

Aufgabe 20

$L^1R^1 = 1111000010101010111100001010101011101111010010100110010101000100$
nach der ersten Runde. Das Ergebnis wurde mittels

`./uebung6.pl 20`

ermittelt, welches die erste Runde implementiert (weiterhin ist DES an sich implementiert, um mit bekannten Klartext/Chiffretext-Paaren die Funktionsfähigkeit der Routine zu überprüfen).

Aufgabe 21

(a)

$$\begin{aligned}
 a &= x^5 - x^4 + x - 1 \\
 b &= x^3 - 3x^2 + 3x - 1 \\
 c &= \frac{1}{4}x - \frac{1}{4} \\
 u &= -\frac{1}{4}x + \frac{3}{8} \\
 v &= \frac{1}{4}x^3 + \frac{1}{8}x^2 - \frac{1}{8} \\
 a \cdot u &= -\frac{1}{4}x^6 + \frac{5}{8}x^5 - \frac{3}{8}x^4 - \frac{1}{4}x^2 + \frac{5}{8}x - \frac{3}{8} \\
 b \cdot v &= \frac{1}{4}x^6 - \frac{5}{8}x^5 + \frac{3}{4}x^4 + \frac{1}{4}x^2 - \frac{3}{8}x + \frac{1}{8} \\
 a \cdot u + b \cdot v &= \frac{1}{4}x - \frac{1}{4}
 \end{aligned}$$

Normiert man dies, so ergibt sich

$$\begin{aligned}
 c &= x - 1 \\
 u &= -x + \frac{3}{2} \\
 v &= x^3 + \frac{1}{2}x^2 - \frac{1}{2} \\
 a \cdot u &= -x^6 + \frac{5}{2}x^5 - \frac{3}{2}x^4 - x^2 + \frac{5}{2}x - \frac{3}{2} \\
 b \cdot v &= x^6 - \frac{5}{2}x^5 + \frac{3}{2}x^4 + x^2 - \frac{3}{2}x + \frac{1}{2} \\
 a \cdot u + b \cdot v &= x - 1
 \end{aligned}$$

(b)

$$\begin{aligned}
a &= x^8 + x^4 + x^3 + x + 1 \\
b &= x^7 + x^5 + x^3 + x \\
c &= 1 \\
u &= x^3 + x + 1 \\
v &= x^4 + x \\
a \cdot u &= x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1 \\
b \cdot v &= x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^2 \\
a \cdot u + b \cdot v &= 1
\end{aligned}$$

Diese Ergebnisse wurde mittels

`./uebung6.pl 21`

ermittelt, welches `poly_extended_euclid` implementiert (mittels eigenem Polynom-Datentyp, dessen `Division/Modulus 1:1` aus der langen Polynomdivision im Skript übernommen ist; ebenso die anderen arithmetischen Operationen); die Koeffizienten werden ebenfalls durch Datentypen dargestellt welche jeweils in F_2 bzw. \mathbb{Q} rechnen.

Aufgabe 22

(a) Dies lässt sich durch Berechnen von $ggT(a, b)$ mit $a = x^2 + x + 1$ und für alle $b \in GF(2^2)$ zeigen; ist dieser immer 1, so ist gezeigt, dass kein Polynom aus $GF(2^2)$ a teilt, und a somit irreduzibel ist. Ein solcher Test wurde implementiert und per

`./uebung6.pl 22`

ausgeführt; Sowohl x als auch $x + 1$ sind hiernach teilerfremd zu a , womit alle möglichen Teilerkandidaten abgedeckt sind.

(b) Würde $x^2 + x$ als Reduktionspolynom benutzt, so ergäbe sich folgende Multiplikationstabelle :

\cdot	0	1	x	$x + 1$
0	0	0	0	0
1	0	1	x	$x + 1$
x	0	x	x	0
$x + 1$	0	$x + 1$	0	$x + 1$

Ein Beispiel, wie diese Tabelle zu Stande kommt :

$$(x + 1) \cdot (x + 1) = x^2 + 1 \pmod{x^2 + x} = x + 1$$

Diese Multiplikationstabelle verletzt Körperaxiome, da nun $(F \setminus \{0\}, \cdot)$ keine abelsche Gruppe mehr ist; Beispielsweise ist das Ergebnis von $(x + 1) \cdot x = 0$, was was nicht wieder in $F \setminus \{0\}$ liegt; wäre $(F \setminus \{0\}, \cdot)$ eine abelsche Gruppe,

so müsste dem aber so sein. Die Körperaxiome setzen voraus, dass $F \setminus \{0\}$ eine abelsche Gruppe ist; somit resultiert die Wahl von $x^2 + x$ als Reduktionspolynom in einer Körperaxiomverletzenden Multiplikationstabelle.

