$ und $a^{\widetilde{m}} \equiv 1 \bmod n$ und gilt für $i = 1, \dots, r$*
$$q_i \nmid \widetilde{m} \quad \text{oder} \quad a^{\frac{\widetilde{m}}{q_i}} \not\equiv 1 \bmod n,$$
so gilt
$$\operatorname{ord}_n(a) = \widetilde{m}.$$

Beweis:

(1) Die Fallunterscheidung ist klar.
- Es gelte $q_i \nmid \widetilde{m}$. Wegen $\operatorname{ord}_n(a) \mid \widetilde{m}$, folgt dann aus $q_i \nmid \widetilde{m}$ sofort $q_i \nmid \operatorname{ord}_n(a)$, also
$$v_{q_i}(\operatorname{ord}_n(a)) = v_{q_i}(\widetilde{m}).$$
- Es gelte nun $q_i \mid \widetilde{m}$ und $a^{\frac{\widetilde{m}}{q_i}} \not\equiv 1 \bmod n$. Dann gilt $\operatorname{ord}_n(a) \mid \widetilde{m}$ und $\operatorname{ord}_n(a) \nmid \frac{\widetilde{m}}{q_i}$. Die Primzahl q_i geht also in $\operatorname{ord}_n(a)$ genauso oft auf wie in $\widetilde{m}$, d.h.
$$v_{q_i}(\operatorname{ord}_n(a)) = v_{q_i}(\widetilde{m}).$$
- Es gelte nun $q_i \mid \widetilde{m}$ und $a^{\frac{\widetilde{m}}{q_i}} \equiv 1 \bmod n$. Dann gilt natürlich $\operatorname{ord}_n(a) \mid \frac{\widetilde{m}}{q_i}$.

(2) Aus (1) folgt $v_{q_i}(\operatorname{ord}_n(a)) = v_{q_i}(\widetilde{m})$ für $i = 1, \ldots, r$. Da aber sowohl $\operatorname{ord}_n(a)$ als auch $\widetilde{m}$ nur $q_1, \ldots, q_r$ als mögliche Primteiler haben, folgt $\operatorname{ord}_n(a) = \widetilde{m}$. ■

Algorithmus Bestimmung der Ordnung einer Zahl a modulo n

Eingabe: $n \in \mathbb{N}$, $a \in \mathbb{Z}$, $m \in \mathbb{N}$ mit $a^m \equiv 1 \bmod n$, $q_1, \ldots, q_r$ seien (alle) Primteiler von m
Ausgabe: $\operatorname{ord}_n(a)$

```
for i = 1, ..., r do
    while q_i | m and a^(m/q_i) ≡ 1 mod n do
        m ← m/q_i
    end while
end for
return m
```

---

3. Die multiplikative Gruppe des Körpers $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$

Bemerkung: Wir haben bisher meist die modulo-Schreibweise verwendet um Aussagen in $\mathbb{Z}/n\mathbb{Z}$ auszudrücken, d.h.
$$\overline{a} = \overline{b} \text{ in } \mathbb{Z}/n\mathbb{Z} \quad \Longleftrightarrow \quad a \equiv b \bmod n.$$
Im Folgenden werden wir im Fall $n = p$ oft Aussagen direkt in $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ angeben. Beim einfachen Rechnen muss man natürlich trotzdem modulo p rechnen.

Sei $a \in \mathbb{F}_p^*$. Die von a erzeugte Untergruppe ist
$$\langle a \rangle = \{a^k : 0 \le k \le \operatorname{ord}_p(a) - 1\} = \{1, a, a^2, \ldots, a^{\operatorname{ord}_p(a)-1}\},$$
insbesondere gilt
$$\#\langle a \rangle = \operatorname{ord}_p(a).$$
Wir wissen bereits, dass
$$\operatorname{ord}_p(a) \mid p - 1$$
gilt.
Nun lernt man in der Algebra, dass die multiplikative Gruppe eines endlichen Körpers $\mathbb{F}_p$ zyklisch ist, d.h. es gibt ein Element $g \in \mathbb{F}_p^*$ mit
$$\mathbb{F}_p^* = \{1, g, g^2, \ldots, g^{p-2}\}.$$
Äquivalent damit ist $\operatorname{ord}_p(g) = p-1$. Jedes $g \in \mathbb{F}_p^*$ mit $\operatorname{ord}_p(g) = p-1$ nennt man eine **Primitivwurzel modulo** p.

Beispiel: Für $p = 11$ und $g = 2$ bestimmen wir die Potenzen $2^k \in \mathbb{F}_p$ für $k = 0, \ldots, 9$:

k	0	1	2	3	4	5	6	7	8	9
$2^k \in \mathbb{F}_{11}$	1	2	4	8	5	10	9	7	3	6

Man sieht, dass sich jedes Element $\neq 0$ als Potenz von 2 schreiben lässt. Insbesondere ist 2 eine Primitivwurzel modulo p.

