Proseminar Sicherheit

Thema 2.5:
Stromchiffre und (Pseudo)Zufallszahlengeneratoren

R. Schwarzer

Zusammenfassung

In dieser Arbeit werde ich auf die Verschlüsselung von Daten mit Hilfe des Stromchiffreverfahrens unter Verwendung von Pseudozufallszahlengeneratoren eingehen. Dabei werden die Begriffe der Stromchiffre und des Pseudozufallszahlengenerators in Kapitel 2 und 3 geklärt. Unterschiede, Vor- und Nachteile in den einzelnen Stromchiffreverfahren erläutere ich im Kapitel 4. Außerdem gehe ich auf die Stärken und Schwächen ein, die dieses Verfahren mit sich bringt. Das letzte Kapitel zieht Bilanz über die Stromchiffre und über Pseudozufallszahlengeneratoren.

1 Motivation

Will man vertrauliche Daten sicher übertragen, dann ist eine wirkungsvolle Verschlüsselung unerlässlich. Die Stromchiffre bietet eine sichere und sehr schnelle Verschlüsselung. Unter bestimmten Bedingungen ist die Stromchiffre sogar nachweisbar sicher. Das allerdings nur, wenn die Schlüssel sehr lang sind, nur einmal benutzt werden und die Schlüssel wirklich zufällig sind. Dies hat den Nachteil von langen unhandlichen Schlüsseln, die ebenfalls geheimzuhalten sind. Als Lösung wird versucht, einen langen Schlüssel aus einem kurzen Schlüssel zu erzeugen, aber nicht durch bloße Aneinanderreihung des kurzen Schlüssels, sondern auf komplexe Art und Weise.

Die im Zweiten Weltkrieg verwendete deutsche Verschlüsselungsmaschine Enigma (Abb. 1) verwendete drei Rotoren, die sich ähnlich wie ein Kilometerzähler bei jedem Buchstaben weiterdrehten. In jedem Rotor ist eine Permutation des Alphabets fest verdrahtet. Die Permutationen aller drei Rotoren sind hintereinandergeschaltet. Auf diese Weise entsteht eine sehr große Zahl unterschiedlicher Permutationen, und jedes Symbol des Klartextes wird mit einer anderen Permutation verschlüsselt. Als Parameter dieses Verschlüsselungsverfahrens war lediglich die Anfangsstellung der Rotoren zu übermitteln, dieser Parameter stellte den täglich wechselnden kurzen Schlüssel dar.

In heutigen Systemen wird die Arbeit der Rotoren von Pseudozufallszahlengeneratoren übernommen. Sie bilden dabei den Grundstock für die Sicherheit der Daten. Je zufälliger dieser arbeitet, desto sicherer ist die Verschlüsselung. Um so schwerer wird es potenziellen Angreifern gemacht, Daten zu entschlüsseln. Tatsächlich wurde die Enigma sowohl von den Polen als auch von den Engländern recht bald geknackt.

Abb. 1 Rotor-Chiffriermaschine Enigma nach der Erfindung von Arthur Scherbius 1919. Hier die spätere 4-Rotoren-Ausführung für die Marine. Das Steckerbrett an der Frontseite der Enigma ermöglichte weitere Verschlüsselungskombinationen. Die Einführung der 4-Rotoren-Enigma unterband das Mitlesen der Briten von Februar.1942 bis Dezember 1942.

2 Stromchiffre

2.1 Was sind Stromchiffren?

Die Stromchiffre ist eine symmetrische Verschlüsselung. In symmetrischen Verfahren existiert ein Schlüssel, der dem Empfänger und dem Sender der Nachricht bekannt sein muss. Der Klartext der Nachricht wird Bitweise mit Hilfe eines Schlüssels nach einem bestimmten Algorithmus verschlüsselt. Da Bit für Bit nacheinander codiert wird, spricht man von sequentieller Chiffre oder Stromchiffre.

2.2 Funktionsweise von Stromchiffren

Jede Bitfolge eines Zeichens aus dem Klartext wird mit der Bitfolge eines Schlüssels nach einem bestimmten Algorithmus (z.B.: XOR) bitweise verknüpft.
Das folgende Schema (Abb. 2.1) zeigt eine bitweise Stromverschlüsselung.

