Teil 3 One-Time-Pads, lineare Schieberegister

Eine Schlüsselwortchiffre ist umso sicherer, je länger das Schlüsselwort ist. Am besten hätte das Schlüsselwort unendliche Länge! Aber je länger das Schlüsselwort ist, um so problematischer gestaltet sich auch der Schlüsselaustausch zwischen Alice und Bob. Als Ausweg bietet sich an, die Buchstaben etwa eines dicken Romans als quasi unendlich langes Schlüsselwort zu nehmen. Dann bräuchte Alice nicht den ganzen Roman an Bob zu schicken, sondern es würde eine Botschaft reichen wie: Eco, Pendel, ab S.51 oben links.

Realistischer als Verschlüsselung mittels eines vorgegebenen Textes ist es, eine beliebig lange und völlig zufällige Folge von Buchstaben zu erzeugen und den Klartext mit Hilfe dieser Buchstaben zu verschlüsseln. Einen solchen **Buchstabenwurm** kann man beispielsweise erzeugen durch ständig wiederholtes Werfen eines *fairen* 26-seitigen Buchstabenwürfels.
Bei einer Botschaft, die mit einem Buchstabenwurm verschlüsselt wurde, können keinerlei statistische Attacken Erfolg haben.
Wenn Alice und Bob über einen gemeinsamen und geheimen Buchstabenwurm verfügen, so ist ein solches System in der Tat absolut sicher!

Auf Buchstabenwürmern basierende Kryptosysteme sind die einzigen bekannten völlig sicheren Systeme. Sie werden heutzutage benutzt in der Form von **One-Time-Pads:**
Statt der Buchstaben eines Alphabets wählt man dabei die Bits 0 und 1. Als Schlüssel braucht man eine zufällige und unbeschränkt lange Bitfolge.
Verschlüsselt wird einfach mittels binärer Addition:

Geheimtext = Klartext + Schlüssel

wobei der Operator "+" die **Boolesche Addition**

0+0=0 , 1+0=1 , 0+1=1 , 1+1 = 0

bedeutet.

Der Name *One-Time-Pad* hat seinen Grund in der Analogie zum Abreißkalender: Jeder Zettel wird genau einmal benutzt und dann weggeworfen.

Beispiel

```
Klartext   01100 10001 1001 ...  |
Schlüssel  10100 11001 1010 ...  | +
-----------------------------
Geheimtext 11000 01000 0011 ...
```

Beim One-Time-Pad ist das Entschlüsseln für Bob besonders einfach.

Er entschlüsselt genauso, wie Alice verschlüsselt hat, also

Schlüssel + Geheimtext = Klartext (warum stimmt das?)

Im Beispiel:

```
Geheimtext 11000 01000 0011 ...  |
Schlüssel  10100 11001 1010 ...  | +
--------------------------------
Klartext   01100 10001 1001 ...  (ok)
```

Chiffren wie das One-Time-Pad, bei denen Chiffrierschritt und Dechiffrierschritt übereinstimmen, werden **involutorisch** genannt.

Boolesche Arithmetik

Neben boolescher Addition brauchen wir später auch **boolesche Multiplikationen**, weswegen wir alle benötigten Regeln hier zusammenstellen:

0+0=0 , 1+0=1 , 0+1=1 , 1+1=0

0*0=0 , 1*0=0 , 0*1=0 , 1*1=1

Offensichtlich gelten die folgenden Regeln für Boolesche Arithmetik:

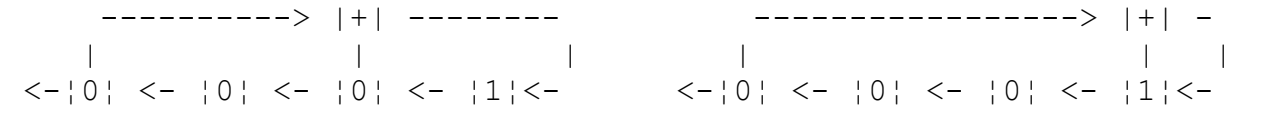
x+x = 0 , x*x = x , x ∈{0,1}

Beim One-Time-Pad ist der Schlüssel ein zufälliger binärer String, der genauso lang sein muß wie die Nachricht. Das Problem für Bob und Alice ist das Übermitteln dieses unbeschränkt langen Binärstrings.
Praktischer wäre es, statt echter Zufalls-Strings nur *pseudozufällige* binäre Strings als Schlüssel zu benutzen. Diese pseudozufälligen Bits kann man sich durch einen Generator erzeugen lassen, so daß Alice und Bob sich nur über die (wenigen) Kenngrößen dieses Generators verständigen müssen.
Natürlich erkauft man sich diese Bequemlichkeit mit der Gefahr, daß der Angreifer hinter das Bildungsgesetz der Pseudozufallszahlen kommen könnte...

Ein einfacher Mechanismus zur Erzeugung von pseudozufälligen Bits sind Schieberegister. An **linearen Schieberegistern** der Länge n=4 soll der Mechanismus demonstriert werden.

Man hat vier Zellen, jede enthält ein Bit. Bei jeder Iteration wird das Register um eine Zelle nach rechts geschoben. Das *links freiwerdende* Bit ist das nächste Bit des Schlüssels.

Die rechts leergewordene Zelle muß neu gefüllt werden. Dies geschieht durch boolesche Addition der Inhalte der *markierten* Zellen.
Zum Starten des Schieberegisters braucht man eine Initialisierung. Dies sind die in den Zellen stehenden Bits:



Die resultierenden Zustände sind nun der Reihe nach:

0	0	0	1					0	0	0	1				
0	0	1	0					0	0	1	1				
0	1	0	1					0	1	1	1				
1	0	1	0					1	1	1	1				
0	1	0	0					1	1	1	0				
1	0	0	0	Periode 6				1	1	0	1				
0	0	0	1					1	0	1	0				
.								0	1	0	1				
.								1	0	1	1				
.								0	1	1	0				
								1	1	0	0				
								1	0	0	1				
								0	0	1	0				
								0	1	0	0				
								1	0	0	0	Periode 15			
								0	0	0	1				
								.							
								.							

Die Schlüsselbits sind jeweils die Zahlen der linken Kolonne, im ersten Beispiel also

0 0 0 1 0 1 0 ...

und dann periodisch(!) so weiter.

Man beachte: Bei linearen Schieberegistern darf der Nullzustand nie vorkommen, da man von dem nie wieder *wegkommt*.

Es fällt auf, daß die Schlüsselbits keineswegs völlig zufällig sind, sondern sich periodisch wiederholen.
Das kann anders auch nicht sein: Jedes Register hat nur endlich viele unterschiedliche innere Zustände, in diesen Beispielen (Registerlänge n=4) 2^4=16 Stück. Damit ist klar, daß der Zustand des Schieberegisters (und damit die Schlüsselfolge) sich periodisch wiederholen *muß*.
Im ersten Beispiel sind 6 Zustände unterschiedlich, im zweiten Beispiel sind dies 15. Letzteres ist gleichzeitig die maximal mögliche Zahl unterschiedlicher Zustände (warum?).

Man wird natürlich nach solchen Schieberegistern suchen, welche die maximal mögliche Anzahl unterschiedlicher Zustände realisieren. Für Schieberegister der Länge n=100 hätte man dann bis zu

$$2^{100}-1 \approx (2^{10})^{10} \approx (10^3)^{10} = 10^{30}$$

unterschiedliche Zustände des Schieberegisters. Das bedeutet, daß die Bitfolge des Schlüsselstrings sich erst ab der Nummer 10^30 , d.h. praktisch nie wiederholt (eine Festplatte mit 100 GB Kapazität enthält 8*10^11 < 10^12 Bits).

Man sollte meinen, das ist Zufall genug: Die Pseudozufallszahlen sind dann praktisch genauso zufällig sind wie echte Zufallszahlen, und der Angreifer hat keinerlei Möglichkeit hat, hinter das Bildungsgesetz der Pseudozufallszahlen zu kommen.
Dies ist jedoch ein eklatanter Trugschluß! Wir werden nämlich zeigen, daß ein lineares Schieberegister durch einen sogenannten *Known-Plaintext-Angriff* auf ganz simple Weise gebrochen werden kann.