Beispiele: Für die Primzahlen $p \leq 71$ sind hier jeweils die kleinsten Primitivwurzeln modulo p angegeben:

p	2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97
g_p	1	2	2	3	2	2	3	2	5	2	3	2	6	3	5	2	2	2	2	7	5	3	2	3	5

Beispiele: Sage berechnet die kleinste Primitivwurzel modulo p mit dem Befehl `primitive_root(p)`. Wir haben die Primzahlen $p \leq 10^8$ betrachtet. In der folgenden Tabelle bezeichnet g, p, dass p die kleinste Primzahl ist, für die g die kleinste Primitivwurzel modulo p ist.

g	p	g	p	g	p	g	p
1	2	30	124153	59	712321	88	
2	3	31	5881	60	697591	89	6366361
3	7	32		61	1171921	90	
4		33	268969	62	658681	91	70716649
5	23	34	48889	63		92	
6	41	35	64609	64		93	65150401
7	71	36		65	11089681	94	5109721
8		37	36721	66	27955201	95	29128969
9		38	55441	67	3384481	96	
10	313	39	166031	68	3733801	97	17551561
11	643	40	1373989	69	110881	98	
12	4111	41	156601	70	5620201	99	
13	457	42	2494381	71	3659401	100	
14	1031	43	95471	72		101	29418841
15	439	44	71761	73	760321	102	
16		45	95525767	74	8954401	103	49443241
17	311	46	273001	75		104	
18	53173	47	275641	76	25291561	105	
19	191	48		77	8359009	106	
20	107227	49		78		107	33358081
21	409	50	23126821	79	7510801	108	
22	3361	51	322999	80		109	67992961
23	2161	52	129361	81		110	
24	533821	53	161831	82	24818641	111	45024841
25		54		83	16889161	112	
26	12391	55	459841	84		113	90441961
27		56		85	23821561	114	$*$
28	133321	57	471769	86	7415641	115	$*$
29	15791	58	336361	87	41299801	116	$*$

Beispiel: Wir haben gezählt, für wieviele Primzahlen $2 \leq p \leq 10^8$ die Zahlen $g = 2, 3, 4, 5, 6, 7, 8, 9$ Primitivwurzeln sind.

g	Anzahl der Primzahlen mit Primitivwurzel g	in %
2	2154733	37.3991
3	2154034	37.3870
4	0	0.0000
5	2268691	39.3770
6	2154463	37.3944
7	2156000	37.4211
8	1292865	22.4399
9	1	0.0000

(Die 9 ist eine Primitivwurzel modulo 2.)

SATZ. *Sei p eine Primzahl und g eine Primitivwurzel modulo p.*

(1) *Ist $a \in \mathbb{F}_p^*$, so gilt* $\operatorname{ord}_p(a) \mid p-1$.
(2) *Ist $d \in \mathbb{N}$ ein Teiler von $p-1$, d.h. gilt $d \mid p-1$, so sind die Elemente der Ordnung d in $\mathbb{F}_p^*$ genau die Elemente*

$$g^{\frac{p-1}{d}\cdot \ell} \text{ mit } 0 \le \ell \le d-1 \text{ und } \operatorname{ggT}(d,\ell) = 1.$$

Insbesondere gibt es genau $\varphi(d)$ Elemente der Ordnung d in $\mathbb{F}_p^*$.
(3) *Die Primitivwurzeln in $\mathbb{F}_p$ sind genau die Elemente*

$$g^{\ell} \text{ mit } 1 \le \ell \le p-1 \text{ und } \operatorname{ggT}(p-1,\ell) = 1.$$

Insbesondere gibt es genau $\varphi(p-1)$ Primitivwurzeln in $\mathbb{F}_p$.

Bemerkungen:

(1) Können wir für eine Primzahl p die Zahl $p-1$ faktorisieren, so können wir mit unserem Ordnungsbestimmungsalgorithmus Ordnungen bestimmen und damit auch Primitivwurzeln.
(2) In Anwendungen wählt man oft (wahrscheinliche) Primzahlen p, sodass $p-1 = 2q$ mit einer (wahrscheinlichen) Primzahl q ist. Für $a \in \mathbb{Z}$ mit $\operatorname{ggT}(p,a) = 1$ gilt dann

$$\operatorname{ord}_p(a) \in \{1, 2, q, 2q\},$$

sodass Ordnungen und Primitivwurzeln leicht bestimmt werden können. $\mathbb{F}_p$ besitzt dann

$$\varphi(p-1) = \varphi(2q) = q-1 = \frac{p-1}{2} - 1 = \frac{p-3}{2}$$

Primitivwurzeln (modulo p).
(3) Will man für die Praxis ein großes p und eine Primitivwurzel g modulo p konstruieren, sucht man zunächst nach wahrscheinlichen Primzahlen, für die sich $p-1$ schnell faktorisieren lässt, also von der Form

$$p-1 = q_1^{e_1} \dots q_{r-1}^{e_{r-1}} \cdot q_r$$

mit kleinen Primzahlen $q_1, \dots, q_{r-1}$ und einer wahrscheinlichen Primzahl q_r ist. Dann probiert man für $g = 2, 3, \dots$, bis man ein g mit $g^{(p-1)/q_i} \neq 1$ für alle i gefunden hat. (Dies funktioniert praktisch schnell.)

Beispiel: $p = 10^{1000} + 10401$ ist (wahrscheinlich) prim und

$$p-1 = 2^5 \cdot 3 \cdot 5^2 \cdot 13313 \cdot q$$

mit einer (wahrscheinlichen) Primzahl q. Mit dem Lemma findet man schnell, dass $g = 7$ die kleinste Primitivwurzel modulo p ist.

