

Kryptographie – ein Überblick für Fortgeschrittene

1. Caesar-Verschlüsselung: Idee, Angriff (bruteforce, Häufigkeit)

Einführung (Wikipedia): <https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung>

Zitat Wikipedia:

Da die Größe des Schlüsselraums nur 25 beträgt, liegt nach Ausprobieren spätestens nach dem 25. Versuch der Klartext vor. Eine erschöpfende Schlüsselsuche (Exhaustion) ist bei der Caesar-Verschlüsselung trivial realisierbar. Da dies auch ohne Computer oder Rechenmaschine mit geringem Aufwand möglich ist, bestand die Sicherheit der Caesar-Verschlüsselung schon zu ihren Anfängen nicht auf der Geheimhaltung des Schlüssels, sondern im Wesentlichen auf der Geheimhaltung des Verfahrens, und entspricht damit nicht dem im 19. Jahrhundert postulierten Prinzip von Auguste Kerckhoffs.

Begriffe:

Klartext: Die von Menschen lesbare Nachricht

Geheimtext: Die verschlüsselte Nachricht (nicht unmittelbar menschenlesbar)

Kryptosystem: Ein Verfahren, welches einen Klartext durch Anwendung eines Schlüssels in einen Geheimtext wandelt und auch wieder zurückwandelt

Schlüssel: (s. Kryptosystem)

Klartextalphabet (Alle möglichen Symbole, die im Klartext vorkommen dürfen)

Geheimtextalphabet (Alle möglichen Symbole, die im Geheimtext vorkommen dürfen)

Schlüsselalphabet (Alle möglichen Symbole, die im Schlüssel vorkommen dürfen)

Beim Kryptosystem Caesar gilt: Sowohl Klartext als auch Geheimtext sind die 26 Buchstaben des lat. Alphabets. Schlüsselalphabet sind alle ganze Zahlen im Intervall [0;25]. Dieser fungiert als Verschiebungssummand.

Beispiel:

Klartext: t i m i s t s p i t z e l

Schlüssel: 2

Geheimtext: V K O K U V U R K V B G N

2. Kerckhoffsches Prinzip (1883)

Verkürzt: "Die Sicherheit eines Verschlüsselungsverfahrens beruht auf der Geheimhaltung des Schlüssels anstatt auf der Geheimhaltung des Verschlüsselungsalgorithmus."

Problematisches Gegenprinzip: „[Security through obscurity](#)“: Sicherheit durch Geheimhaltung des Verschlüsselungsalgorithmus selbst.

Warum ist das Problematisch? U.a. wegen:

* Es ist viel schwieriger, einen Algorithmus geheim zu halten als einen Schlüssel.

* Es ist schwieriger, einen enttarnten Algorithmus durch einen anderen zu ersetzen als einen enttarnten Schlüssel.

Beispiele für „[Security through obscurity](#)“:

• In gewisser Weise ist Caesar „Security through obscurity“, denn wenn man weiß, dass mit Caesar verschlüsselt worden ist, hat man den Klartext mit vertretbarem Aufwand gefunden.

- „Navajo-Code-Talkers“ (Film „Windtalkers“) im 2. WK: Angehörige eines Indianervolkes kommunizieren in ihrer Sprache, die vom Feind nicht verstanden werden kann.

3. Vigenere Verschlüsselung

Einführung (Wikipedia): <https://de.wikipedia.org/wiki/Vigen%C3%A8re-Chiffre>

Idee:

- Statt Verschiebungssummanden (0 – 25 wie bei Caesar) benutzt man die Buchstaben a-z
- Man benutzt nicht nur einen einzigen Schlüsselbuchstaben sondern ein Schlüsselwort und verschlüsselt aufeinanderfolgende Zeichen des Klartextes mit aufeinanderfolgenden Zeichen des Schlüssels.

Angriff durch Häufigkeitsanalyse bei kurzen(!) Schlüsseln und langen(!) Texten.