Abb. 2.1 *Prinzip der Stromchiffre*

Dieses Verfahren funktioniert wie folgt:
p_i sei ein Bit aus dem Klartext, c_i ein Bit aus dem Chiffretext und k_i ein Bit aus dem Schlüsselstrom, dann gilt:

$$p_i \oplus k_i = c_i \rightarrow c_i \oplus k_i = p_i$$

$$p_i \oplus k_i \oplus k_i = p_i$$

Beispiel:

Die Nachricht besteht aus dem Buchstaben „A“, der Schlüssel aus einem Zeichen „*“. Nach dem ASCII-Zeichencode entspricht die Bitfolge für die Nachricht (A) 00100001 und für den Schlüssel (*) sei die Bitfolge 00101010.
Schlüssel und Klartext werden durch die bitweise Verknüpfung mittels exklusivem Oder codiert. 00100001 $\oplus$ 00101010 = 00001011
Dabei wird ein Bit gesetzt, wenn eines der beiden zu verknüpfenden Bits 0 ist und das andere 1 ist. Das Bit wird nicht gesetzt, wenn beide zu verknüpfende Bits 0 oder 1 sind.
Dies wird in der Wahrheitstabelle (siehe Abb. 2.2) dargestellt.

p_i	k_i	c_i
0	0	0
0	1	1
1	0	1
1	1	0

Abb. 2.2 *Wahrheitstabelle*

Bei der Dechiffrierung wird der Chiffretext bitweise mit dem Schlüssel des Empfängers verknüpft (auch wieder XOR).
00001011 $\oplus$ 00101010 = 00100001.

3 Pseudozufallszahlengenerator

Um bei der Stromchiffre möglichst sicher zu sein, sollten Schlüssel nur einmal benutzt werden. In der Praxis wird oft zur Erzeugung der Schlüsselfolge ein *Pseudozufallszahlengenerator* (**P**seudo **R**andom **N**umber **G**enerator) eingesetzt. Der Vorteil ist, dass nur eine kurze geheime Information zur Initialisierung des Generators benötigt wird. Dieser Schlüssel wird vom Sender und Empfänger benutzt und muss geheim bleiben. Das Verfahren ist jedoch nicht mehr absolut sicher.

Ein Pseudozufallszahlengenerator besteht im wesentlichen aus drei Basiselementen (vgl. Abb. 3). Der interne Zustand beschreibt den aktuellen Zustand des Schlüsselstromgenerators. Zwei Pseudozufallszahlengeneratoren mit dem gleichen Schlüssel und dem gleichen internen Zustand liefern den gleichen Schlüsselstrom. Die Ausgabefunktion nimmt den internen Zustand und erzeugt daraus ein Bit des Schlüsselstroms. Die Weiterschaltfunktion ermittelt aus dem aktuellen internen Zustand den nächsten internen Zustand [Schnei 1996, Kapitel 9].

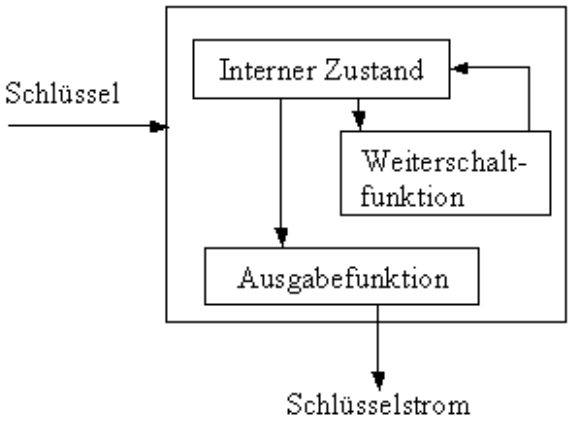

Abb. 3 *Prinzip eines Pseudozufallszahlengenerators*

3.1 Lineare und nichtlineare Pseudozufallszahlengeneratoren

Es gibt viele Möglichkeiten den Pseudozufallszahlengenerator mit Zufallsdaten zu füttern, da es aber nur schwer möglich ist echte Zufälle in Software oder Hardware zu implementieren, muss man sich andere Methoden einfallen lassen.

Eine davon ist die Verwendung von rückgekoppelten Schieberegistern (Linear Feed-Back Shift Register - LFSR). Das folgende Bild zeigt ein LFSR mit $n = 4$ Stellen.

