Zufallsgenerator des `plumChips` in der Version v2.4

Als Zufallsgenerator des `plumChips` wird ein alternierender Stop–and–go–Generator eingesetzt wie er in [Schneier15], S. 383ff, beschrieben wird. Im `plumChip` v2.4 besteht das erste Schieberegister aus 61, das zweite aus 73 und das dritte aus 89 Flipflops.

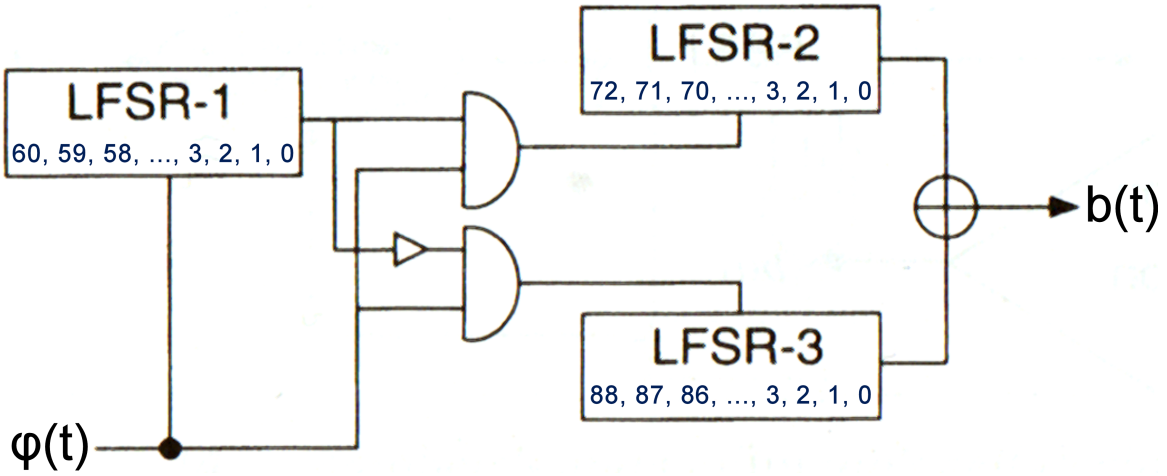

Abb. 1: Der Zufallsgenerator wird aus drei Schieberegistern (LFSR) zusammengesetzt.

Eintragen der Zufallssaat

Die Startbelegung der drei Schieberegister (LFSR) des Zufallsgenerators erfolgt durch eine Zufallssaat bestehend aus **dreißig Bytes**. Die Zufallssaat wird in der Version v2.4 durch den Befehl 'D' an den `plumChip` übermittelt. Man achte darauf, dass keines der drei Schieberegister ausschließlich NULL-Zeichen erhält, die Zufallssaat also geeignet gewählt wird.[1]

Die Zufallssaat wird von hinten nach vorne zuerst ins Schieberegister 1 und dann in die Schieberegister 2 und 3 „eingetragen". Damit ist gemeint, dass die Zustände der Flipflops gemäß der gesetzten und nicht gesetzten Bits in den Zeichen der Saat verändert werden. Enthält die Saat beispielsweise ein Zeichen `X`, dann nimmt man dessen ASCII-Code `0101 1000` ($58 hexadezimal) und setzt die zugeordneten Flipflops entsprechend auf die Zustände '0', '1', '0', '1', '1', '0', '0', '0'.

Wie die Zufallssaat von hinten nach vorne in die drei Schieberegister eingebracht wird, sei mit dem Text `Donaudampfschifffahrtskapitaen` als Zufallssaat erläutert, die also aus genau 30 Zeichen besteht.

Schieberegister 1

Das erste Schieberegister `LFSR-1` besteht aus 61 Flipflops. Die ersten 8 Zeichen der Saat, also `Donaudam`, werden von Flipflop 0 (ganz rechts) bis Flipflop 60 (ganz links) eingetragen. Vom letzten Zeichen `m` (`0110 1101`) der Saat bleiben die vorderen Bits ungenutzt. Damit ergibt sich wie folgt der hexadezimal und binär dargestellte Zustand der Flipflops des `LFSR-1`.

```
(m)aduanoD
(01101) $61 $64 $75 $61 $6E $6F $44

Flipflop
 60                                                                  0

  0 1101 0110 0001 0110 0100 0111 0101 0110 0001 0110 1110 0110 1111 0100 0100
6     D    6    1    6    4    7    5    6    1    6    E    6    F    4    4
```

[1] In der Version v2.4 kommen in den Schieberegistern nur XOR-Gatter zum Einsatz. Wenn sich alle Flipflops eines Schieberegisters im Zustand '0' befinden, verlassen sie daher diesen Zustand nie wieder.