Wenn Schlüssellänge \geq Klartextlänge und Schlüssel echt zufällig gewählt (nicht aus dem Wörterbuch) gilt, dann ist Vigenere tatsächlich unknackbar (da identisch mit One-Time-Pad)

Beispiel (ohne aufwändiges Vigenere-Quadrat): Klartext: "tim ist spitzel", Schlüssel "bagdad"

Klartext:	t i m i s t s p i t z e l
Schlüssel:	b a g d a d b a g d a d b
(Verschiebung)	1 0 6 3 0 3 1 0 6 3 0 3 1
Geheimtext:	U I S L S W T P O W Z H M

4. Kryptosystem Digi-Vigenere oder: XOR-Verknüpfung

Den Namen „Digi-Vigenere“ habe ich mir nur ausgedacht. Das Prinzip wird aber überall angewandt:

Idee: Alle Alphabete (Klartext-, Schlüssel- und Geheimtextalphabet) bestehen nur aus {0,1}.

Beispiel:

Klartext:	00100110
Schlüssel:	01001100
Geheimtext:	01101010

Digi-Vigenere ist also nichts anderes, als eine XOR-Verknüpfung ("entweder-oder" - aber nicht beides) von Klartext und Schlüssel. Hier ist der Schlüssel genauso lang wie der Klartext, so dass in diesem Fall die Sicherheit eines One-Time-Pads gegeben ist – s.u.)

5. One-Time-Pad

Recherchiere: <https://de.wikipedia.org/wiki/One-Time-Pad>

Das One-Time-Pad ist praktisch „Digi-Vigenere“ mit einem Schlüssel, der mindestens so lang ist, wie die Nachricht selbst.

Ein One-Time-Pad ist theoretisch unknackbar. Dies erkaufte man sich mit folgenden Nachteilen:

Der Einmalschlüssel muss

- mindestens so lang sein wie die Nachricht,
- gleichverteilt zufällig gewählt werden,
- geheim bleiben und
- darf nicht wiederverwendet werden, auch nicht teilweise.

6. Kryptosystem „Vigenere mit Autokey“

Problem und Idee: Das Standard-Vigenere-System verwendet das Schlüsselwort bei langen Klartexten mehrmals hintereinander. Damit ist das Verfahren angreifbar (siehe <https://de.wikipedia.org/wiki/Vigenere-Chiffre#Kryptanalyse>). Es wäre also besser, wenn man den Schlüssel so erweitern könnte, dass nicht immer die gleichen Buchstaben vorkommen. Eine Methode ist „Autokey“: Man benutzt den Klartext selbst als Schlüssel (https://en.wikipedia.org/wiki/Autokey_cipher)!

Beispiel:

Klartext: **treffenumneunamturm**

Schlüssel: **otto**

Die ersten vier Buchstaben des Geheimtexts lauten zunächst: **HKXT**

Wenn der Empfänger die ersten vier Zeichen der Nachricht mit Standard-Vigenere entschlüsselt, so erhält dieser auch die ersten vier Zeichen des Klartexts, nämlich „tref“. Da der Empfänger diese Zeichen nun kennt, können diese also selbst als Schlüssel verwendet werden! Wenn der Absender den Schlüssel „otto“ verwendet, so wird beim Kryptosystem „Vigenere mit Autokey“ faktisch folgendermaßen verschlüsselt:

Klartext: **treffenumneunamturm**

Schlüssel: **ottotreffenumneunam**

Geheimtext: **HKXTYVRZRRROZNQNHRY**

Dieses konkrete System ist gegen klassische Vigenere-Angriffe sicher (aber nach aktuellem Standard längst nicht unknackbar: Siehe entsprechenden Abschnitt in obigem Autokey-Artikel!). Dennoch wird die Idee, praktisch unendlich lange Schlüssel zu verwenden, die irgendwie aus dem Passwort und Teilen des Klartextes erzeugt werden, auch heutzutage in der Praxis verwendet.