Aufgabe 23

Generatorpolynome in $GF(2^4)$ sind $x, x + 1, x^2, x^2 + 1, x^3 + 1, x^3 + x + 1, x^3 + x^2 + 1$ und $x^3 + x^2 + x$. Das einfachste von diesen ist wohl x , wodurch sich folgende Tabelle ergibt :

b als:	Polynom	4-Bit-Vektor	Hex-Wert	Potenz von a	Log von b
	0	0000	0	—	-
	1	0001	1	a^{15}	15
	x	0010	2	a^1	1
	$x + 1$	0011	3	a^4	4
	x^2	0100	4	a^2	2
	$x^2 + 1$	0101	5	a^8	8
	$x^2 + x$	0110	6	a^5	5
	$x^2 + x + 1$	0111	7	a^{10}	10
	x^3	1000	8	a^3	3
	$x^3 + 1$	1001	9	a^{14}	14
	$x^3 + x$	1010	a	a^9	9
	$x^3 + x + 1$	1011	b	a^7	7
	$x^3 + x^2$	1100	c	a^6	6
	$x^3 + x^2 + 1$	1101	d	a^{13}	13
	$x^3 + x^2 + x$	1110	e	a^{11}	11
	$x^3 + x^2 + x + 1$	1111	f	a^{12}	12

Diese wurde mittels

```
./uebung6.pl 23
```

ermittelt, welches mittels eines $GF(2^4)$ -Datentyps alle 15 Generatorkandidaten ausprobiert; da Zyklizität bzgl. des Generatorpolynoms herrscht (nach Satz 4.2.7 im Skript), ist dies eine recht einfache Schleife.

Aufgabe 24

Wir haben AES implementiert; ein Aufruf von

```
./uebung6.pl 24
```

ergibt folgendes Ergebnis :

$$\begin{array}{l}
\text{PT}_0 = \begin{pmatrix} 32 & 88 & 31 & \text{e0} \\ 43 & 5\text{a} & 31 & 37 \\ \text{f6} & 30 & 98 & 07 \\ \text{a8} & 8\text{d} & \text{a2} & 34 \end{pmatrix} \\
\text{RK}_0 = \begin{pmatrix} 2\text{b} & 28 & \text{ab} & 09 \\ 7\text{e} & \text{ae} & \text{f7} & \text{cf} \\ 15 & \text{d2} & 15 & 4\text{f} \\ 16 & \text{a6} & 88 & 3\text{c} \end{pmatrix} \\
\text{PT}_2 = \begin{pmatrix} \text{a4} & 68 & 6\text{b} & 02 \\ 9\text{c} & 9\text{f} & 5\text{b} & 6\text{a} \\ 7\text{f} & 35 & \text{ea} & 50 \\ \text{f2} & 2\text{b} & 43 & 49 \end{pmatrix} \\
\text{RK}_2 = \begin{pmatrix} \text{f2} & 7\text{a} & 59 & 73 \\ \text{c2} & 96 & 35 & 59 \\ 95 & \text{b9} & 80 & \text{f6} \\ \text{f2} & 43 & 7\text{a} & 7\text{f} \end{pmatrix} \\
\text{SboxState}_2 = \begin{pmatrix} 49 & 45 & 7\text{f} & 77 \\ \text{de} & \text{db} & 39 & 02 \\ \text{d2} & 96 & 87 & 53 \\ 89 & \text{f1} & 1\text{a} & 3\text{b} \end{pmatrix} \\
\text{shiftRows}_2 = \begin{pmatrix} 49 & 45 & 7\text{f} & 77 \\ \text{db} & 39 & 02 & \text{de} \\ 87 & 53 & \text{d2} & 96 \\ 3\text{b} & 89 & \text{f1} & 1\text{a} \end{pmatrix} \\
\text{mixColumns}_2 = \begin{pmatrix} 58 & 1\text{b} & \text{db} & 1\text{b} \\ 4\text{d} & 4\text{b} & \text{e7} & 6\text{b} \\ \text{ca} & 5\text{a} & \text{ca} & \text{b0} \\ \text{f1} & \text{ac} & \text{a8} & \text{e5} \end{pmatrix} \\
\text{PT}_1 = \begin{pmatrix} 19 & \text{a0} & 9\text{a} & \text{e9} \\ 3\text{d} & \text{f4} & \text{c6} & \text{f8} \\ \text{e3} & \text{e2} & 8\text{d} & 48 \\ \text{be} & 2\text{b} & 2\text{a} & 08 \end{pmatrix} \\
\text{RK}_1 = \begin{pmatrix} \text{a0} & 88 & 23 & 2\text{a} \\ \text{fa} & 54 & \text{a3} & 6\text{c} \\ \text{fe} & 2\text{c} & 39 & 76 \\ 17 & \text{b1} & 39 & 05 \end{pmatrix} \\
\text{SboxState}_1 = \begin{pmatrix} \text{d4} & \text{e0} & \text{b8} & 1\text{e} \\ 27 & \text{bf} & \text{b4} & 41 \\ 11 & 98 & 5\text{d} & 52 \\ \text{ae} & \text{f1} & \text{e5} & 30 \end{pmatrix} \\
\text{shiftRows}_1 = \begin{pmatrix} \text{d4} & \text{e0} & \text{b8} & 1\text{e} \\ \text{bf} & \text{b4} & 41 & 27 \\ 5\text{d} & 52 & 11 & 98 \\ 30 & \text{ae} & \text{f1} & \text{e5} \end{pmatrix} \\
\text{mixColumns}_1 = \begin{pmatrix} 04 & \text{e0} & 48 & 28 \\ 66 & \text{cb} & \text{f8} & 06 \\ 81 & 19 & \text{d3} & 26 \\ \text{e5} & 9\text{a} & 7\text{a} & 4\text{c} \end{pmatrix} \\
\text{PT}_3 = \begin{pmatrix} \text{aa} & 61 & 82 & 68 \\ 8\text{f} & \text{dd} & \text{d2} & 32 \\ 5\text{f} & \text{e3} & 4\text{a} & 46 \\ 03 & \text{ef} & \text{d2} & 9\text{a} \end{pmatrix} \\
\text{RK}_3 = \begin{pmatrix} 3\text{d} & 47 & 1\text{e} & 6\text{d} \\ 80 & 16 & 23 & 7\text{a} \\ 47 & \text{fe} & 7\text{e} & 88 \\ 7\text{d} & 3\text{e} & 44 & 3\text{b} \end{pmatrix} \\
\text{SboxState}_3 = \begin{pmatrix} \text{ac} & \text{ef} & 13 & 45 \\ 73 & \text{c1} & \text{b5} & 23 \\ \text{cf} & 11 & \text{d6} & 5\text{a} \\ 7\text{b} & \text{df} & \text{b5} & \text{b8} \end{pmatrix} \\
\text{shiftRows}_3 = \begin{pmatrix} \text{ac} & \text{ef} & 13 & 45 \\ \text{c1} & \text{b5} & 23 & 73 \\ \text{d6} & 5\text{a} & \text{cf} & 11 \\ \text{b8} & 7\text{b} & \text{df} & \text{b5} \end{pmatrix} \\
\text{mixColumns}_3 = \begin{pmatrix} 75 & 20 & 53 & \text{bb} \\ \text{ec} & 0\text{b} & \text{c0} & 25 \\ 09 & 63 & \text{cf} & \text{d0} \\ 93 & 33 & 7\text{c} & \text{dc} \end{pmatrix}
\end{array}$$