4. Diskrete Logarithmen

Gilt für $g, a \in \mathbb{F}_p^*$ und $x \in \mathbb{N}_0$ die Gleichung

$$g^x = a \text{ in } \mathbb{F}_p,$$

so heißt x **diskreter Logarithmus von a zur Basis g (in $\mathbb{F}_p$ oder modulo p)**. x ist natürlich nur modulo $\operatorname{ord}_p(g)$ bestimmt. Wollen wir eine Normierung einführen, so können wir $0 \le x \le \operatorname{ord}_p(g) - 1$ wählen.)

Beispiel: Wir betrachten $p = 10^6 + 3$. Durch Probieren findet man:

- $2^x \equiv 3 \bmod p$ wird gelöst von $x = 254277$,
- $3^x \equiv 2 \bmod p$ hat keine Lösung.

Die Lösbarkeit der Gleichung $g^x = a$ wird in folgendem Lemma charakterisiert:

LEMMA. *Sei p eine Primzahl und $g, a \in \mathbb{F}_p^*$. Die Gleichung $g^x = a$ ist genau dann lösbar, wenn gilt* $\operatorname{ord}_p(a) \mid \operatorname{ord}_p(g)$.

Beweis:

- Es gelte $g^x = a$ für ein $x \in \mathbb{N}_0$. Es folgt
$$1 = (g^{\operatorname{ord}_p(g)})^x = (g^x)^{\operatorname{ord}_p(g)} = a^{\operatorname{ord}_p(g)},$$
woraus sofort
$$\operatorname{ord}_p(a) \mid \operatorname{ord}_p(g)$$
folgt.
- Es gelte umgekehrt $\operatorname{ord}_p(a) \mid \operatorname{ord}_p(g)$. Sei ζ eine Primitivwurzel in $\mathbb{F}_p$. Nach dem vorangegangenen Satz können wir schreiben
$$g = \zeta^{\frac{p-1}{\operatorname{ord}_p(g)} \cdot k} \text{ mit } \operatorname{ggT}(\operatorname{ord}_p(g), k) = 1 \quad \text{und} \quad a = \zeta^{\frac{p-1}{\operatorname{ord}_p(a)} \cdot \ell} \text{ mit } \operatorname{ggT}(\operatorname{ord}_p(a), \ell) = 1.$$
Für $x \in \mathbb{N}_0$ haben wir die Äquivalenzen:
$$\begin{aligned}
g^x = a \quad &\iff \quad \zeta^{\frac{p-1}{\operatorname{ord}_p(g)} \cdot k \cdot x} = \zeta^{\frac{p-1}{\operatorname{ord}_p(a)} \cdot \ell} \quad \iff \\
&\iff \quad \frac{p-1}{\operatorname{ord}_p(g)} \cdot kx \equiv \frac{p-1}{\operatorname{ord}_p(a)} \cdot \ell \bmod p-1 \quad \iff \\
&\iff \quad \frac{p-1}{\operatorname{ord}_p(g)} \cdot kx \equiv \frac{p-1}{\operatorname{ord}_p(g)} \cdot \frac{\operatorname{ord}_p(g)}{\operatorname{ord}_p(a)} \cdot \ell \bmod \frac{p-1}{\operatorname{ord}_p(g)} \cdot \operatorname{ord}_p(g) \quad \iff \\
&\iff \quad kx \equiv \frac{\operatorname{ord}_p(g)}{\operatorname{ord}_p(a)} \cdot \ell \bmod \operatorname{ord}_p(g).
\end{aligned}$$
Wegen $\operatorname{ggT}(\operatorname{ord}_p(g), k) = 1$ ist die Gleichung lösbar und die Behauptung folgt. ■

Wir wissen, dass man g^x mit der square-and-multiply-Methode schnell berechnen kann. Wie aber kann man diskrete Logarithmen berechnen? Zunächst gibt es die

Naive Methode zur Berechnung diskreter Logarithmen: Ist p eine Primzahl und sind g und a gegeben mit $2 \leq g, a \leq p-1$, so probiert man für $x = 0, 1, 2, \ldots, p-1$, ob $g^x \equiv a \bmod p$ gilt. Genauer:

- Man setzt $g_0 = 1$ und berechnet rekursiv $g_n = g_{n-1} \cdot g \bmod p$ für $n = 1, 2, \ldots$. (Dann ist $g^n \equiv g_n \bmod p$.) Für jedes n testet man:
- Gilt $g_n = a$, so ist n der gesuchte diskrete Logarithmus $\log_g a$.
- Gilt $g_n = 1$, so gibt es keine Lösung und man hört auf.