7. Stromchiffre (oder „Stream Cipher“) und Zufallsgeneratoren

Die Idee mit „unendlich langen Passwörtern“ lässt sich auch folgendermaßen umsetzen: Man entwirft einen Zufallszahlen-Generator, der sowohl bei Sender als auch Empfänger die gleiche Zufallszahlen-Folge erzeugt, die dann als Schlüssel verwendet wird. Weil dieser Generator auf beiden Seiten zuverlässig die gleichen Zahlen erzeugen muss, ist der Begriff „Zufall“ nicht mehr ganz zutreffend. Man spricht daher von „Pseudo-Zufallszahlen“. Diese kann man z.B. folgendermaßen erzeugen: (Das Beispiel kann man direkt in der Windows-Powershell (mit Suffix `.ps1`) laufen lassen:)

```
# Linearer Kongruenzgenerator (siehe Wikipedia Kongruenzgenerator)
$seed = 124; # (Geheimer) Startwert
$b     = 7;  # b ist Inkrement-Parameter
$a     = 35; # a ist Faktor
$m     = 17; # m ist der Modul, also 0 <= Erg < m
# Für gute Ergebnisse sollte gelten:
# (a-1) sollte durch alle Primfaktoren von m teilbar sein
# (und durch 4, falls m durch 4 teilbar ist)
# b und m sollten teilerfremd sein!
for ($i=0; $i -le 18; $i++) { # 19-mal. (solange i <= 18)
    $seed = ($a * $seed + $b) % $m # eigentliche Formel, % ist mod
    Write-Host (" " + $i + " current seed: " + $seed)
}
pause
```

Die enthaltene Formel, die immer wieder neu mit dem aktualisierten Seed-Wert aufgerufen wird, erzeugt Zahlen zwischen 0 und 16 (also 17 verschiedene Zahlen, vgl. Wert für m). Wenn sich Sender und Empfänger auf einen geheimen Startwert einigen, können im Prinzip beliebig lange Zufallszahlfolgen (man muss m nur genügend groß wählen) erzeugt werden.

Wichtig bei dieser Methode ist, dass man einen Startwert („Seed“) hat, der passend gewählt wurde. Wenn sich Sender und Empfänger auf den gleichen Seed-Wert einigen, können sie mit dem obigen Generator die gleichen (Pseudo-)Zufallszahlen erzeugen und diese Zahlenfolge als One-Time-Pad benutzen.

ACHTUNG: Wie so oft ist diese einfache Idee nicht sicher. Kryptografisch sichere Pseudo-Zufallszahl-Generatoren arbeiten ganz anders!

8. Zwischenfazit zu symmetrischen Verschlüsselungsverfahren

Wikipedia: „Ein symmetrisches Kryptosystem ist ein Kryptosystem, bei welchem im Gegensatz zu einem asymmetrischen Kryptosystem beide Teilnehmer denselben Schlüssel verwenden.“

Bereits Digi-Vigenere erlaubt es uns, beliebige Daten ziemlich sicher zu Verschlüsseln. (Vorsicht: In der Praxis sind Klartexte lang und Schlüssel kurz. In diesem Fall ist Digi-Vigenere sehr unsicher. Es gibt für diese Fälle wesentlich bessere - aber auch kompliziertere Kryptosysteme).

Allgemeines Problem bleibt bislang, dass ein Schlüssel immer im Geheimen ausgetauscht werden muss.

Übungen:

A) Verschlüssele das Wort „informatik“:

1. Mit Cäsar-Verschlüsselung und dem Schlüssel (also Verschiebung) 3
2. Mit Vigenere-Verschlüsselung und dem Schlüssel „bad“
3. Mit Vigenere-Verschlüsselung plus Autokey und dem Schlüssel „bad“

B) Warum müssen die beim One-Time-Pad beschriebenen Voraussetzungen gelten?

C) Bestimme das Ergebnis von $91 \text{ XOR } 205$ (wobei die Zahlen als Binärzahlen zu deuten sind).

D) Ein linearer Kongruenzgenerator erzeugt die Zahlenfolge 2,4,1,3,0,2,4,1,3,0, usw. , also Zahlen zwischen 0 und 4. Es gilt: Parameter a (also der Faktor) ist gleich 1. Gib die beiden anderen Parameter an.

E) Stichwort „[Security through obscurity](#)“: In dem verlinkten Wikipedia-Artikel steht folgendes Beispiel:

In einem Interview [aus dem Jahre 2010] mit dem Magazin CIO gab der Projektleiter des E-Postbriefs, Georg Rau, an: „Grundsätzlich gilt, dass wir hier bei uns keine Sicherheitslücke sehen. Mehr will ich nicht sagen. Denn ein wesentlicher Aspekt unseres Sicherheitskonzeptes ist: Wir reden in der Öffentlichkeit nicht darüber. Das ist Teil des Sicherheitskonzeptes.“

Beurteile diese Haltung aus Security-Perspektive.