Schieberegister 2

In gleicher Weise dienen die nächsten 10 Zeichen `pfschifffa` der Saat für die 73 Flipflops des zweiten Schieberegisters `LFSR-2`.

```
(a)fffihcsfp
(1) $66 $66 $66 $69 $68 $63 $73 $66 $70

1 0110 0110 0110 0110 0110 0110 0110 1001 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000
    6    6    6    6    6    6    6    9    6    8    6    3    7    3    6    6    7    0
```

Schieberegister 3

Die 89 Flipflops des dritten Schieberegisters `LFSR-3` werden durch die letzten 11 Zeichen `hrtskapitaen` der Saat gesetzt.

```
(n)eatipakstrh
(0) $65 $61 $74 $69 $70 $61 $6B $73 $74 $72 $68

0 0110 0101 0110 0001 0111 0100 0110 1001 0111 0000 0110 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000
    6    5    6    1    7    4    6    9    7    0    6    1    6    B    7    3    7    4    7    2    6    8
```

Kontrolle der Funktion der Schieberegister

Mit Hilfe des `plumChip`–Befehls 'e' lässt sich anzeigen, welche Ergebnisse die drei Schieberegister nach jeweils einer Aktivierung liefern. Das Ergebnis wird gebildet, indem die Zustände bestimmter Flipflops mit XOR verknüpft werden, **bevor** alle Flipflops ihre Zustände in Richtung zum höchstwertigen Flipflop (nach links) weiterschieben. Danach wird das Ergebnis als Zustand des 0. Flipflops eingesetzt. Als Ergebnisse der einzelnen Schieberegister wird vom `plumChip` bei den ersten 32 Aktivierungen geliefert:

```
LFSR-1: 0000 1111 0110 0011 1110 1010 0011 0111
LFSR-2: 1101 1010 0101 1011 0101 0000 0100 0111
LFSR-3: 0011 1110 1001 1101 1101 0100 0101 1010
```

Die Kontrolle dieser `plumChip`–Ergebnisse lässt sich rechnerisch von Hand oder durch eine Simulation der Schieberegister mit einem Programm durchführen.

Schieberegister 1

Flipflop

60 59 45 44

0 1 101 0110 0001 01 1 0 0100 0111 0101 0110 0001 0110 1110 0110 1111 0100 0100

Bei der händischen, rechnerischen Kontrolle notiert man sich zunächst die Zustände der Flipflops nach dem Eintragen der Saat und markiert diejenigen Flipflops, deren Zustände mit XOR zum Ergebnis verknüpft werden sollen. Beim Schieberegister 1 sind das die Flipflops 60, 59, 45 und 44 (siehe oben).

Dann berechnet man das Ergebnis (hier: 0 XOR 1 XOR 1 XOR 0 = 0), schiebt alle Zustände um eine Stelle nach links, wobei der vorderste wegfällt, und fügt das Ergebnis hinten an (im Folgenden gelb markiert):

1 1 01 0110 0001 011 0 0 100 0111 0101 0110 0001 0110 1110 0110 1111 0100 0100 0

1 0 1 0110 0001 0110 0 1 00 0111 0101 0110 0001 0110 1110 0110 1111 0100 0100 00

0 1 0110 0001 0110 0 1 0 0 0111 0101 0110 0001 0110 1110 0110 1111 0100 0100 000

1 0 110 0001 0110 01 0 0 0111 0101 0110 0001 0110 1110 0110 1111 0100 0100 0000

0 1 10 0001 0110 010 0 0 111 0101 0110 0001 0110 1110 0110 1111 0100 0100 0000 1

1 1 0 0001 0110 0100 0 1 11 0101 0110 0001 0110 1110 0110 1111 0100 0100 0000 11

1 0 0001 0110 0100 0 1 1 1 0101 0110 0001 0110 1110 0110 1111 0100 0100 0000 111

und so fort.

Auf diese Weise erhält man die Ergebnisse, die mit den `plumChip`–Ergebnissen des `LFSR-1` zu vergleichen sind. Diese und auch die weitere Rechnung zeigte keine Unterschiede zwischen den theoretischen und praktischen Resultaten.

Auch die Nachbildung des Schieberegisters 1 durch ein Pascal–Programm bestätigte die korrekte Funktion des `LFSR-1` des `plumChips`, indem es die Ausgabe 0000111101100011111010100011 0111 lieferte.

```
 1  function lfsr61takt: boolean;
 2  var
 3    i : integer;
 4    b : boolean;
 5  begin
 6    b := lfsr61[60] xor lfsr61[59] xor lfsr61[45] xor lfsr61[44];
 7    for i := 60 downto 1 do lfsr61[i] := lfsr61[i-1];
 8    lfsr61[0] := b;
 9    lfsr61takt := b
10  end;
```

In Zeile 6 wird das Ergebnis des Schieberegisters berechnet. Zeile 7 sorgt für das Verschieben der Zustände, Zeile 8 für das Anfügen des Ergebnisses als Zustand des 0. Flipflops. Zeile 9 stellt dem aufrufenden Programm das Funktionsergebnis bereit. Die globale Variable `lfsr61` ist als ARRAY OF BOOLEAN vereinbart, der Zustand '0' eines Flipflops wird als Wahrheitswert `FALSE`, der Zustand '1' als Wahrheitswert `TRUE` aufgefasst.