$$\begin{array}{rcl}
PT_4 & = & \begin{pmatrix} 48 & 67 & 4d & d6 \\ 6c & 1d & e3 & 5f \\ 4e & 9d & b1 & 58 \\ ee & 0d & 38 & e7 \end{pmatrix} & PT_5 & = & \begin{pmatrix} e0 & c8 & d9 & 85 \\ 92 & 63 & b1 & b8 \\ 7f & 63 & 35 & be \\ e8 & c0 & 50 & 01 \end{pmatrix} \\
RK_4 & = & \begin{pmatrix} ef & a8 & b6 & db \\ 44 & 52 & 71 & 0b \\ a5 & 5b & 25 & ad \\ 41 & 7f & 3b & 00 \end{pmatrix} & RK_5 & = & \begin{pmatrix} d4 & 7c & ca & 11 \\ d1 & 83 & f2 & f9 \\ c6 & 9d & b8 & 15 \\ f8 & 87 & bc & bc \end{pmatrix} \\
SboxState_4 & = & \begin{pmatrix} 52 & 85 & e3 & f6 \\ 50 & a4 & 11 & cf \\ 2f & 5e & c8 & 6a \\ 28 & d7 & 07 & 94 \end{pmatrix} & SboxState_5 & = & \begin{pmatrix} e1 & e8 & 35 & 97 \\ 4f & fb & c8 & 6c \\ d2 & fb & 96 & ae \\ 9b & ba & 53 & 7c \end{pmatrix} \\
shiftRows_4 & = & \begin{pmatrix} 52 & 85 & e3 & f6 \\ a4 & 11 & cf & 50 \\ c8 & 6a & 2f & 5e \\ 94 & 28 & d7 & 07 \end{pmatrix} & shiftRows_5 & = & \begin{pmatrix} e1 & e8 & 35 & 97 \\ fb & c8 & 6c & 4f \\ 96 & ae & d2 & fb \\ 7c & 9b & ba & 53 \end{pmatrix} \\
mixColumns_4 & = & \begin{pmatrix} 0f & 60 & 6f & 5e \\ d6 & 31 & c0 & b3 \\ da & 38 & 10 & 13 \\ a9 & bf & 6b & 01 \end{pmatrix} & mixColumns_5 & = & \begin{pmatrix} 25 & bd & b6 & 4c \\ d1 & 11 & 3a & 4c \\ a9 & d1 & 33 & c0 \\ ad & 68 & 8e & b0 \end{pmatrix} \\
PT_6 & = & \begin{pmatrix} f1 & c1 & 7c & 5d \\ 00 & 92 & c8 & b5 \\ 6f & 4c & 8b & d5 \\ 55 & ef & 32 & 0c \end{pmatrix} & PT_7 & = & \begin{pmatrix} 26 & 3d & e8 & fd \\ 0e & 41 & 64 & d2 \\ 2e & b7 & 72 & 8b \\ 17 & 7d & a9 & 25 \end{pmatrix} \\
RK_6 & = & \begin{pmatrix} 6d & 11 & db & ca \\ 88 & 0b & f9 & 00 \\ a3 & 3e & 86 & 93 \\ 7a & fd & 41 & fd \end{pmatrix} & RK_7 & = & \begin{pmatrix} 4e & 5f & 84 & 4e \\ 54 & 5f & a6 & a6 \\ f7 & c9 & 4f & dc \\ 0e & f3 & b2 & 4f \end{pmatrix} \\
SboxState_6 & = & \begin{pmatrix} a1 & 78 & 10 & 4c \\ 63 & 4f & e8 & d5 \\ a8 & 29 & 3d & 03 \\ fc & df & 23 & fe \end{pmatrix} & SboxState_7 & = & \begin{pmatrix} f7 & 27 & 9b & 54 \\ ab & 83 & 43 & b5 \\ 31 & a9 & 40 & 3d \\ f0 & ff & d3 & 3f \end{pmatrix} \\
shiftRows_6 & = & \begin{pmatrix} a1 & 78 & 10 & 4c \\ 4f & e8 & d5 & 63 \\ 3d & 03 & a8 & 29 \\ fe & fc & df & 23 \end{pmatrix} & shiftRows_7 & = & \begin{pmatrix} f7 & 27 & 9b & 54 \\ 83 & 43 & b5 & ab \\ 40 & 3d & 31 & a9 \\ 3f & f0 & ff & d3 \end{pmatrix} \\
mixColumns_6 & = & \begin{pmatrix} 4b & 2c & 33 & 37 \\ 86 & 4a & 9d & d2 \\ 8d & 89 & f4 & 18 \\ 6d & 80 & e8 & d8 \end{pmatrix} & mixColumns_7 & = & \begin{pmatrix} 14 & 46 & 27 & 34 \\ 15 & 16 & 46 & 2a \\ b5 & 15 & 56 & d8 \\ bf & ec & d7 & 43 \end{pmatrix}
\end{array}$$