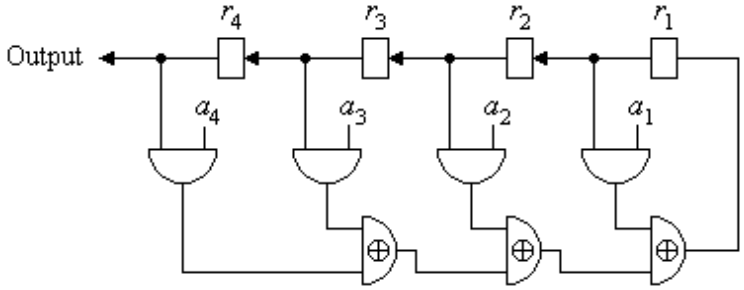

Abb. 3.1 *Prinzip eines LFSR*

Diese sind mit $r_1, r_2, r_3, r_4 \in \{0,1\}$ vorbesetzt. Diese Werte werden mit konstanten Werten $a_1, a_2, a_3, a_4 \in \{0,1\}$ multipliziert (logisches Und) und anschließend modulo 2 addiert (logisches Exklusiv-Oder). Das Ergebnis wird im nächsten Takt an die Stelle von r_1 übernommen, die vorherigen Werte werden alle um eine Stelle nach links weitergeschoben und r_4 wird ausgegeben Ein Linear Feed-Back Shift Register der Länge n kann maximal 2^n-1 verschiedene Zustände annehmen. Die Ausgabefolge eines LFSR der Länge n ist periodisch mit maximaler Periode 2^n-1 [EZ].

Neben linearen Verfahren existieren nicht lineare Verfahren, die Pseudozufallszahlen erzeugen. Nichtlineare Verfahren haben aber den Nachteil das sie schlecht zu erforschen und sie damit schlecht beweisbar sind. Deshalb sind sie nicht unumstritten. [GK1995, S. 22]

Beispiele für nichtlineare Verfahren:

$$s_n := 997 * s_{n-1} - \boldsymbol{INT}(997 * s_{n-1})$$ Hewlett Packard HP-25

oder

$$s_n := \boldsymbol{FRAC}(\boldsymbol{FRAC}(1000 * s_{n-1}) + 3 * s_{n-1})$$

oder schließlich

```
var x, y, z: integer;
function rnd: real;
begin x := 170 * (x mod 178) – 63 * (x div 178);
        if x<0 then x:= x + 30323;
        y := 171 * (y mod 177) – 2 * (y div 177);
        if y<0 then y := y + 30269 ;
        z := 172 * (z mod 176) – 35 * (z div 176) ;
        if z<0 then z := z + 30307 ;
        rnd := frac(x/30323.0 + y/30269.0 + z/30307.0);
end;
```

Die erste Folge hat für die Startzahl 0.5284163 die Periode $5 \cdot 10^5$. Für die zweite Folge ist die Periode $5 \cdot 10^7$ und für die dritte existieren Startwerte, die zu einer Periode von $6 \cdot 10^{12}$ führen können.

3.2 Anforderungen an einen Pseudozufallszahlengenerator

Es gibt einige Anforderungen, die ein Pseudozufallszahlengenerator erfüllen muss, um wirkungsvolle Schlüsselströme zu erzeugen. Es muss gelten, dass die generierten Zahlen keine wiederholenden Muster aufweisen, d.h. der Pseudozufallszahlengenerator sollte einen sehr langen Kreislauf haben. Mit Ausnahme des One-Time-Pad Verfahrens sind alle Pseudozufallszahlengeneratoren periodisch, da der Schlüsselstromgenerator auf einen endlichen Automaten (Computer) implementiert ist.

Die Qualität der erzeugten Zufallszahlen spielt eine sehr große Rolle. Es sollte für einen Außenstehenden unmöglich sein, weitere Zahlen vorauszusagen, sogar mit Kenntnis der bisherigen Zahlen. Des weiteren sollten lange eintönige Schlüsselströme vermieden werden. Erst dann wird eine Verschlüsselung weitgehend sicher.

4 Stromchiffre Varianten

4.1 Synchrone Stromchiffre

Bei der *synchronen Stromchiffre* wird der Schlüsselstrom textunabhängig generiert, d.h. ohne Chiffretextrückkopplung. Abbildung 4.1 zeigt eine synchrone Stromchiffre. Diese Variante verlangt eine perfekte Synchronisation zwischen Sender und Empfänger. Wird die Synchronisation unterbrochen, oder geht bei der Übertragung ein Bit verloren, werden alle nachfolgenden Bits falsch entschlüsselt. Tritt dieser Fall ein, dann muss der Pseudozufallszahlengenerator auf beiden Seiten neu synchronisiert werden. Ein zurücksetzen auf einen früheren Zustand sollte nicht in Betracht gezogen werden, da sonst Teile des Schlüsselstroms wiederholt würden. Aber es gibt auch Vorteile, die synchrone Stromchiffren mit sich bringen. Wird bei der Übertragung ein Bit verfälscht, so hat dies nur Einfluss auf dieses betroffene Bit. Vorherige und nachfolgende Bits sind nicht davon betroffen. [Schnei 1996, Kapitel 9]