Schieberegister 2

Flipflop

72 47

$\boxed{1}$ 0110 0110 0110 0110 0110 0110 $\boxed{0}$110 1001 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000

Bei der Kontrolle des Schieberegisters 2, bei dem die Zustände der Flipflops 72 und 47 mit XOR zum Ergebnis zusammengerechnet werden, geht man händisch genauso wie beim Schieberegister 1 vor.

$\boxed{0}$ 110 0110 0110 0110 0110 0110 0$\boxed{1}$10 1001 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000 1

$\boxed{1}$ 10 0110 0110 0110 0110 0110 01$\boxed{1}$0 1001 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000 11

$\boxed{1}$ 0 0110 0110 0110 0110 0110 011$\boxed{0}$ 1001 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000 110

$\boxed{0}$ 0110 0110 0110 0110 0110 0110 $\boxed{1}$ 001 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000 1101

$\boxed{0}$ 110 0110 0110 0110 0110 0110 1$\boxed{0}$ 01 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000 11011

$\boxed{1}$ 10 0110 0110 0110 0110 0110 10$\boxed{0}$ 1 0110 1000 0110 0011 0111 0011 0110 0110 0111 0000 110110

und so fort. Das stimmt mit den `plumChip`–Ergebnissen überein.

Ein Pascal-Programm, welches analog zu oben das Schieberegister 2 nachbildet, lieferte die Ausgabe 110110100101101101010000010001 11, womit die korrekte Arbeitsweise des `LFSR-2` des `plumChips` belegt sei.

Schieberegister 3

Flipflop

88 50

$\boxed{0}$ 0110 0101 0110 0001 0111 0100 0110 1001 0111 0$\boxed{0}$00 0110 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000

Zur Kontrolle der fehlerfreien Funktion des dritten Schieberegisters LFSR-3 sind die Zustände der Flipflops 88 und 50 mit XOR zum Ergebnis zu verknüpfen.

0 110 0101 0110 0001 0111 0100 0110 1001 0111 00 0 0 0110 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000 0

1 10 0101 0110 0001 0111 0100 0110 1001 0111 000 0 0110 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000 00

1 0 0101 0110 0001 0111 0100 0110 1001 0111 0000 0 110 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000 001

0 0101 0110 0001 0111 0100 0110 1001 0111 0000 0 1 10 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000 0011

0 101 0110 0001 0111 0100 0110 1001 0111 0000 01 1 0 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000 00111

1 01 0110 0001 0111 0100 0110 1001 0111 0000 011 0 0001 0110 1011 0111 0011 0111 0100 0111 0010 0110 1000 001111

und so fort. Auch diese Folge an von Hand gerechneten Ergebnissen stimmt mit den `plumChip`–Ergebnissen überein. Das Gleiche gilt für die Simulation mit dem zugehörigen Pascal-Programm, das die Ausgabe 00111110100111011101010001011010 liefert.

Zusammenfassung: Die drei Schieberegister des Zufallsgenerators des `plumChip`s arbeiten bei den ersten 32 Aktivierungen exakt genauso wie durch Rechnung und Simulation ermittelt. Das begründet die Annahme, dass die drei Schieberegister auch bei größeren Anzahlen an Aktivierungen und anderen Zufallssaaten verlässlich und fehlerfrei funktionieren.

Kontrolle der Funktion des Stop–and–go–Generators

Die Ausgaben des alternierende Stop–and–go–Generators des `plumChip`s, der aus den Schieberegistern `LFSR-1`, `LFSR-2` und `LFSR-3` wie in Abb. 1 zusammengesetzt ist, lassen sich mit den `plumChip`–Befehlen 'b' und 'd' beobachten. Alle Beispiele wurden mit der oben angegebenen Zufallssaat durchgerechnet beziehungsweise getestet und simuliert. Der Generator des `plumChip`s liefert als erste dreißig Zufallsbits:

000 111 001 000 110 101 101 011 100 010

Die Kontrolle der Korrektheit dieser Ausgaben erfolgt wiederum händisch durch Rechnung und durch Simulation mit einem Programm. Zunächst sei festgehalten, in welcher Reihenfolge der Generator End- und Zwischenergebnisse bestimmt.

1. Schritt: Die mit XOR verknüpften Zustände der 0. Flipflops von `LFSR-2` und `LFSR-3` werden berechnet und bereitgestellt (Zeile 3).
2. Schritt: Je nach Zustand des 0. Flipflops des `LFSR-1` werden entweder die Zustände des `LFSR-2` oder `LFSR-3` weitergeschoben (Zeile 4).
3. Schritt: Die Zustände des `LFSR-1` werden weitergeschoben (Zeile 5).

```
1  function zufbitplum: boolean;
2  begin
3    zufbitplum := lfsr73[0] xor lfsr89[0];
4    if lfsr61[0] then lfsr73takt else lfsr89takt;
5    lfsr61takt
6  end;
```

Die Simulation mit einem Programm in Pascal (mit dem Unterprogrammtext für den Generator wie oben rechts abgedruckt) liefert 00011100100011010110101110001 0 und zeigt also keine Unterschiede zur Ausgabe des `plumChip`–Generators.