Algorithmus 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 ← 1                                          ▷ h ist g^n
5: for n = 1, ..., p − 2 do
6:     h ← 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, ord_p(g) = n
12:    end if
13: end for
```

Die folgenden Beispiele wurden mit dem naiven Logarithmenberechnungsverfahren gerechnet:

Beispiel: $p = 10^7 + 19$:

$$\begin{aligned} 2^x &\equiv 3 \bmod p \text{ hat keine Lösung (1 sec)}, \\ 6^x &\equiv 2 \bmod p \text{ für } x = 3619301 \text{ (3 sec)}, \\ 6^x &\equiv 3 \bmod p \text{ für } x = 6380718 \text{ (5 sec)}, \\ 6^x &\equiv 5 \bmod p \text{ für } x = 6033274 \text{ (6 sec)}, \\ 6^x &\equiv 7 \bmod p \text{ für } x = 2226069 \text{ (2 sec)}. \end{aligned}$$

Wir haben $p - 1 = 2 \cdot 7^2 \cdot 67 \cdot 1523$ und finden damit

$$\operatorname{ord}(2) = \frac{p-1}{7}, \quad \operatorname{ord}(3) = \frac{p-1}{2}, \quad \operatorname{ord}(6) = p - 1.$$

6 ist also Primitivwurzel modulo p.

Beispiel: Für $p = 10^8 + 7$ ist $g = 5$ eine Primitivwurzel.

$$\begin{aligned} 2^x &\equiv 3 \bmod p \text{ für } x = 14580236 \text{ (12 sec)}, \\ 2^x &\equiv 5 \bmod p \text{ hat keine Lösung (42 sec)}, \\ 5^x &\equiv 2 \bmod p \text{ für } x = 41802970 \text{ (36 sec)}, \\ 5^x &\equiv 3 \bmod p \text{ für } x = 31531094 \text{ (27 sec)}, \\ 5^x &\equiv 7 \bmod p \text{ für } x = 88423753 \text{ (72 sec)}. \end{aligned}$$

Beispiel: Für $p = 10^8 + 7$ und $g = 5$ (eine Primitivwurzel) haben wir für alle a mit $10000 \leq a \leq 10099$ naiv diskrete Logarithmen berechnet. Als Durchschnittswert für die Logarithmen ergab sich 48953788.88, was in der Größenordnung von $\frac{p}{2}$ ist, wie zu erwarten war. Die durchschnittliche Rechenzeit betrug 42.02 Sekunden. Das lässt sich nun leicht extrapolieren:

p	erwartetet durchschnittliche Rechenzeit
$\approx 10^8$	> 0.5 min
$\approx 10^9$	> 5 min
$\approx 10^{10}$	> 50 min
$\approx 10^{11}$	> 8 Stunden

Bemerkungen:

(1) Die obigen Beispiele deuten schon etwas an, dass die Berechnung diskreter Logarithmen praktisch nicht schnell geht. Zwar werden wir später effizientere Algorithmen kennenlernen, dennoch bleibt die praktische Logarithmenberechnung ein schwieriges Problem. Im Englischen nennt man dies: **discrete logarithm problem**.

(2) Um einen Eindruck von der Schwierigkeit des Berechnens von Logarithmen mit aktuellen Methoden zu geben, zitieren wir zwei Mitteilungen aus der Zahlentheorieliste (http://listserv.nodak.edu/archives/nmbrthry.html):
   (a) Am 26. Mai 1998 teilten A. Joux und R. Lercier mit, dass sie einen neuen Rekord für das allgemeine diskrete Logarithmusproblem aufgestellt haben: Für die 90-stellige Primzahl $p = \lfloor 10^{89}\pi \rfloor + 156137$ und $a = \lfloor 10^{89}e \rfloor$ haben sie die diskreten Logarithmen $\log_2 a$, $\log_2(a+1)$, $\log_2(a+2)$, $\log_2(a+3)$ und $\log_2(a+4)$ berechnet.
   (b) Am 30. Juni 1998 teilte D. J. Bernstein mit, dass er demjenigen \$100 zahlt, der als erster eine 500-stellige Primzahl p und $x, y \in \mathbb{N}$ angibt mit $4^x \equiv 9 \bmod p$ und $4^y \equiv 25 \bmod p$.

(3) Das RSA-Kryptoverfahren beruht darauf, dass Primfaktorzerlegung großer Zahlen schwierig ist. Wir werden im weiteren einige (von vielen) Kryptoverfahren vorstellen, die darauf beruhen, dass die Berechnung diskreter Logarithmen schwierig ist.

Ein aktuelles Rekordbeispiel: Am 2. Dezember 2019 wurde von Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomeé und Paul Zimmermann folgender Rekord bekanntgegeben[1]: ...

Ein aktuelles Rekordbeispiel: Am 16. Juni 2016 wurde von Thorsten Kleinjung, Claus Diem, Arjen K. Lenstra, Christine Priplata und Colin Stahlke folgender Rekord bekannt gegeben[2]: Ausgangspunkt war die Primzahl

$$p = \lfloor 2^{766} \cdot \pi \rfloor + 62762.$$

Sie wurde so gewählt, dass $p - 1 = 2q$ mit einer Primzahl q gilt. (Dadurch kann man leicht Ordnungen modulo p bestimmen.) Man stellt fest, dass $g = 11$ die kleinste Primitivwurzel modulo p ist. Dann wurde