Genauer: Hat das Schieberegister z.B. die Länge n=100, so genügt es dem Angreifer schon, 2*n = 200 Klartextbits nebst zugehörigem Geheimtext zu kennen, um das benutzte Schieberegister (d.h. Startbesetzung der Zellen und innere *Verschaltung*) komplett zu enträtseln!

Am Beispiel eines linearen Schieberegisters der Länge n=4 soll dieser **Known-Plaintext-Angriff** vorgeführt werden:
Angenommen, der Angreifer hat die ersten 2n=8 verschlüsselten Bits (Geheimtext) aufgefangen, und er kennt auch deren Bedeutung (Klartext):

```
Klartext  :  1 1 0 0 1 0 1 1
Geheimtext:  1 1 0 1 0 1 0 1
```

Nach der Regel

Schlüssel = Klartext + Geheimtext (gilt warum?)

kann er daraus zunächst die ersten 8 Bits der Schlüsselfolge ausrechnen:

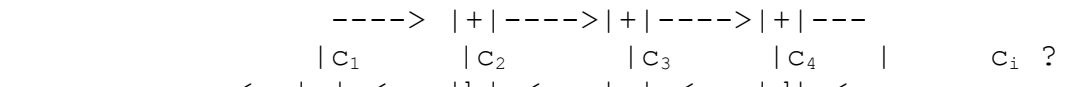
```
Schlüsselbits:  0 0 0 1 1 1 1 0
                a b c d A B C D
```

Dies sind also (von links nach rechts) die ersten 8 vom Schieberegister gelieferten Bits. Damit ist bereits klar, was die Initialisierung ist, nämlich

```
<-¦0¦ <- ¦0¦ <- ¦0¦ <- ¦1¦      -> Initialisierung
   a      b      c      d
```

Was der Angreifer nun noch herausbekommen muß, ist die *Schaltung*

des Registers:

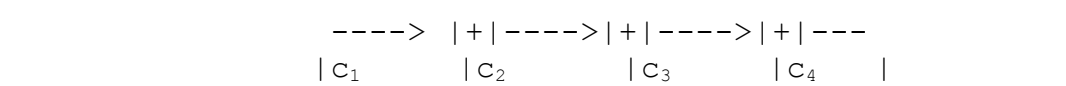


Zu bestimmen sind die Binärvariablen $c_i \in \{0,1\}$:

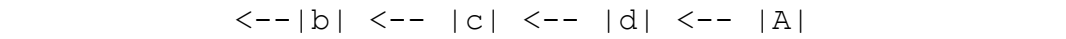
$c_i = 1$ bedeutet: die Zelle i trägt zur Summe bei
$c_i = 0$ bedeutet: die Zelle i trägt zur Summe nicht bei

Wir notieren, was wir wissen.
Die ersten 2n=8 Schlüsselbits : a b c d A B C D
erste Besetzung der Registerzellen : a b c d
zweite Besetzung der Registerzellen : A B C D <- nach vier Rotationen

Die Startsituation ist:



Nach einer Iteration ist der Zustand der Zellen:

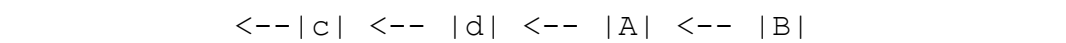


und es gilt

$$a*c_1 + b*c_2 + c*c_3 + d*c_4 = A$$

[Man muß a addieren, falls c_1=1, d.h. man muß $a*c_1$ addieren!]