Zur Kontrolle durch Rechnung per Hand lässt sich eine Tabelle anlegen, in deren erste Spalte man den Zustand der 0. Flipflops der Schieberegister (hier orange markiert) und in deren oberster Zeile man ab der zweiten Spalte die Ausgaben des `LFSR-1` einträgt, wie sie oben bereits berechnet wurden. In die Ergebniszeile schreibt man die mit XOR verknüpften Einträge aus den Zeilen `LFSR-2` und `LFSR-3`.

LFSR-1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	1
LFSR-2	0																							
LFSR-3	0																							
Ergebnis	0																							

Für jede Eins, die das LFSR-1 liefert, überträgt man in die Folgespalte die nächste Ausgabe vom LFSR-2, wie oben bereits berechnet (1101 1010 0101 1011 0101 0000 0100 0111), für jede Null überträgt man die nächste Ausgabe des LFSR-3, wie ebenfalls oben berechnet (0011 1110 1001 1101 1101 0100 0101 1010):

LFSR-1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	1
LFSR-2	0																							
LFSR-3	0	0	0	1	1	1																		
Ergebnis	0	0	0	1	1	1																		

Nun zeigt das LFSR-1 Einsen an, also wird ab der nächsten Spalte die `LFSR-2`–Zeile mit den `LFSR-2`–Ausgaben befüllt:

LFSR-1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	1
LFSR-2	0						1	1	0	1														
LFSR-3	0	0	0	1	1	1																		
Ergebnis	0	0	0	1	1	1	0	0	1	0														

Man beachte, dass in die Zeile mit den Ergebnissen die geXORten beiden letzten Ausgaben der `LFSR-2`– und `LFSR-3`–Zeile eingetragen werden. Steht hier also beispielsweise zum Schluss eine 1 in der `LFSR-2`–Zeile, dann ist in die Zelle darunter gedanklich eine 1 einzusetzen, die aus Gründen der Übersichtlichkeit hier nicht hingeschrieben wurde.

Nun kommt wieder eine Null, also geht es ab der nächsten Spalte mit der `LFSR-3`–Zeile weiter:

LFSR-1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	1
LFSR-2	0						1	1	0	1														
LFSR-3	0	0	0	1	1	1					1													
Ergebnis	0	0	0	1	1	1	0	0	1	0	0													

Nach dreißig Takten sieht die Tabelle dann wie folgt aus:

LFSR-1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	1
LFSR-2	0						1	1	0	1		1	0				1	0	0	1	0		1	
LFSR-3	0	0	0	1	1	1					1			1	0	1						0		0
Ergebnis	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	1	0	1	1	0	1	0	1	1

LFSR-1	0	0	0	1	1	0	1
LFSR-2	1				0	1	
LFSR-3		1	1	1			0
Ergebnis	1	0	0	0	1	0	

Der Ergebniszeile lässt sich die Zufallsbitfolge 000 111 001 000 110 101 101 011 100 010 entnehmen, die mit den Ergebnissen von `plumChip` und der Simulation durch ein Programm übereinstimmt.

Zusammenfassung: Auch vom Zufallsgenerator des `plumChip` ist anzunehmen, dass er seine korrekte Funktion auch für große Anzahlen an Zufallszahlen erfüllt.

Befüllen einer A–Matrix, so dass sie spaltenstarr wird

Es soll beschrieben werden, wie der `plumChip`–Befehl 'z' mit Hilfe des `plumChip`–Zufallsgenerators eine spaltenstarre Matrix generiert.

Wie gerade berechnet, liefert der Zufallsgenerator des `plumChip` nach Eintragen der eingangs genannten Saat die Zufallsbits 000 111 001 000 110 101 101 011 100 010. Zum Erzeugen einer spaltenstarren 8x8–Matrix werden je drei Zufallsbit durch den `plumChip` in die Zahlen 0, 7, 4, 0, 3, 5, 5, 6, 1, 2 gewandelt (niedrigwertigstes Bit zuerst). Da diese zehn Zahlen für das Befüllen der gesamten Matrix noch nicht reichen, wurden vom Zufallsgenerator mit dem beschriebenen Vorgehen weitere erzeugt: 3, 5, 6, 7, 1, 5, 6, 1, 3, 7, 3, 2, 5, 0, 6, 2, 6, 4, 7, 4, 0, 4, 4, 4, 6, 4, 3, 6, 6, 3, 2, 5. Mit diesen Zufallszahlen werden zunächst für die erste Spalte der Matrix diejenigen Positionen der Matrix bestimmt, in denen die vier Einsen stehen sollen. Dafür werden die Zufallszahlen 0, 7, 4, 0, 3 gebraucht, also fünf Zahlen, weil eine davon doppelt auftaucht.

	0	1	2	3	4	5	6	7
0	1							
1								
2								
3	1							
4	1							
5								
6								
7	1							

Für die zweite Spalte werden die Zufallszahlen 5, 5, 6, 1, 2 benötigt und so fort.