$$t = \lfloor 2^{766} \cdot e \rfloor$$

gewählt. Mit Zahlkörpersiebmethoden wurde dann ein diskreter Logarithmus von t zur Basis g modulo p berechnet, also ℓ mit $g^\ell = t$ in $\mathbb{F}_p$ (bzw. $g^\ell \equiv t \bmod p$). Hier sind die Zahlenwerte:

$$\begin{aligned}
p &= 12193448583342869326963419091957961095266573861542513280292736561757668709803065\\
&\quad 05584577389125860826715201547225794072935883258868036433287217994721542199148182\\
&\quad 84150580043314841086968359065934684765951910839383741456789273057916231 9\\
g &= 11\\
t &= 10550454296637299607534962144346753266614553839354518512268868743137690639829257\\
&\quad 32368663611279512238037079039709219817105351667827641603142528649181727795342195\\
&\quad 36684033922108056005893359024146695390020146823557623819331953250209 2663\\
\ell &= 32592361791827056223861598597862370912834133883372105854395081352176815629509163\\
&\quad 83480306379202371756381173524422992340416587484710799119774978643019959726382667\\
&\quad 81162575370644813703762423329783129621567127479417280687495231463348812
\end{aligned}$$

5. Schlüsselaustausch nach Diffie-Hellmann

Wollen Leute mit einem klassischen Kryptosystem verschlüsselte Nachrichten austauschen, müssen sie sich vorher auf einen gemeinsamen Schlüssel einigen. Natürlich kann man dazu z.B. RSA verwenden. Es gibt aber noch andere Möglichkeiten. Diffie und Hellman haben 1976 das folgende Verfahren vorgeschlagen, es war das erste Public-Key-Verfahren:

Schlüsselaustausch nach Diffie-Hellman. Wir nehmen an, A und B wollen sich auf einen gemeinsamen Schlüssel k_{AB} einigen, den wir uns als große Zahl vorstellen können.

- A und B einigen sich öffentlich auf eine Primzahl p und auf eine Zahl g mit $2 \le g \le p-2$. (Es sollte praktisch unmöglich sein, diskrete Logarithmen zur Basis g modulo p zu berechnen.)
- A wählt sich eine Zahl e_A mit $0 \le e_A \le p-2$ und berechnet $f_A = g^{e_A} \bmod p$. A gibt f_A als seinen öffentlichen Diffie-Hellman-Schlüssel bekannt.
- B wählt sich eine Zahl e_B mit $0 \le e_B \le p-2$ und berechnet $f_B = g^{e_B} \bmod p$. B gibt f_B als seinen öffentlichen Diffie-Hellman-Schlüssel bekannt.
- Der gemeinsame Schlüssel k_{AB} von A und B ist

  $$k_{AB} = g^{e_A e_B} \bmod p,$$

  den sich A als

  $$k_{AB} = f_B^{e_A} \bmod p$$

  und B als

  $$k_{AB} = f_A^{e_B} \bmod p$$

  berechnen kann.

Ein Außenstehender C kennt die Zahlen p, g, $f_A = g^{e_A} \bmod p$ und $f_B = g^{e_B} \bmod p$. Kann C daraus den gemeinsamen Schlüssel $k_{AB} = g^{e_A e_B} \bmod p$ berechnen?

[1] 795-bit factoring and discrete logarithms

[2] Discrete Logarithms in GF(p) - 768 bits

(1) Könnte C einen diskreten Logarithmus ℓ von $f_A = g^{e_A} \bmod p$ zur Basis g modulo p bestimmen, d.h. $g^\ell \equiv f_A \bmod p$, so hätte er wegen
$$k_{AB} \equiv f_A^{e_B} \equiv g^{\ell e_B} \equiv f_B^\ell \bmod p$$
sofort den gemeinsamen Schlüssel k_{AB}. (Es ist $e_A \equiv \ell \bmod \operatorname{ord}_p(g)$.) Diese Logarithmenberechnung sollte aber praktisch sehr schwer bzw. unmöglich sein.
(2) Es ist nicht bekannt, ob man aus der Kenntnis der drei Zahlen
$$g, \quad g^{e_A} \bmod p, \quad g^{e_B} \bmod p$$
(und p) den Wert $g^{e_A e_B} \bmod p$ ohne den Umweg über die Logarithmenberechnung finden kann. Dies nennt man das Diffie-Hellman-Problem.

Beispiel: Man einigt sich auf $p = 10^6 + 3$ und $g = 2$. A's öffentlicher Schlüssel ist $f_A = 491373$, B's öffentlicher Schlüssel ist $f_B = 911253$. Was ist der vereinbarte Schlüssel k_{AB}? (Lösung: $e_A = 38628$, $e_B = 729944$, $k_{AB} = 609491$.)

6. Die ElGamal-Verschlüsselung

Überlegungen: Wir nehmen an, A hat einen öffentlichen Diffie-Hellman-Schlüssel (p, g, f_A) (mit $f_A = g^{e_A} \bmod p$).