$$\begin{array}{l}
PT_8 = \begin{pmatrix} 5a & 19 & a3 & 7a \\ 41 & 49 & e0 & 8c \\ 42 & dc & 19 & 04 \\ b1 & 1f & 65 & 0c \end{pmatrix} \\
RK_8 = \begin{pmatrix} ea & b5 & 31 & 7f \\ d2 & 8d & 2b & 8d \\ 73 & ba & f5 & 29 \\ 21 & d2 & 60 & 2f \end{pmatrix} \\
SboxState_8 = \begin{pmatrix} be & d4 & 0a & da \\ 83 & 3b & e1 & 64 \\ 2c & 86 & d4 & f2 \\ c8 & c0 & 4d & fe \end{pmatrix} \\
shiftRows_8 = \begin{pmatrix} be & d4 & 0a & da \\ 3b & e1 & 64 & 83 \\ d4 & f2 & 2c & 86 \\ fe & c8 & c0 & 4d \end{pmatrix} \\
mixColumns_8 = \begin{pmatrix} 00 & b1 & 54 & fa \\ 51 & c8 & 76 & 1b \\ 2f & 89 & 6d & 99 \\ d1 & ff & cd & ea \end{pmatrix} \\
SboxState_{10} = \begin{pmatrix} e9 & cb & 3d & af \\ 09 & 31 & 32 & 2e \\ 89 & 07 & 7d & 2c \\ 72 & 5f & 94 & b5 \end{pmatrix} \\
shiftRows_{10} = \begin{pmatrix} e9 & cb & 3d & af \\ 31 & 32 & 2e & 09 \\ 7d & 2c & 89 & 07 \\ b5 & 72 & 5f & 94 \end{pmatrix} \\
PT_9 = \begin{pmatrix} ea & 04 & 65 & 85 \\ 83 & 45 & 5d & 96 \\ 5c & 33 & 98 & b0 \\ f0 & 2d & ad & c5 \end{pmatrix} \\
RK_9 = \begin{pmatrix} ac & 19 & 28 & 57 \\ 77 & fa & d1 & 5c \\ 66 & dc & 29 & 00 \\ f3 & 21 & 41 & 6e \end{pmatrix} \\
SboxState_9 = \begin{pmatrix} 87 & f2 & 4d & 97 \\ ec & 6e & 4c & 90 \\ 4a & c3 & 46 & e7 \\ 8c & d8 & 95 & a6 \end{pmatrix} \\
shiftRows_9 = \begin{pmatrix} 87 & f2 & 4d & 97 \\ 6e & 4c & 90 & ec \\ 46 & e7 & 4a & c3 \\ a6 & 8c & d8 & 95 \end{pmatrix} \\
mixColumns_9 = \begin{pmatrix} 47 & 40 & a3 & 4c \\ 37 & d4 & 70 & 9f \\ 94 & e4 & 3a & 42 \\ ed & a5 & a6 & bc \end{pmatrix}
\end{array}$$