Mit den weiteren Zufallszahlen füllt der 'z'–Befehl dann die restlichen Spalten in gleicher Weise und liefert zum Schluss:

	0	1	2	3	4	5	6	7
0	1					1	1	
1		1		1				
2		1			1	1		1
3	1		1	1	1			1
4	1					1	1	1
5		1	1	1	1			
6		1	1	1		1	1	1
7	1		1		1		1	

Aufgabe

Um zu überprüfen, ob der 'z'–Befehl auch für eine 128x128–Matrix so arbeitet wie oben erläutert, wird im Anhang dieses Dokuments eine ebenfalls mit der Saat `Donaudampfschifffahrtskapitaen` erzeugte Matrix dieser Größe in hexadezimaler Darstellung so abgedruckt, wie sie die Version v2.4 zur Anzeige bringt. Man teste durch Simulation, ob die Matrix durch den 'z'–Befehl fehlerfrei generiert wurde.

Anhang

```
 0: DB 87 17 59 0E 63 87 09 C8 A6 F9 32 04 D0 FB 1B  ...Y.c.....2.... [62]
 1: C4 6A CC 30 BA AF 19 C6 20 A4 87 F0 36 68 E1 94  .j.0.... ...6h.. [57]
 2: 35 09 34 6A 89 FA C0 49 8A 9D FB 66 7A 6A F7 EB  5.4j...I...fzj.. [68]
 3: FC 2F 39 AA 4D C2 FF C4 D3 69 C3 86 56 FD 52 4F  ./9.M....i..V.RO [72]
 4: 5E 21 87 35 D4 0E B3 08 CA 71 49 60 0F DB 54 EF  ^!.5.....qI`..T. [61]
 5: 1C D7 34 2B 14 8E BA D4 48 EE 4A 4A F9 6A 46 59  ..4+....H.JJ.jFY [62]
 6: 73 11 9D E5 60 11 4D 30 7E C0 DD 0E 98 AF BA 9A  s...`.M0~....... [62]
 7: 62 7B 9A 9E 29 F8 AD A7 CC 8C 59 FB A5 33 6C 56  b{..).....Y..3lV [70]
 8: 38 E4 32 7D D2 86 2D 84 CF FB 04 C5 A2 BD 41 F4  8.2}..-.......A. [63]
 9: C3 92 FE 04 96 CC F4 1F 6C 64 06 5E 1B 27 6D E2  ........ld.^.'m. [64]
10: 55 FA 8D D0 83 DF B3 E5 3C 80 8D 48 77 DF B9 DF  U.......<..Hw... [73]
11: 1D 65 5F BE FC 20 10 E6 39 98 D3 69 7C 3E 87 73  .e_.. ..9..i|>.s [68]
12: 6D 62 09 14 A4 84 40 6A 89 0E 38 31 A3 91 98 F9  mb....@j..81.... [50]
13: 83 BC 14 06 51 38 61 42 5E E1 C6 8E D2 A1 C3 EB  ....Q8aB^....... [57]
14: CB 98 9C CB 00 E2 A2 AD CD C4 63 F6 EE FA 9B 54  ..........c....T [67]
15: 48 4C 80 62 BA 88 4A 1E 7C E1 40 AC 94 CB D0 AB  HL.b..J.|.@..... [53]
16: 95 49 92 93 B9 E1 93 35 06 EA 1E 0D E8 AC 39 C6  .I.....5......9. [61]
17: EC FB E2 57 D1 D0 9F 15 0B AA 76 4E 81 C0 B1 C2  ...W......vN.... [64]
18: 97 68 7C 85 FF 6F 1D 81 22 3F E0 AF 23 CF 92 CD  .h|..o.."?..#... [70]
19: 6E 05 F7 A9 FE F3 0C E8 52 68 8E F4 73 11 AA 12  n.......Rh..s... [65]
20: 0C C9 BE 2E 48 4B 02 E7 27 CD 5F C5 7E D3 78 37  ....HK..'._.~.x7 [68]
21: BF BB C9 E0 00 76 DE FC 20 07 7C 94 DD 2A 8F 1F  .....v.. .|..*.. [68]
22: 64 DE 08 19 BC 65 47 9B 28 32 60 B1 2D 00 C1 8D  d....eG.(2`.-... [53]
23: 1F 26 A7 40 F1 17 3F A4 78 A4 3E 6D 51 D7 06 C2  .&.@..?.x.>mQ... [63]
24: A0 ED DD 5A B1 8D FD 66 DD 1B 16 E5 44 71 88 07  ...Z...f....Dq.. [66]
25: F2 91 A9 8F 0D 2B 48 69 05 9D E1 60 71 E2 86 CB  .....+Hi...`q... [59]
26: 58 01 8D EE 09 65 54 EA AF D5 BB CE CE 62 82 AB  X....eT......b.. [65]