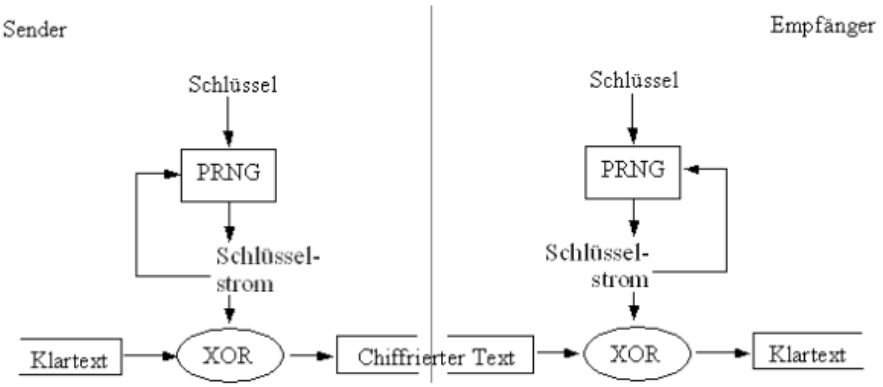

Abb. 4.1 *Synchrone Stromchiffre*

4.2 Selbstsynchronisierende Stromchiffre

Selbstsynchronisierende Stromchiffre haben eine Chiffretextrückkopplung (siehe Abb. 4.2), so dass auftretende Synchronisationsfehler nach einigen Zeichen automatisch behoben sind. Bei raffinierten Implementierungen beginnt jede Nachricht mit einem zufälligen Header. Dieser hat nun die Aufgabe beide Pseudozufallszahlengeneratoren zu synchronisieren. Ein großer Nachteil der selbstsynchronisierenden Stromchiffre ist die Fehlerfortpflanzung. Wurde ein Bit während der Übertragung gestört, so liefert der Pseudozufallszahlengenerator des Empfängers einen falschen Schlüsselstrom. Dies führt nun wiederum dazu, dass alle weiteren Bits falsch entschlüsselt werden, bis sich der Pseudozufallszahlengenerator des Empfängers neu synchronisiert hat. [Schnei 1996, Kapitel 9]

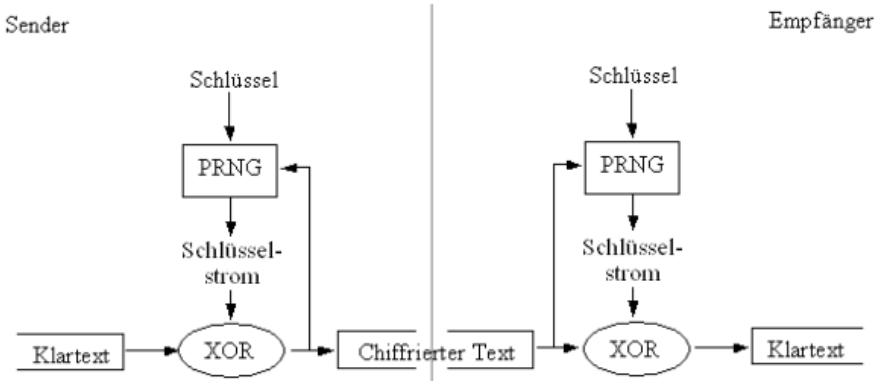

Abb. 4.2 *Selbstsynchronisierende Stromchiffre*

4.3 One-Time-Pad

Das sicherste Verfahren, welches als einziges beweisbar sicher ist, ist das *One-Time-Pad* (auch Vernam-Chiffre) Verfahren. Dabei muss der Schlüssel genauso lang wie die zu verschlüsselnde Nachricht sein und aus einer Folge von echten Zufallszeichen bestehen. Dieser Algorithmus ist absolut sicher. Ohne Ermittlung des Klartextes oder des Schlüssels ist es selbst mit beliebigem technischen Aufwand unmöglich, aus dem Chiffretext auf den Klartext zu schließen. Der One-Time-Pad ist aber ohne große praktische Bedeutung, da der Aufwand (sichere Übertragung des Schlüssels) zu groß ist. Deshalb wird dieser nur bei sehr hohem Geheimhaltungsbedürfnis verwendet.