PT_i bezeichnet hier den Plaintext der Runde i , RK_i den Rundenschlüssel der Runde i , $SboxState_i$ den Zustand nach Anwendung der i ten S-Box, $shiftRows_i$ den Zustand nach Anwendung von ShiftRows in Runde i , $mixColumns_i$ den Zustand nach der i ten Anwendung von mixColumnsn.

Der Ciphertext ist : **39 25 84 1d 02 dc 09 fb dc 11 85 97 19 6a 0b 32**

Weitere aufrufe mit anderen Testvektoren fuehrten ebenfalls zum Erfolg (insbesondere faellt einem bei der Suche danach auf, dass Google den Key des Uebungzettels schon recht gut kennt :)

Anhang

```
#!/usr/local/bin/perl -w
use strict; $|=1; no strict 'refs';
use Number::Fraction;
```

```

sub prob20 {
  sub sb {
    my %s = ( 1 => [[14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7],
                  [0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8],
                  [4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0],
                  [15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13]],
             2 => [[15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10],
                  [3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5],
                  [0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15],
                  [13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9]],
             3 => [[10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8],
                  [13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1],
                  [13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7],
                  [1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12]],
             4 => [[7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15],
                  [13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9],
                  [10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4],
                  [3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14]],
             5 => [[2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9],
                  [14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6],
                  [4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14],
                  [11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3]],
             6 => [[12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11],
                  [10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8],
                  [9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6],
                  [4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13]],
             7 => [[4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1],
                  [13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6],
                  [1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2],
                  [6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12]],
             8 => [[13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7],
                  [1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2],
                  [7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8],
                  [2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11]]);
    split //, sprintf '%04b', $s{$_[0]}->
      [oct'b'.$_[1].$_[6]]->[oct'b'.join'',splice @_,2,4]
  }

  sub f {
    my @a = @{shift()}; my @j = @{shift()}; my $t;
    my @e = qw(32 1 2 3 4 5 4 5 6 7 8 9 8 9 10 11
              12 13 12 13 14 15 16 17 16 17 18 19 20 21 20 21
              22 23 24 25 24 25 26 27 28 29 28 29 30 31 32 1);
    my @p = qw(16 7 20 21 29 12 28 17 1 15 23 26 5 18 31 10
              2 8 24 14 32 27 3 9 19 13 30 6 22 11 4 25);
    $_[$_] = $a[$e[$_]-1] for 0..@e-1;
    $_[$_] ^= $j[$_]+0 for 0..@j-1;
    my @t; my @r; push @r, sb(++$t, @t) while @t = splice @_,0,6;
    $_[$_] = $r[$p[$_]-1] for 0..@p-1;
  }
}