9. Diffie-Hellman-Schlüsselaustausch

Der Diffie-Hellman-Schlüsselaustausch ist ein Verfahren, mit dem sich zwei Kommunikationspartner (online, für alle einsehbar) auf einen Schlüssel (z.B. für ein Strom-Chiffre) einigen können, ohne sich im Geheimen treffen zu müssen.

Wir nennen die Kommunikationspartner Alice und Bob. Um mögliche Angriffsszenarien zu beschreiben stellen wir uns vor, dass außerdem ein Angreifer namens Carlo die Kommunikation belauschen – aber nicht verändern – kann.

Grundlegendes Prinzip ist, dass man eine mathematische Operation benutzt, die einfach zu berechnen ist (Potenzieren mit anschließendem Modulo), deren Umkehroperation aber nicht einfach ist.

Einführendes Beispiel (ACHTUNG: NICHT DER ECHE DIFIE-HELLMAN!!!):

Angenommen, aus irgendwelchem Grund ist es auf dem Planeten Melmac ganz einfach, Additionen durchzuführen. Die Umkehroperation (Subtraktion) ist dort aber noch nicht erfunden worden.

Der „Melmac-Schlüsselaustausch“ sieht nun so aus:

- | | |
|---|--|
| A) Alice schlägt eine Zahl g vor und schickt diese offen an Bob. | z.B. $g=100$ |
| B) Alice denkt sich im geheimen für sich die Zahl a aus.
Alice rechnet $x = a + g$. Die Zahl x wird offen an Bob gesandt. | z.B. $a=3$
also $x=103$ |
| C) Bob denkt sich im geheimen für sich die Zahl b aus.
Bob rechnet $y = b + g$. Die Zahl y wird offen an Alice gesandt. | z.B. $b=8$
also $y=108$ |
| D) Alice rechnet $s = y + a$
Bob rechnet $s = x + b$ | also $s = 108 + 3 = 111$
also $s = 103 + 8 = 111$ |

Der Clou ist: s ist für Angreifer Carlo nicht berechenbar, weil er nicht $x - g$ subtrahieren kann (Subtraktion ist auf Melmac ja noch nicht erfunden worden) um a zu errechnen.

Die Kommunikationspartner Alice und Bob kennen nun eine Zahl s (die Carlo nicht kennt) die nun für einen Strom-Chiffre benutzt werden kann.

Auf der Erde ist dieses System natürlich Unsinn, da wir hier sehr einfach Addieren UND Subtrahieren können. Aber es gibt eine andere Rechenoperation, die nicht so einfach umkehrbar ist:

die Potenzierung mit anschließender Modulo-Rechnung.

Beispiel: $5^3 \bmod 11$ ist einfach berechenbar, aber die Lösung der Gleichung $5^x \bmod 11 = 4$ ist schwer!

Der echte Diffie-Hellman-Schlüsselaustausch funktioniert so:

- A) Alice schlägt zwei Zahlen vor und schickt diese an Bob:
Eine (große) Primzahl p und eine ganze Zahl g aus dem Intervall $[1;p[$
Diese beiden Zahlen kann Carlo abhören.
- B) Alice denkt sich im geheimen für sich eine Zahl a aus dem Intervall $[1;p[$ aus.
Alice berechnet $x = g^a \bmod p$
Alice schickt x (für Carlo einsehbar) an Bob
- C) Bob denkt sich im geheimen für sich eine Zahl b aus dem Intervall $[1;p[$ aus.
Bob berechnet $y = g^b \bmod p$
Bob schickt y (für Carlo einsehbar) an Alice
- D) Alice berechnet $s = y^a \bmod p$
Bob berechnet $s = x^b \bmod p$
Die berechnete Zahl s ist bei Bob und Alice identisch, aber für Carlo nicht ohne Weiteres ermittelbar
- E) Die Zahl s kann nun als Schlüssel z.B. für einen Stromchiffre benutzt werden.