4.4 Stärken der Stromchiffre

Da es sich bei den Stromchiffren um sehr schnelle Verschlüsselungsverfahren handelt, eignen sie sich gut für die Implementierung in Hardware und zum Verschlüsseln von Echtzeit-Kommunikation und für die Verschlüsselung kontinuierlicher Datenströme, etwa einer Standleitung zwischen zwei Computern. Auch kann jedes Bit sofort nach dem Erhalt dechiffriert werden, es muss also nicht die ganze chiffrierte Nachricht vorliegen. Die Sicherheit dieses Verfahrens hängt sehr stark vom verwendeten Zufallsgenerator ab. Die Sicherheitsstufen reichen vom extrem unsicher bis hin zu extrem sicher (One-Time-Pad).

4.5 Sicherheitsschwächen der Stromchiffre

Die Sicherheit beruht vollständig auf dem Schlüssel und dem Pseudozufallszahlengenerator. Liefert dieser bei jedem Einsatz den gleichen Bitstrom, so lässt sich das System leicht knacken. Folgende zwei Beispiele sollen dies demonstrieren.

Beispiel 1

Besitzt man den Klartext einer Nachricht, dann kann man diesen XOR mit dem zugehörigen Chiffretext verknüpfen und man erhält den Schlüsselstrom.

Beispiel 2

Hat man zwei verschiedene Chiffretexte, die mit dem selben Schlüsselstrom chiffriert wurden, so verknüpft man beide XOR miteinander. Als Resultat erhält man eine XOR Verknüpfung der beiden Klartexte. Einer der Klartexte wird geknackt, was jetzt sehr einfach ist, und dieser wird noch mal mit dem Chiffretext XOR verknüpft. Dieses liefert den Schlüsselstrom, der für alle Nachrichten verwendet werden kann.

Alle weiteren Nachrichten können nun mit dem gewonnenen Schlüsselstrom dechiffriert werden, außerdem alle früheren Nachrichten, die aufgezeichnet wurden.
Kennt ein Angreifer nur Teile des Klartextes, so kann er versuchen diesen mit dem dazugehörigen Chiffretext XOR zu verknüpfen. Man erhält so einen Teil des Schlüsselstroms. Durch Analyse dieser Bits kann man versuchen, das nächste Bit zu erraten. Erreicht der Angreifer dabei eine höhere Trefferwahrscheinlichkeit für das nächste Bit als ½, so kann er den Pseudozufallszahlengenerator wirkungsvoll angreifen [Frie1993, S.398].
Um sich vor solchen Angriffen zu schützen, sollte man niemals zwei verschiedene Nachrichten mit dem selben Schlüssel codieren..

5 Resümee

Die Stromchiffre ist ein sehr einfaches Verfahren zur Chiffrierung von Daten. Ein Schlüsselstrom und ein Klartextstrom werden bitweise XOR verknüpft. Die Kunst besteht nun darin, einen möglichst zufälligen Schlüsselstrom, aus einem kurzen Initialisierungsschlüssel, zu erzeugen. Ein Pseudozufallszahlengenerator liefert einen solchen Schlüsselstrom, in einer Qualität, die entscheidend für die Sicherheit dieses Verfahrens ist.

In der Praxis hat das Verfahren vor allen bei dem Militär und bei den Kommunikationsmedien Anwendung gefunden. Die Stromchiffre eignet sich aber in vielen Fällen nicht für den Einsatz. Andere Verfahren, wie Blockchiffre oder Asymmetrische Verfahren, bieten bessere Möglichkeiten. Zum Beispiel ist die Schlüsselvergabe bei den Asymmetrischen Verfahren kein Problem, da es einen öffentlichen und einen privaten Schlüssel gibt. Blockchiffre eignen sich besser für die Softwareimplementierung, sie müssen nicht rechenintensive Bitoperationen ausführen. Wenn man sich für die Stromchiffre entscheidet, dann kann man sogar Chiffretexte erzeugen, die nachweisbar sicher sind (One-Time-Pad). In vielen Fällen wird ein so hoher Aufwand eher unnötig sein.

6 Literatur

[Schnei1996]	Bruce Schneier: Angewandte Kryptographie, Addison-Wesley, Bonn (1996)
[Frie1993]	O. Fries, A. Fritsch, V. Kessler, B. Klein: Sicherheitsmechanismen: Bausteine zur Entwicklung sicherer Systeme, Oldenburg-Verlag, München (1993)
[GK1995]	Ernst Günter Giessmann, Dirk Verworner: Kryptologie (1995)
[EZ]	Erzeugung von Zufallsbits: http://www.iti.fh-flensburg.de/lang/algorithmen/code/krypto/zufall.htm
Bildquelle Abb 1:	"Entzifferte Geheimnisse - Codes und Chiffren und wie sie gebrochen werden" von Friedrich L. Bauer. Springer-Verlag, 1995