```

```

    splice @_,0,32
}

sub e_des_round1 {
    my @k = @{shift()}; my @r = @{shift()}; my @l = splice @r, 0, 32;
    (@r, map { $_^0+shift @l } f(@r], [@k]))
}

sub e {
    my $r = shift;
    my @k = map { split //, sprintf '%04b', oct'0x'.$_ } split //, shift;

    my @pc1 = qw(57 49 41 33 25 17 9 1 58 50 42 34 26 18
                 10 2 59 51 43 35 27 19 11 3 60 54 44 36
                 63 55 47 39 31 23 15 7 62 54 46 38 30 22
                 14 6 61 53 45 37 29 21 13 5 28 20 12 4);
    my @pc2 = qw(14 17 11 24 1 5 3 28 15 6 21 10 23 19 12 4
                 26 8 16 7 27 20 13 2 41 52 31 37 47 55 30 40
                 51 45 33 48 44 49 39 56 34 53 46 42 50 36 29 32);
    my @ro = qw(1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1);

    $_[$_] = $k[$pc1[$_-1] for 0..@pc1-1;
    for (1..$r) {
        my @l = splice @_,0,28; my @r = @_;
        @_ = splice @l,0,$ro[$_-1]; @l = (@l, @_);
        @_ = splice @r,0,$ro[$_-1]; @_ = (@l, @r, @_)
    }
    $k[$_] = $_[$pc2[$_-1] for 0..@pc2-1;
    splice @k,0,48
}

sub e_des {
    sub permute {
        my @pix = @{shift()}; my @x = @{shift()};
        $_[$_] = $x[$pix[$_-1] for 0..@pix-1; @_
    }

    sub reversepi {
        my @a = splice @_, 0, @_;
        $_[$a[$_-1] = $_+1 for 0..@a-1; @_
    }

    my @ip = qw(58 50 42 34 26 18 10 2 60 52 44 36 28 20 12 4
                62 54 46 38 30 22 14 6 64 56 48 40 32 24 16 8
                57 49 41 33 25 17 9 1 59 51 43 35 27 19 11 3
                61 53 45 37 29 21 13 5 63 55 47 39 31 23 15 7);

    my $k = shift; my @r = permute (\@ip, [map { split //, sprintf '%04b',
        oct'0x'.$_ } split //, shift]);
    for (1..16) {

```

```

    my @l = splice @r, 0, 32;
    @r = (@r, map { $_^0+shift @l } f(@r), [e($_, $k)]));
  }
  @r = permute ([reverseipi (@ip)], [splice (@r,32,32), @r]);
  $_ .= sprintf ('%x',oct'b'.join'',@_) while @_ = splice @r,0,4;
  $_
}

print join '', e_des_round1
  ([split //, '00011011000000101110111111111000111000001110010'],
  [split //, '110011000000000011001100111111111'
  '11110000101010101111000010101010']), "\n";

print e_des ('133457799BBCDF1', '0123456789ABCDEF'), "\n";
}

sub prob21 {
  sub poly_extended_euclid {
    @{$_[1]} or return $_[0], new polynomial (1), new polynomial ();
    my ($d, $x, $y) = poly_extended_euclid ($_[1], $_[0] % $_[1]);
    $d, $y, $x-($_[0]/$_[1])*$y
  }

  for ([new polynomial (map { new Number::Fraction ($_,1) } 1,-1,0,0,1,-1),
    new polynomial (map { new Number::Fraction ($_,1) } 1,-3,3,-1)],
    [new polynomial (map { new f2 ($_) } 1,0,0,0,1,1,0,1,1),
    new polynomial (map { new f2 ($_) } 1,0,1,0,1,0,1,0)]) {
    my ($a, $b) = @{$_};
    my ($c, $u, $v) = poly_extended_euclid ($a, $b);
    print "a = $a\nb = $b\nc = $c\nu = $u\nv = $v\na*u = ", $a*$u,
      "\nb*v = ", $b*$v, "\na*u+b*v = ", $a*$u+$b*$v, "\n\n"
  }
}

sub prob22 {
  sub poly_euclid {
    @{$_[1]} ? poly_euclid ($_[1], $_[0]%$_[1]) : $_[0]
  }

  my $a = new polynomial (map { new f2 ($_) } 1,1,1);
  print poly_euclid ($a, new polynomial (map { new f2 ($_) }
    split //, sprintf "%02b", $_)), "\n" for 2..3;
}

sub prob23 {
  for my $i (1..15) {
    my $c = new gf24 (split //, sprintf "%04b", $i);
    my $d = $c; my %gf; my $e = 1;
    while (! exists $gf{$d}) {
      $gf{$d}{p} = $d;
    }
  }
}