(1) B will eine stromchiffrierte Nachricht an A schicken. B kann seine Nachricht (nach einem vereinbarten Schema) in eine Zahlenfolge a_i mit $0 \le a_i \le p-1$ umwandeln. Nun braucht B eine Schlüsselfolge k_i mit $0 \le k_i \le p-1$, die er und A kennt. Dann berechnet B die Folge c_i mit $c_i = a_i + k_i \bmod p$ und schickt sie an A. Daraus erhält A sofort a_i wegen $a_i = c_i - k_i \bmod p$.
(2) Wie kann B eine Schlüsselfolge k_i erzeugen, die nur ihm und A bekannt ist? Das geht mit einem Diffie-Hellman-Schlüsselaustausch: B wählt für jedes i eine (zufällige) Zahl z_i mit $0 \le z_i \le p-2$ und berechnet $b_i = g^{z_i} \bmod p$. Wählen wir jetzt
$$k_i = g^{z_i e_A} \bmod p,$$
so ist
$$k_i = b_i^{e_A} \bmod p \quad \text{und} \quad k_i = f_A^{z_i} \bmod p.$$
Schickt also B seinen öffentlichen Schlüssel b_i auch an A, so können nach obigen Gleichungen sowohl A als auch B die Zahl k_i berechnen.
(3) Wir fassen zusammen: B berechnet $b_i = g^{z_i} \bmod p$, $k_i = f_A^{z_i} \bmod p$ und $c_i = a_i + k_i \bmod p = a_i + f_A^{z_i} \bmod p$ und schickt (b_i, c_i) an A. A berechnet damit $k_i = b_i^{e_A} \bmod p$ und $a_i = c_i - k_i \bmod p$.
(4) Dies ist im Wesentlichen die ElGamal-Verschlüsselung, nur dass man statt $c_i = a_i + k_i \bmod p$ dabei $c_i = a_i k_i \bmod p$ bildet.

ElGamal-Verschlüsselung:

(1) **Schlüsselerzeugung:**
   (a) Es wird eine Primzahl p und eine Zahl g mit $2 \le g \le p-2$ zugrundegelegt. (Allgemeine) Logarithmen zur Basis g modulo p sollten sich praktisch nicht berechnen lassen.
   (b) Jeder Teilnehmer A wählt sich eine (zufällige) Zahl e_A mit $0 \le e_A \le p_A - 2$ als geheimen Schlüssel, berechnet $f_A = g^{e_A} \bmod p$ und gibt (p, g, f_A) als seinen öffentlichen ElGamal-Schlüssel bekannt.
(2) **Verschlüsselung:** Wie kann ein Teilnehmer B eine Nachricht T verschlüsselt an A schicken?
   (a) Man muss sich darüber verständigt haben, wie man bei gegebener Primzahl p einen Text in eine Zahlenfolge a_i mit $0 \le a_i \le p-1$ umwandelt und umgekehrt.
   (b) B besorgt sich den öffentlichen Schlüssel (p, g, f_A) von A.
   (c) B wandelt die Nachricht T in eine Folge von Zahlen a_i mit $0 \le a_i \le p-1$ um nach dem vereinbarten Schema.
   (d) Zu jeder Zahl a_i wählt sich B eine zufällige Zahl z_i mit $0 \le z_i \le p-2$.
   (e) B berechnet nun
   $$b_i = g^{z_i} \bmod p \quad \text{und} \quad c_i = a_i \cdot f_A^{z_i} \bmod p.$$
   (f) Der Chiffretext besteht aus der Folge (b_i, c_i) und wird an A geschickt.

(3) **Entschlüsselung:**

(a) A erhält die Folge (b_i, c_i) und berechnet mit Hilfe seines privaten Schlüssels e_A die Ausgangszahl a_i mit

$$a_i = b_i^{p-1-e_A} c_i \bmod p$$

oder alternativ

$$a_i = c_i / b_i^{e_A} \bmod p.$$

(b) Begründung für die Richtigkeit der Entschlüsselung:

$$\begin{aligned} b_i^{p-1-e_A} c_i &\equiv g^{z_i(p-1-e_A)} \cdot a_i (g^{e_A})^{z_i} \equiv a_i \cdot g^{z_i(p-1-e_A)+e_A z_i} \equiv \\ &\equiv a_i \cdot g^{z_i(p-1)} \equiv a_i \bmod p, \end{aligned}$$

weil nach dem kleinen Satz von Fermat $g^{p-1} \equiv 1 \bmod p$ gilt. Die zweite Formel ergibt sich direkt mit $b_i^{e_A} \equiv f_A^{z_i} \bmod p$.

Beispiel: Wir wollen den Text *HEUTE IST DIENSTAG* mit dem ElGamal-Schlüssel

$$(p, g, f) = (10000000019, 2, 2750778126)$$

ElGamal-verschlüsseln. Ein aus Großbuchstaben und Leerzeichen bestehender Text wird zunächst in Blöcke der Länge 5 eingeteilt, dann wird jedes A durch 01, jedes B durch 02, ..., jedes Z durch 26 und jedes Leerzeichen durch 00 ersetzt, woraus schließlich eine Zahlenfolge a_i mit $0 \le a_i \le 10^{10}$ entsteht.

H	E	U	T	E		I	S	T		D	I	E	N	S	T	A	G		
08	05	21	20	05	00	09	19	20	00	04	09	05	14	19	20	01	07	00	00

Nach Wahl von Zufallszahlen $z_1, \ldots, z_4$ erhalten wir dann aus $b_i = g^{z_i} \bmod p$ und $c_i = a_i f^{z_i} \bmod p$ folgende Werte:

i	a_i	z_i	b_i	c_i
1	805212005	6068388851	6608117635	4717065131
2	9192000	407006449	2247779652	4191137189
3	409051419	8187499971	6611839510	7164727505