27: 3D 0B 76 14 4B 04 5F 31 57 06 55 31 F3 91 90 60  =.v.K._1W.U1...` [56]
28: 88 7E 9A D5 31 81 59 24 13 DB FB F9 7C 2E A0 00  .~..1.Y$....|... [61]
29: 8B D8 D0 9D 5F B4 67 74 E1 FB A2 A7 32 04 C4 4B  ...._.gt....2..K [65]
30: 57 6C 43 37 82 49 A2 CC 44 A2 94 8D 8D 73 8B 6B  WlC7.I..D....s.k [59]
31: B1 E0 E5 88 E2 4C D7 72 F5 4A 48 20 DB 0B 0A BF  .....L.r.JH .... [61]
32: 58 2A B7 EB CD 21 6A 2C 0F F6 28 1B 0C 62 0C A3  X*...!j,..(..b.. [59]
33: A8 EE 23 A6 98 AC 7A 3A E5 D1 ED 2A CD 5E 14 08  ..#...z:...*.^.. [63]
34: 07 D9 7E 0D 56 92 61 DB DA 42 34 3C 75 27 8A 40  ..~.V.a..B4<u'.@ [60]
35: 0F 75 77 08 18 4B A6 DE 20 64 0B 7C 33 9F C5 D9  .uw..K.. d.|3... [63]
36: EC B5 6C 97 32 8F BF 0F A3 3A 8F 9E EF 48 F3 73  ..l.2....:...H.s [76]
37: 4F 27 E5 9B FE E7 E3 18 A6 9A 07 01 4A 94 15 7E  O'..........J..~ [66]
38: 27 68 7C 79 7B 50 14 AD A6 71 A4 9E 9F 35 22 9C  'h|y{P...q...5". [64]
39: D4 30 EA 48 0C BE B9 12 AC BA 38 51 95 8E E8 4E  .0.H......8Q...N [59]
40: D4 32 67 1F 23 94 54 A4 ED 3C BA 7C 6E EE 71 D4  .2g.#.T..<.|n.q. [68]
41: B4 BB A5 F0 4D 05 59 D0 8B 6F B3 18 F9 F0 D7 F5  ....M.Y..o...... [70]
42: 8C 46 48 57 D8 1C 0E 52 82 5E 09 3B DF 10 C4 A1  .FHW...R.^.;.... [54]
43: 7E AA 0E B1 6F F9 93 78 B0 3A 0A 4E 93 14 7F 66  ~...o..x.:.N...f [67]
44: DF 10 7D 5F C7 BF 02 F2 85 BE 39 91 42 6A E7 A6  ..}_......9.Bj.. [70]
45: A5 A9 53 07 FF 59 9A BD AA 83 CB BF F1 1C 64 AC  ..S..Y........d. [71]
46: 52 E2 2B E8 D7 5C 2D D7 F0 93 70 7E 62 DA 5B 43  R.+..\-...p~b.[C [68]
47: AE 6A F3 DF 84 E2 4A 4D 1B F4 75 6B A9 EE 65 E5  .j....JM..uk..e. [73]
48: 5A 4B B5 86 87 54 D0 5D A4 6D 64 CF 26 46 36 38  ZK...T.].md.&F68 [61]
49: 7A FF E1 72 12 5A 45 0C 90 D7 37 FD 86 0D 7A 39  z..r.ZE...7...z9 [67]
50: BA B4 61 E1 6D F1 4A 5B 4C DC D9 7F CD CE F4 DC  ..a.m.J[L....... [74]
51: 32 71 AA 36 B9 C5 6D 50 FA 95 AD 15 D3 7C 1B 4D  2q.6..mP.....|.M [67]
52: 7F 0E C1 F0 26 8E 06 BC 43 D0 94 EE 4C EA 7E 41  ....&...C...L.~A [62]
53: 29 58 DE A7 6D FD 76 FE 97 17 E4 B5 FD 20 96 F4  )X..m.v...... .. [76]
54: D3 2F D8 31 26 6B 6C 97 58 C2 67 F3 21 4F E0 F0  ./.1&kl.X.g.!O.. [65]
55: 64 D0 0B 57 88 1C 80 47 12 26 37 65 6F 51 DA C9  d..W...G.&7eoQ.. [56]
56: 92 0C B4 21 A8 67 A9 6C A1 79 46 C4 83 F0 6D 86  ...!.g.l.yF...m. [56]
```

```
 57: 9C 94 A1 21 73 00 CC 82 25 17 CE E5 01 1D 87 78  ...!s...%......x [53]
 58: 4A D1 12 FF 69 B1 0D 59 FD 88 86 3F A2 67 53 B1  J...i..Y...?.gS. [66]
 59: 1C 51 FF 5B 84 27 13 73 73 1D 54 A3 DE 24 4D 37  .Q.[.'.ss.T..$M7 [66]
 60: D0 C7 B9 F4 F5 D0 E1 B0 5F 32 67 40 34 DB E1 58  ........_2g@4..X [65]
 61: C1 1E 34 2E 11 5C 91 2B 3C A7 9D 8A 78 C5 79 BC  ..4..\.+<...x.y. [62]
 62: A3 6B 55 50 C7 6C AE 88 BD F6 D4 D1 40 17 99 E6  .kUP.l......@... [65]