```

```

    $gf{$d}{e} = $e++;
    $d = $d * $c;
}
if (keys %gf == 15) {
    print "Generatorpolynom a=$c : \n ";
    print sprintf('%11s','0')." 0000 0      - -\n ";
    print join "\n ",
        (map { sprintf('%11s',$_).' '.
            sprintf('%04s',join(' ',@{$gf{$_}{p}})).' '.
            sprintf('%1x', oct'b'.join(' ',@{$gf{$_}{p}})).' '.
            sprintf('%6s', "a^\{$gf{$_}{e}\").' '.
            sprintf('%2s', $gf{$_}{e}) }
        sort { oct'b'.join(' ',@{$gf{$a}{p}}) <=>
            oct'b'.join(' ',@{$gf{$b}{p}})}
        keys (%gf)), "\n\n";
}
}
}

use vars qw(@aesSBox $r);

sub prob24 {
    sub matrixprint {
        while (my @t = splice @_,0,4) {
            print ' ', join ' ', (map { sprintf '%02x', $_ } @t), "\n"
        }
    }

    sub performVoodoo {
        @_ = split //, shift;
        push @_, oct'0x'.join'',splice @_,0,2 for 0..@_/2-1; @_
    }

    @aesSBox = performVoodoo join '',
        qw(637c777bf26b6fc53001672bfed7ab76ca82c97dfa5947f0add4a2af9ca472c0
        b7fd9326363ff7cc34a5e5f171d8311504c723c31896059a071280e2eb27b275
        09832c1a1b6e5aa0523bd6b329e32f8453d100ed20fcb15b6acbbe394a4c58cf
        d0efaafb434d338545f9027f503c9fa851a3408f929d38f5bcb6da2110fff3d2
        cd0c13ec5f974417c4a77e3d645d197360814fdc222a908846eeb814de5e0bdb
        e0323a0a4906245cc2d3ac629195e479e7c8376d8dd54ea96c56f4ea657aae08
        ba78252e1ca6b4c6e8dd741f4bbd8b8a703eb5664803f60e613557b986c11d9e
        e1f8981169d98e949b1e87e9ce5528df8ca1890dbfe6426841992d0fb054bb16);

    sub substitute {
        my ($r, $c) = map {hex $_} split //, sprintf '%02x', shift;
        $aesSBox [$r*16+$c]
    }

    sub rotWord {
        push @_, shift; @_
    }
}

```

```

}

sub subWord {
    push @_, substitute(shift @_) for 0..3; @_
}

sub shiftRows {
    for my $i (1..3) {
        my @t = splice @_,4,4;
        @t = rotWord @t for 1..$i;
        push @_, @t
    }
    print "shiftRows$r:\n"; matrixprint @_;
    @_
}

sub xtime {
    my $b = ($_ = shift) > 127 ? 0x1b : 0;
    $_ = ($_ << 1) % 256 ^ $b
}

sub mixColumns {
    my @old = @_;
    for my $i (0..3) {
        my $T = $old[0+$i] ^ $old[4+$i] ^ $old[8+$i] ^ $old[12+$i];
        $_[4*$i+$i] ^= xtime($old[4*$i+$i] ^ $old[(4*$i+4)%16+$i]) ^ $T for 0..3
    }
    print "mixColumns$r:\n"; matrixprint @_;
    @_
}

sub SboxState {
    push @_, subWord splice @_,0,4 for 0..3;
    print "SboxState$r:\n"; matrixprint @_;
    @_
}

sub expandKey {
    my @RCon = performVoodoo join '',
        qw(0100000002000000040000000800000010000000
            200000004000000080000001b00000036000000);

    for my $i (4..43) {
        my @tmp;
        $tmp[4+$_] = $_[$_] for -4..-1;

        unless ($i % 4) {
            @tmp = subWord rotWord @tmp;
            $tmp[$_] ^= $RCon [$i-4+$_] for 0..3;
        }
    }
}

```

```

    push @_, $_[($i-4) * 4 + $_]^$tmp[$_] for 0..3
  }
  @_
}

sub addRoundKey {
  my @rk = @{shift ()}; my @tc = @{shift ()};

  for my $i (0..3) {
    for my $j (0..3) {
      $tc[$i + $j * 4] ^= $rk [$j + $i * 4]
    }
  }
  @tc
}

sub twistAndShout {
  my @newM;
  for my $i (0..3) {
    for my $j (0..3) {
      $newM[$i + $j * 4] = $_[$j + $i * 4]
    }
  }
  @newM
}

sub e_AES {
  my @ek = expandKey performVoodoo shift;
  my @pt = twistAndShout performVoodoo shift;
  my @rk; $r = 0;

  print "PT$r:\n"; matrixprint @pt;
  print "RK$r:\n"; matrixprint twistAndShout @rk = splice @ek,0,16;

  @pt = addRoundKey \@rk, \@pt;
  for $r (1..9) {
    print "PT$r:\n"; matrixprint @pt;
    print "RK$r:\n"; matrixprint twistAndShout @rk = splice @ek,0,16;
    @pt = addRoundKey \@rk, [mixColumns shiftRows SboxState @pt];
    print "\n"
  }
  $r=10;
  my @ct = twistAndShout addRoundKey \@ek, [shiftRows SboxState @pt];
  @ct
}

@_ = e_AES (defined $_[0] ? $_[0] : '2B7E151628AED2A6ABF7158809CF4F3C',
           defined $_[1] ? $_[1] : '3243F6A8885A308D313198A2E0370734');