4	2001070000	6210377615	8909106189	4528216432

Die Zahlenfolge

$$805212005, \quad 9192000, \quad 409051419, \quad 2001070000$$

wird also zu

$$(6608117635, 4717065131), (2247779652, 4191137189), (6611839510, 7164727505), (8909106189, 4528216432)$$

verschlüsselt. (Der private Schlüssel ist $e = 7419668241$.)

Zur Sicherheit der ElGamal-Verschlüsselung: Sei jetzt C ein Außenstehender, der (p, g, f_A) und (b_i, c_i) in Erfahrung gebracht hat. Wie kann C damit auf a_i schließen?

(1) Kann C diskrete Logarithmen zur Basis g modulo p berechnen, so kann er sich aus $f_A = g^{e_A} \bmod p$ den/einen privaten Schlüssel e_A berechnen, also die Rolle von A einnehmen. Daher müssen p und g so gewählt sein, dass sich im Allgemeinen diskrete Logarithmen zur Basis g modulo p praktisch nicht berechnen lassen.

(2) Wegen $c_i \equiv a_i f_A^{z_i} \bmod p$ ist die Kenntnis von a_i äquivalent zur Kenntnis von $f_A^{z_i}$. C kennt die Zahlen

$$g, \quad f_A \equiv g^{e_A} \bmod p, \quad b_i \equiv g^{z_i} \bmod p$$

und würde damit gerne

$$f_A^{z_i} \equiv g^{e_A z_i} \bmod p$$

berechnen. Dies ist wieder das Diffie-Hellman-Problem. Dies darf C nicht lösen können. Heutzutage muss man zur Lösung dieser Aufgabe diskrete Logarithmen berechnen, was wir schon erwähnt haben.

(3) Vermutet C, dass für die Zufallszahl z_i eine Zahl $\widetilde{z}_i$ gewählt wurde, so kann er dies testen, indem er prüft, ob

$$b_i = g^{\widetilde{z}_i} \bmod p$$

gilt. Ist dies der Fall, kann C sofort a_i aus

$$a_i = c_i f_A^{-\widetilde{z}_i} \bmod p$$

berechnen. Für die Sicherheit der ElGamal-Verschlüsselung ist also die Wahl der Zufallszahlen entscheidend. Kann man Zufallszahlen erraten, kann man auch entschlüsseln.

Heutzutage scheint das ElGamal-Verfahren sicher, solange man diskrete Logarithmen modulo p zur Basis g nicht berechnen kann und solange der verwendete Zufallszahlengenerator gut arbeitet.

Bemerkungen:

(1) Bei ElGamal-Verschlüsselung ist der Chiffretext ungefähr doppelt so lang wie der Ausgangstext, da aus einer Zahl a_i beim Verschlüsseln ein Paar (b_i, c_i) wird. (Nachrichtenexpansion)
(2) Für das ElGamal-Kryptosystem braucht man einen (guten) Zufallszahlengenerator. Je nach Wirken des Zufallszahlengenerators erhält man dann bei wiederholtem Verschlüsseln der gleichen Nachricht mit dem gleichen öffentlichen Schlüssel verschiedene verschlüsselte Nachrichten. (Die ElGamal-Verschlüsselung ist randomisiert.)
(3) Vermutet man den i-ten Block des Ausgangstexts zu kennen, also eine Zahl $\widetilde{a}_i$, so müsste man testen, ob $\widetilde{a}_i \equiv c_i / b_i^{e_A} \bmod p$ gilt, was man natürlich nicht kann. (Dies ist bei RSA ganz anders.)
(4) Gilt $b_i = b_j$ für Indizes $i \neq j$, so folgt $b_i^{e_A} \equiv b_j^{e_A} \bmod p$, also

$$\frac{c_i}{a_i} \equiv b_i^{e_A} \equiv b_j^{e_A} \equiv \frac{c_j}{a_j} \bmod p, \quad \text{also} \quad a_j = a_i \cdot \frac{c_j}{c_i} \bmod p.$$

Kennt oder vermutet man a_i, so kann man damit direkt auf a_j schließen. Der Fall $b_i = b_j$ tritt ein, wenn $z_i = z_j$ (oder $z_i \equiv z_j \bmod \operatorname{ord}_p(g)$) gilt. Man sollte also eine Zufallszahl nicht mehrfach verwenden.
(5) Man sollte überlegen, welche Größe man für p wählen muss um gleiche Sicherheit wie RSA bei Vorgabe von $N = \widetilde{p}\widetilde{q}$ zu erhalten. Dies hängt natürlich von den vorhandenen Faktorisierungs- und Logarithmenberechnungsalgorithmen ab. (Aktuelle Vorstellung: $p \approx N$.)

7. Massey-Omura-Verschlüsselung

Das folgende Kryptosystem kommt ohne Schlüsselaustausch und ohne öffentliche Schlüssel aus.

Das Massey-Omura-Kryptosystem zur Nachrichtenübertragung.

(1) Man einigt sich auf eine Primzahl p und darauf, wie man Nachrichten in Zahlenfolgen a_i mit $0 \leq a_i \leq p-1$ übersetzt.