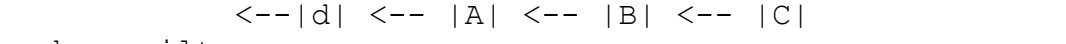
Nach zwei Iterationen ist der Zustand der Zellen:



und es gilt

$$b*c_1 + c*c_2 + d*c_3 + A*c_4 = B$$

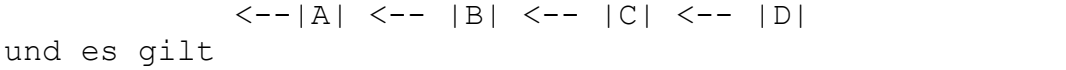
Nach drei Iterationen ist der Zustand der Zellen:



und es gilt

$$c*c_1 + d*c_2 + A*c_3 + B*c_4 = C$$

Und nach vier Iterationen ist der Zustand der Zellen:



und es gilt

$$d*c_1 + A*c_2 + B*c_3 + C*c_4 = D$$

Zusammengefaßt erhalten wir das **boolesche Gleichungssystem**:

$$\textbf{(BGS)} \quad \left| \begin{array}{l} a*c_1 + b*c_2 + c*c_3 + d*c_4 = A \\ b*c_1 + c*c_2 + d*c_3 + A*c_4 = B \\ c*c_1 + d*c_2 + A*c_3 + B*c_4 = C \\ d*c_1 + A*c_2 + B*c_3 + C*c_4 = D \end{array} \right| \; , \; c_1,c_2,c_3,c_4 \; ?$$

mit den bekannten Größen a b c d A B C D und den Unbekannten c_1, c_2, c_3, c_4 .

Lösen kann man dieses Gleichungssystem mit einer auf Boolesche Verhältnisse angepaßten Variante des Gaußschen Algorithmus. - Wir nehmen unsere obigen Daten:

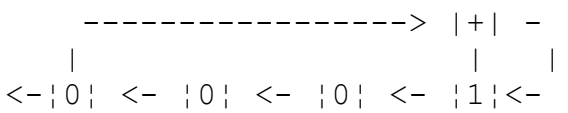
a b c d A B C D = 0 0 0 1 1 1 1 0

und setzen in das Gleichungssystem ein:

c_1	c_2	c_3	c_4		
0	0	0	1	|	1
0	0	1	1	|	1
0	1*	1	1	|	1
1	1	1	1	|	0
0	0	0	1	|	1
0	0	1*	1	|	1
0	**1**	1	1	|	1
1	0	0	0	|	1
0	0	0	1*	|	1
0	0	**1**	1	|	1
0	**1**	0	0	|	0
1	0	0	0	|	1
0	0	0	**1**	|	1
0	0	**1**	0	|	0
0	**1**	0	0	|	0
1	0	0	0	|	1

also $(c_1, c_2, c_3, c_4) = (1,0,0,1)$

Der Angreifer hat das lineare Schieberegister enträtselt, es ist das von früher bekannte Register mit Periode 15:



Der Known-Plaintext-Angriff auf lineare Schieberegister ist desillusionierend: Lineare Schieberegister halten nicht einmal dieser simplen Attacke stand.
Ein Ausweg ist es, *nichtlineare* Schieberegister zu nehmen.
Bei solchen Registern wird der Wert der linken Zelle nicht nur durch boolesche Addition aktualisiert, sondern es werden kompliziertere (nichtlineare) Berechnungsfunktionen zugelassen.

Solche Register sind für den Angreifer viel schwieriger zu enträtseln als lineare. Sie werden bei allen aktuellen Kryptosystemen benutzt z.B. DES, RIJNDAEL...
Die nichtlinearen Register werden normalerweise tabellarisch angegeben -> *S-Boxen*

Ergänzungen zur Kryptologie, Teil 3
==========

1) Konstruieren Sie ein lineares Schieberegister der Länge 4, das einen Nicht-Null-Zustand in den Nullzustand überführt.

2) Konstruieren Sie ein lineares Schieberegister der Länge 3 mit maximaler Periode.

3) Der folgende Output wurde von einem linearen Schieberegister der Länge 5 erzeugt: 0 0 0 0 1 0 0 0 1 1 .
Rekonstruieren Sie das Schieberegister.

4) Jemand behauptet, der folgende String wäre von einem linearen Schieberegister der Länge 4 erzeugt worden: 0 0 0 0 1 0 0 0
Kommentieren Sie diese Behauptung.

5) Das zum Enträtseln des linearen Schieberegisters aufgestellte Gleichungssystem ist nicht notwendigerweise eindeutig lösbar.
Was bedeutet es, daß zwei Lösungen existieren?
Was bedeutet es, daß gar keine Lösung existiert?