```

```

    print 'Ciphertext : '; printf "%02x ", $_[$_] for 0..@_-1; print "\n";
}

&{'prob'.$ARGV[0]} (splice @ARGV,1);

package f2;

use overload
    '+' => \&add, '-' => \&add,
    '*' => \&mul, '/' => \&mul,
    '==' => \&equals, '!=' => sub { ! equals (@_) },
    '""' => sub { $_[0]->[0] };

sub new { bless ref $_[1] ? $_[1] : [$_[1]], $_[0] }
sub clone { f2->new ($_[0]->[0]) }
sub add {
    my ($l, $r) = (shift, shift);
    $r = ref($r) ? $r : f2->new($r);
    f2->new (($l->[0] + $r->[0])%2)
}
sub mul {
    my ($l, $r) = (shift, shift);
    $r = ref($r) ? $r : f2->new($r);
    f2->new ($l->[0]*$r->[0])
}
sub equals {
    my ($l, $r) = (shift, shift);
    $r = ref($r) ? $r : f2->new($r);
    $r->[0] == $l->[0]
}
1;

package polynomial;

use overload
    '+' => \&add,
    '-' => \&sub,
    '*' => \&mul,
    '/' => \&div,
    '%' => \&mod,
    '==' => \&equals,
    '""' => \&tostring;

sub new { bless [splice @_,1,@_-1], $_[0] }
sub clone { polynomial->new (@{$_[0]}) }

sub tostring {
    @_ = @{$_[0]};
    my $l = @_-1;
    $_ = '('.join ("+", map { $l--; $_==0 ? () :

```

```

    ($_ == 1 ? '' : $_).'x^'.($1+1) } @_.)''';
s/x\^0/ 1/g; s/(\\(\\))/g; s/x\^1([\^0-9]|$)/x$1/g;
$_
}

sub mul {
my @a = @{$_[0]}; my @b = @{$_[1]}; @_ = ();
for my $i (0..@b-1) {
  for my $j (0..@a-1) {
    $_[$i+$j] = defined $_[$i+$j] ? $_[$i+$j] : 0;
    $_[$i+$j] += $b[$i] * $a[$j];
  }
}
new polynomial (@_)
}

sub subt {
my @a = @{$_[0]}; my @b = @{$_[1]}; @_ = ();
my $i = @a; my $j = @b;
while ($i > 0 or $j > 0) {
  unshift @_, 0;
  $_[0] += $a[$i] if $i-- > 0;
  $_[0] -= $b[$j] if $j-- > 0;
}
new polynomial (@_)
}

sub mod {
my @a = @{$_[0]}; my @b = @{$_[1]};
@a > @a and return \@a;
while (@a >= @b) {
  my @t;
  push @t, $a[0] / $b[0];
  @a-@b > 0 and push @t,0 for 1..@a-@b;
  @t = @{mul ([@t], [@b])};
  @a = @{subt ([@a], [@t])};
  while (@a > 0 and $a[0] == 0) { shift @a }
}
new polynomial (@a)
}

sub div {
my @a = @{$_[0]}; my @b = @{$_[1]};
@a > @a and return \@a;
my @r = ();
@a-@b > 0 and push @r,0 for 1..@a-@b;
while (@a >= @b) {
  $r[@a-@b] = $a[0] / $b[0];
  my @t;
  push @t, $a[0] / $b[0];

```

```

    @a-@b > 0 and push @t,0 for 1..@a-@b;
    @t = @{mul ([@t], [@b])};
    @a = @{subt ([@a],[@t])};
    while (@a > 0 and $a[0] == 0) { shift @a }
  }
  new polynomial (reverse @r)
}

sub equals {
  @{$_[0]} != @{$_[1]} and return 0;
  my $t = 1;
  $_[0]->[$_] != $_[1]->[$_] and $t = 0 for 0..@{$_[0]}-1;
  $t
}

sub add {
  my @a = @{$_[0]}; my @b = @{$_[1]}; @_ = ();
  my $i = @a; my $j = @b;
  while ($i > 0 and $j > 0) {
    unshift @_, 0;
    $_[0] += $a[$i] if $i-- > 0;
    $_[0] += $b[$j] if $j-- > 0;
  }
  new polynomial (@_);
}

1;

package gf24;

use vars ('@ISA');
BEGIN { @ISA = qw(polynomial);
use overload '*' => \&mul;
}

sub new {
  my $class = shift;
  my $self = $class->SUPER::new (map { ref $_ ? $_ : new f2 ($_) } @_);
  bless $self, $class;
  $self
}

sub mul {
  new gf24(@{new polynomial (@{$_[0]}) *
    new polynomial (@{$_[1]}) %
    new polynomial (map { new f2 ($_) } 1,0,0,1,1)});
}

1

```