(2) Jeder Teilnehmer wählt sich eine (zufällige) Zahl e_A mit $\operatorname{ggT}(e_A, p-1) = 1$ und berechnet sich d_A mit $d_A e_A \equiv 1 \bmod p-1$. (Dann ist $x \mapsto x^{d_A} \bmod p$ invers zu $x \mapsto x^{e_A} \bmod p$ in $\{0, 1, \ldots, p-1\}$.) A hält die Zahlen e_A und d_A geheim. (A hat also keinen öffentlichen Schlüssel.)
(3) Wie kann A eine Nachricht T, die jetzt durch eine Zahlenfolge a_i gegeben wird, verschlüsselt an B senden?
   (a) A berechnet $b_i = a^{e_A} \bmod p$ und sendet b_i and B, womit B natürlich nichts anfangen kann.
   (b) B berechnet nun $c_i = b_i^{e_B} \bmod p$ und schickt c_i zurück an A.
   (c) Nun berechnet A die Zahl $d_i = c_i^{d_A} \bmod p$ und schickt d_i zurück an B.
   (d) B berechnet schließlich $e_i = d_i^{d_B} \bmod p$, was wegen

$$e_i \equiv d_i^{d_B} \equiv c_i^{d_A d_B} \equiv b_i^{e_B d_A d_B} \equiv a_i^{e_A e_B d_A d_B} \equiv (a_i^{e_A d_A})^{e_B d_B} \equiv a_i \bmod p$$

genau die gesendete Nachricht a_i liefert.

A will a_i an B senden		
A berechnet $b_i = a_i^{e_A} \bmod p$		
A sendet b_i an B	$\xrightarrow{b_i}$	B erhält b_i
		B berechnet $c_i = b_i^{e_B} \bmod p$
A erhält c_i	$\xleftarrow{c_i}$	B sendet c_i an A
A berechnet $d_i = c_i^{d_A} \bmod p$		
A sendet d_i an B	$\xrightarrow{d_i}$	B erhält d_i
		B berechnet $e_i = d_i^{d_B} \bmod p$
		B hat nun $a_i = e_i$

Bemerkung: Zunächst gilt mit obigen Bezeichungen modulo p

$$b_i \equiv a_i^{e_A}, \quad c_i \equiv a_i^{e_A e_B}, \quad d_i \equiv a_i^{e_A e_B d_A} \equiv a_i^{e_B}.$$

Auf dem Übertragungsweg sind eventuell b_i, c_i und d_i öffentlich zugänglich. Zwischen diesen Größen bestehen folgende Beziehungen:

$$c_i \equiv b_i^{e_B}, \quad d_i \equiv c_i^{d_A}, \quad d_i \equiv b_i^{d_A e_B},$$

außerdem gilt

$$a_i \equiv b_i^{d_A} \equiv c_i^{d_A d_B} \equiv d_i^{d_B} \bmod p.$$

- Kann man den diskreten Logarithmus von c_i zur Basis b_i (modulo p) berechnen, so erhält man e_B, dann $d_B \equiv \frac{1}{e_B} \bmod p-1$ und damit $a_i \equiv b_i^{d_A} \bmod p$.
- Kann man den diskreten Logarithmus von d_i zur Basis c_i berechnen, so erhält man d_A und damit direkt $a_i = b_i^{d_A} \bmod p$.

Wesentlich ist also zunächst, dass ein Außenstehender C diskrete Logarithmen modulo p nicht berechnen kann.

Beispiel: A und B haben die Primzahl $p = 1009$ vereinbart. Nun schickt A an B die Zahl 190, B schickt 188 zurück, A wiederum 165. Welche Zahl teilt A mit dem Massey-Omura-Verfahren B damit mit? Mit unseren früheren Bezeichnungen ist $b_i = 190$, $c_i = 188$, $d_i = 165$. Mit Maple berechnen wir den diskreten Logarithmus von $d_i = 165$ zur Basis $c_i = 188$ modulo $p = 1009$ und erhalten $d_A = 109$. Es folgt

$$a_i \equiv b_i^{d_A} \equiv 190^{109} \equiv 843 \bmod p,$$

also $a_i = 843$.

Bemerkungen:

(1) Ein großer Vorteil ist, dass man bei diesem Kryptosystem außer p und dem Verschlüsselungsverfahren nichts vereinbaren muss. Man muss auch keine öffentlichen Schlüssel verwalten.
(2) Ein großer Nachteil ist, dass man 3 Übertragungen braucht, um eine Nachricht von A nach B zu übermitteln.
(3) Stellt A nicht sicher, dass die Nachricht T wirklich an B geht, könnte dies ein Außenstehender C ausnutzen: Er fängt a^{e_A} auf, schickt $a^{e_A e_C}$ zurück, fängt dann wieder $a^{e_C} = a^{e_A e_C d_A}$ auf und berechnet sich daraus $a = a^{e_C d_C}$. Man muss also auf andere Weise sicherstellen, dass die Nachricht den richtigen Empfänger erreicht.
(4) Mir ist nicht bekannt, ob dieses Verfahren irgendwo praktiziert wird.

Bemerkung: Hier ist ein einfaches symmetrisches Kryptosystem: Man wählt sich eine große Primzahl p, einen Exponenten e mit $\mathrm{ggT}(e, p-1) = 1$ und berechnet d mit $ed \equiv 1 \bmod p-1$. Dann ist $x \mapsto x^d \bmod p$ invers zu $x \mapsto x^e \bmod p$. Der Schlüssel ist (p, e), oder einfach nur e, falls p öffentlich bekannt ist. Eine Nachricht stellen wir als eine Folge von Zahlen $a_i \in \{0, 1, \ldots, p-1\}$ dar. Verschlüsselt wird sie zu $b_i = a_i^e \bmod p$. Die Entschlüsselung ergibt sich dann aus $a_i = b_i^d \bmod p$.