 63: 9B 9F 9E 9E 02 43 E0 C0 62 09 9E 4A 17 7D D1 FD  .....C..b..J.}.. [64]
 64: DC A0 1B 84 2E 61 0F 67 1C 1E 19 88 AE B8 D7 A7  .....a.g........ [61]
 65: FD D9 2A B9 2F 3F 55 48 CB CC 03 1F 1B 55 19 9A  ..*./?UH.....U.. [68]
 66: 47 B6 46 47 C8 7B FD A7 E6 22 23 21 86 35 95 8D  G.FG.{..."#!.5.. [64]
 67: 69 0D 01 6F EB 1A 17 34 40 33 CF 62 DB 84 94 DE  i..o...4@3.b.... [61]
 68: 8A 23 2A 7A 33 E7 49 E5 8D 13 ED 7D 1F 44 39 3A  .#*z3.I....}.D9: [66]
 69: 3C 47 F4 E8 4B 73 F9 3C 7C 89 0E BE 76 EA 32 C9  <G..Ks.<|...v.2. [70]
 70: 49 16 E6 78 27 98 E1 B1 95 2D 0E B0 9C DF 0F B2  I..x'....-...... [63]
 71: 84 33 52 11 94 AE CA ED 56 28 5D DA DC AF 0C 57  .3R.....V(]....W [63]
 72: A3 8F 22 F2 1D A7 3C 9E 2C 5C 21 6B 82 FD E7 88  .."...<.,\!k.... [65]
 73: B3 ED 59 BC 34 DA BF 87 44 8E 86 E9 D3 A3 7B 22  ..Y.4...D.....{" [70]
 74: EA 67 98 08 EF 5C B9 D0 A2 53 F1 17 84 FD 3F 45  .g...\...S....?E [67]
 75: 1D 56 E3 D3 6E 5B A6 2D B8 9F B3 D9 75 53 0C E3  .V..n[.-....uS.. [72]
 76: 97 87 4F 50 4C 88 FE 18 BC E1 9A 97 49 D1 43 09  ..OPL.......I.C. [60]
 77: E6 27 4B 38 98 B4 F8 96 FE 17 34 96 5D 17 E6 72  .'K8......4.]..r [68]
 78: F1 FC 73 E7 F3 BB 86 27 BA 92 83 EF 01 56 2C 85  ..s....'.....V,. [70]
 79: F1 8C 52 EC D7 FA CB CA DE 2B 26 DA 27 40 A8 1F  ..R......+&.'@.. [68]
 80: DE D6 81 E2 B1 31 A8 F7 DC F1 72 28 31 96 5E A6  .....1....r(1.^. [66]
 81: A5 E3 4E 44 74 A7 ED BA 43 69 62 85 DD FB 1F 9E  ..NDt...Cib..... [71]
 82: 59 B6 B8 46 7D 97 16 E5 26 6D FD 03 34 CB F8 3B  Y..F}...&m..4..; [70]
 83: 65 28 E3 5B 4F A7 33 53 62 4B 28 E5 AE 56 79 38  e(.[O.3SbK(..Vy8 [65]
 84: B0 46 5E A3 63 1F 82 42 D5 34 E9 F2 69 AC 65 27  .F^.c..B.4..i.e' [62]
 85: EC E6 F7 91 EF D1 30 8F 56 B9 60 C0 32 5D EB AE  ......0.V.`.2].. [70]
 86: 97 B3 89 A5 30 75 4E FE B3 47 B1 E2 A0 21 40 B4  ....0uN..G...!@. [61]
 87: 0A 4F 64 E7 B2 73 9B B9 59 DA F3 01 98 F1 72 30  .Od..s..Y.....r0 [65]
 88: DB D0 05 90 BA BE 4D 8C 1D 78 1E D3 42 86 A4 67  ......M..x..B..g [61]
 89: 0A 11 79 E7 20 1A EE FA 09 A2 D0 77 35 02 00 E4  ..y. ......w5... [54]
 90: B1 B5 99 B9 B2 2C 93 E9 51 FB 98 03 A5 30 35 34  .....,..Q....054 [62]
 91: 43 72 63 11 F9 3A E2 5B 55 D4 47 28 F5 7C 17 20  Crc..:.[U.G(.|.  [62]
 92: 6A 48 00 7D 06 BF 6A 82 3D 47 D3 00 ED A1 60 4B  jH.}..j.=G....`K [56]
 93: 31 9C C2 EA 42 64 A6 B1 F9 35 5C 11 E1 EA 7C 70  1...Bd...5\...|p [61]
 94: 63 8A 32 88 07 37 DC EE 62 AE AD 32 EB 3A DE 3C  c.2..7..b..2.:.< [67]
 95: A2 35 CD 37 93 0E 64 50 19 F9 F6 EA B4 99 6C D1  .5.7..dP......l. [65]
 96: 17 FE D6 84 BF 03 A5 23 5B 6D 12 EE 3A C0 AA F2  .......#[m..:... [67]
 97: 2A 40 35 7C 01 D8 82 33 21 F0 EE 92 96 B7 04 00  *@5|...3!....... [50]
 98: 94 4A 99 D9 BE 61 4D 60 39 6B 0A 06 A2 9B 77 29  .J...aM`9k....w) [60]