Eigenschaften der Modulo-Rechnung:

Beispiel: $(4 \cdot 4) \bmod 3 = ((4 \bmod 3) \cdot (4 \bmod 3)) \bmod 3 = 1$

(hilfreich um Zwischenergebnisse z.B. bei der Programmierung klein zu halten)

Idee beim obigen Schlüsselaustausch:

Alice rechnet: $x = g^a \bmod p$

Bob rechnet: $y = g^b \bmod p$

Alice rechnet: $s = y^a = (g^b)^a \bmod p$

Bob rechnet: $s = x^b = (g^a)^b \bmod p$

Warum ist das Diffie-Hellman-Verfahren „belauschungssicher“?

$2^4 \bmod 5$ zu berechnen ist einfach (hier: $2^4 \bmod 5 = 16 \bmod 5 = 1$)

Aber aus $2^d \bmod 5 = 1$ auf den Exponenten d zu schließen ist schwierig.

Im Allgemeinen muss man probieren:

Wertetabelle:	d	1	2	3	4	
	$2^d \bmod 5$	2	4	3	1	also muss d gleich 4 sein.

Es gibt im Allgemeinen kein Verfahren, um schneller an das d zu gelangen! In der Praxis sind die beteiligten Zahlen natürlich riesig, so dass ein Durchprobieren nicht praktikabel ist.

10. Angriffsmöglichkeiten für den Lauscher Carlo:

Beispielkommunikation:

Alice schlägt vor: $p = 13$ $g = 8$ (diese Zahlen werden offen verschickt)

Alice denkt sich aus: $a = 5$

Alice rechnet: $x = (8^5) \bmod 13 = 8$ (dieses Ergebnis wird offen verschickt)

Bob denkt sich aus: $b = 12$

Bob rechnet: $y = (8^{12}) \bmod 13 = 1$ (dieses Ergebnis wird offen verschickt)

Alice rechnet den Schlüssel aus: $s = y^a \bmod 13 = 1^5 \bmod 13 = 1$

Bob rechnet den Schlüssel aus: $s = x^b \bmod 13 = 8^{12} \bmod 13 = 1$

Carlo kennt die Zahlen: $p = 13$, $g = 8$, $x = 8$, $y = 1$

Carlo versucht nun, die Zahl a zu ermitteln, indem er versucht, die Rechnung von Alice zu rekonstruieren, indem er folgende Gleichung löst:

$$8^a \bmod 13 = 8$$

Carlo MUSS die Gleichung durch probieren lösen (da es keine effizienteren Verfahren gibt):

Wertetabelle:

a:	1	2	3	4	5
$8^a \bmod 13$:	8	12	5	1	8

Weitere Beispielrechnung:

Alice schlägt (offen) vor: $g=2$ und $p=19$

Alice' geheime Zahl: $a=8$

Bobs geheime Zahl: $b=5$

Alice berechnet die Zahl x : $x = (g^a) \bmod p = (2^8) \bmod 19 = 256 \bmod 19 = 9$ *

Bob berechnet die Zahl y : $y = (g^b) \bmod p = (2^5) \bmod 19 = 32 \bmod 19 = 13$

Die Zahlen x und y werden auch offen ausgetauscht.

Alice berechnet geheim: $s = (y^a) \bmod p = (13^8) \bmod 19 = 815730721 \bmod 19 = 16$

Bob berechnet geheim: $s = (x^b) \bmod p = (9^5) \bmod 19 = 59049 \bmod 19 = 16$

Der Schlüssel (den nur Alice und Bob berechnen können) lautet: $s=16$

Diese Zahl $s=16$ kann ein möglicher Angreifer nicht ohne weiteres selbst erzeugen. Denn der Angreifer kennt nur folgende Zahlen:

$g=2$ $p=19$ $x=9$ $y=13$

Um s zu erzeugen benötigt der Angreifer entweder a oder b (beide Zahlen sind aber geheim).

Um z.B. a zu erzeugen, kann der Angreifer durchprobieren (indem er versucht, die Rechnung * zu rekonstruieren):

Probiere $2^? \bmod 19 = 9$

Probiere $2^1 \bmod 19 = 2 \neq 9$ Also: Alice hat sich NICHT $a=1$ ausgedacht!