 99: 62 5D 2F 98 18 F2 11 5B 02 91 19 F7 00 46 BF 4D  b]/....[.....F.M [58]
100: F9 2F C4 AA B9 D0 36 9B E9 4D A1 07 B8 51 EB 91  ./....6..M...Q.. [66]
101: 67 C7 CF 0B BA B9 4E 17 9C 38 CE 19 65 97 A0 A1  g.....N..8..e... [66]
102: CD 0B 5E 5B 64 05 91 BF 5B 63 B4 0B 06 92 3D 16  ..^[d...[c....=. [62]
103: EF F3 88 27 06 12 95 EB B6 1C 19 CB 00 7D 97 C5  ...'.........}.. [64]
104: 20 8E 29 6B D6 F4 6F 5F 26 47 8A 82 A1 B8 5B 3F   .)k..o_&G....[? [65]
105: E5 84 81 5E 65 26 74 41 8B D7 C9 45 D4 8A 77 27  ...^e&tA...E..w' [61]
106: B3 11 32 32 7C CF 91 C0 E1 00 31 5B 8A 2C 2E D3  ..22|.....1[.,.. [56]
107: 26 F8 26 23 98 8D 25 6A 82 55 90 51 48 3F 82 3D  &.&#..%j.U.QH?.= [54]
108: 65 D5 F8 53 FF AB EE 24 A9 EB DE B5 C5 19 5A D8  e..S...$......Z. [75]
109: F0 96 1D 1E F6 66 DE 22 D0 F5 38 10 11 96 AF CC  .....f."..8..... [63]
110: A9 E5 7E 8F 11 89 75 AD 76 02 DF 6D 4A 09 A9 94  ..~...u.v..mJ... [65]
111: 0C 31 1D 84 54 86 7C 53 D7 99 D1 B6 6C B0 2D 59  .1..T.|S....l.-Y [60]
112: 5B 7C 9C CE D6 37 4A 8A 1B 55 76 3A 7E AE 4B E6  [|...7J..Uv:~.K. [72]
113: D2 19 85 48 17 C6 BC 3B 9B 54 DD 34 19 29 CF 3E  ...H...;.T.4.).> [64]
114: 21 50 A0 16 DA E9 C9 C7 73 6F AC D3 03 46 1C 32  !P......so...F.2 [59]
115: 18 18 66 C6 04 7C FA 3F EF 0C 42 FE E8 8E 2C 3A  ..f..|.?..B...,: [63]
```

```
116: 21 D4 C8 64 EF 41 3B 15 F9 BE E4 88 5A AC D7 0D  !..d.A;.....Z... [64]
117: 4A 0C 7D D5 0C 2B FB 79 F7 C0 E7 B8 7E 44 98 15  J.}..+.y....~D.. [67]
118: 5F 2F 4D AC 73 68 70 6E 2D 2E 29 85 AB 34 EC DA  _/M.shpn-.)..4.. [67]
119: 41 C4 66 BC CD 98 51 D4 23 AF 7B DB 7A 2D C0 C0  A.f...Q.#.{.z-.. [63]
120: 3F A3 D9 D9 49 07 32 93 54 A8 F9 9C 8E E3 F8 3F  ?...I.2.T......? [69]
121: 1C B3 82 8F CA 50 F5 17 87 48 A0 16 8D 8D 1B 35  .....P...H.....5 [58]
122: F7 98 53 EB 6D 18 A2 49 A1 90 6F 48 BD CA FF 0C  ..S.m..I..oH.... [66]
123: 8C EA 0A 4D CA 49 AB 29 D6 45 67 F9 73 09 10 62  ...M.I.).Eg.s..b [59]
124: 02 DE F2 24 A9 D0 14 AD 77 97 A8 25 08 A1 AA CD  ...$....w..%.... [58]
125: A4 B1 AC 70 63 D4 00 5A CF 65 7B 6C 8E F9 87 CF  ...pc..Z.e{l.... [66]
126: A6 1D 73 ED F3 FB B3 95 3E 76 1F DE FB 77 36 18  ..s.....>v...w6. [81]
127: 12 FF AE EA C4 BE 14 92 D6 8A 94 14 DE 61 25 0A  .............a%. [61]

Spaltensummen:
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64

Antwortzeit: 0 s
```

Quellen

[Schneier15] Bruce Schneier: „Applied Cryptography, Second Edition — Protocols, Algorithms, and Source Code in C — 20th Anniversary Edition“, John Wiley & Sons, Indianapolis 2015, ISBN 978-1-119-09672-6

[Dierks19-7] imbit/Andreas Dierks: „LFSR für plumChip v2.0“, 30. Oktober 2019, imbit, Hildesheim 2019

Autor: a.dierks@imbit.net

Stand: 04.05.2020