Probiere $2^2 \bmod 19 = 4 \neq 9$ Also: Alice hat sich NICHT $a=2$ ausgedacht!

Probiere $2^3 \bmod 19 = 8 \neq 9$ Also: Alice hat sich NICHT $a=3$ ausgedacht!

Probiere $2^4 \bmod 19 = 16 \neq 9$ Also: Alice hat sich NICHT $a=4$ ausgedacht!

Probiere $2^5 \bmod 19 = 13 \neq 9$ Also: Alice hat sich NICHT $a=5$ ausgedacht!

Probiere $2^6 \bmod 19 = 7 \neq 9$ Also: Alice hat sich NICHT $a=6$ ausgedacht!

Probiere $2^7 \bmod 19 = 14 \neq 9$ Also: Alice hat sich NICHT $a=7$ ausgedacht!

Probiere $2^8 \bmod 19 = 9 == 9$ Also: Alice hat sich also möglicherweise $a=8$ gedacht!

Damit kann der Angreifer nun (wie Alice selbst) den Schlüssel s berechnen:

$s = (y^a) \bmod p = (13^8) \bmod 19 = 815730721 \bmod 19 = 16$

Beispielrechnung in der Rolle des Angreifers:

Wir versetzen uns in die Rolle des Angreifers. Wir belauschen den Austausch folgender Zahlen:

$g=7$ $p=17$ $x=11$ $y=9$

Finde in der Rolle des Angreifers den Schlüssel s !

Wir ermitteln durch Probieren die geheime Zahl a von Alice:

Probiere $7^? \bmod 17 = 11$

Probiere $7^1 \bmod 17 = 7 \neq 11$ Also: Alice hat sich NICHT $a=1$ ausgedacht!

Probiere $7^2 \bmod 17 = 15 \neq 11$ Also: Alice hat sich NICHT $a=2$ ausgedacht!

Probiere $7^3 \bmod 17 = 3 \neq 11$ Also: Alice hat sich NICHT $a=3$ ausgedacht!

Probiere $7^4 \bmod 17 = 4 \neq 11$ Also: Alice hat sich NICHT $a=4$ ausgedacht!

Probiere $7^5 \bmod 17 = 11 == 11$ Also: Alice hat sich also möglicherweise $a=5$ gedacht!

Als Angreifer können wir nun rechnen:

$s = (y^a) \bmod p = (9^5) \bmod 17 = 8$

Weitere Beispiele (programmtechnisch erzeugt):

Beispiel 1

$g=7$ $p=13$

Alices geheime Zahl: $a=8$

Bobs geheime Zahl: $b=9$

Alice berechnet $x:3$

Bob berechnet $y:8$

Alice berechnet $s:1$

Bob berechnet $s:1$

Angreifer Carlo kennt nur g,p,x und y .

Angreifer Carlo versucht a durch probieren herauszufinden.

(Die Probierkandidaten sind hier nun i)

Probieren $g^i \bmod p = 7^1 \bmod 13 = 7$

Probieren $g^i \bmod p = 7^2 \bmod 13 = 10$

Probieren $g^i \bmod p = 7^3 \bmod 13 = 5$

Probieren $g^i \bmod p = 7^4 \bmod 13 = 9$

Probieren $g^i \bmod p = 7^5 \bmod 13 = 11$

Probieren $g^i \bmod p = 7^6 \bmod 13 = 12$

Probieren $g^i \bmod p = 7^7 \bmod 13 = 6$

Probieren $g^i \bmod p = 7^8 \bmod 13 = 3$

Eine (mögliche) geheime Zahl a von Alice lautet: 8

Carlo rechnet: $s=(y^a) \bmod p = 8^8 \bmod 13 = 1$

Probieren $g^i \bmod p = 7^9 \bmod 13 = 8$

Probieren $g^i \bmod p = 7^{10} \bmod 13 = 4$

Probieren $g^i \bmod p = 7^{11} \bmod 13 = 2$

Probieren $g^i \bmod p = 7^{12} \bmod 13 = 1$