IT Sicherheit: Kryptologie

Dr. Christian Rathgeb

Hochschule Darmstadt, CRISP, da/sec Security Group

22.04.2020

Teilgebiete

Das Wissenschaftsgebiet Kryptologie setzt sich zusammen aus den Teilgebieten:

- Kryptographie: altgriechisch „$\kappa\rho\upsilon\pi\tau o\sigma$“ (geheim) und „$\gamma\rho\alpha\varphi\epsilon\iota\nu$“ (schreiben)
  Ursprünglich also Wissenschaft der Verschlüsselung von Nachrichten
- Kryptoanalyse: altgriechisch „ $\alpha\nu\alpha\lambda\upsilon\sigma\iota\varsigma$ “ (Auflösung)
  Analyse der Verfahren, um diese zu brechen oder Sicherheit zu erhöhen

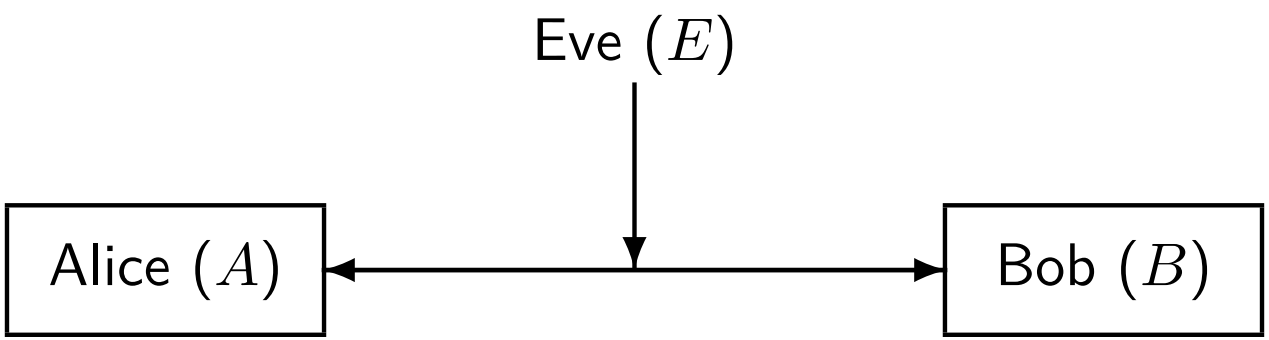
$$\text{Kryptologie} = \text{Kryptographie} + \text{Kryptoanalyse}$$

Geschichte

- Die klassische Kryptologie diente der Geheimhaltung von Nachrichten und wurde hauptsächlich von Militärs, Geheimdiensten und Diplomaten genutzt.
  - Schutziel Vertraulichkeit
- Die moderne Kryptographie (etwa seit 1975) beschäftigt sich mit erheblich weitergehenden Kommunikations- und Sicherheitsproblemen.
  - Schutziele Vertraulichkeit, Integrität, Authentizität, Nichtabstreitbarkeit

Kommunikationsmodell

- Alice (A) und Bob (B) wollen sicher kommunizieren (vgl. Schutzziele)
- Eve (E) versucht, die Schutzziele zu durchbrechen
  - Passiver Angriff: Abhören der Daten
  - Aktiver Angriff: Manipulation (z.B. Verändern, Fälschung Sender) der Daten



Schutzziele

- *Vertraulichkeit:* Nachricht zwischen A und B kann nicht von E gelesen werden
- *Integrität:* Nachricht zwischen A und B wird nicht verändert bzw. A und B können erkennen, ob Nachrichten verändert wurden
- *Datenauthentizität:* B kann Nachricht von A zweifelsfrei A zuordnen
- *Instanzauthentizität:* B kann die Identität von A zweifelsfrei feststellen
- *Nichtabstreitbarkeit:* B kann Nachricht von A zweifelsfrei auch einer dritten Partei als Nachricht von A nachweisen

Kryptographische Verfahren

Beispiele für kryptographische Verfahren für alle Schutzziele:

- *Vertraulichkeit:* Verschlüsselung
- *Integrität:* Hashfunktionen, Message Authentication Codes (MAC), Signaturen
- *Datenauthentizität:* Message Authentication Codes (MAC), Signaturen
- *Nichtabstreitbarkeit:* Signaturen
- *Instanzauthentizität:* Challenge Response Protokolle

Verschlüsselung

Grundsätzlich werden zwei Klassen von Verschlüsselungsverfahren unterschieden:

1. *Symmetrische Verfahren*
2. *Asymmetrische Verfahren* oder *Public Key Verfahren*

Wichtigster Unterschied:

- Symmetrische Verfahren verwenden zum Ver-/Entschlüsseln einen (symmetrischen) Schlüssel während Public Key Verfahren einen Public Key zum Verschlüsseln und einen Private Key zum Entschlüsseln verwenden (Schlüsselpaar).

Was folgt daraus?

Kerckhoffsche Prinzipien

Auguste Kerckhoffs (ein niederländischer Kryptograph) 1883:

- Ist ein System nicht beweisbar sicher, so sollte es praktisch sicher sein.
- Das Design eines System sollte keine Geheimhaltung erfordern und sollte sich ohne Gefahr in den Händen des Feindes befinden können.
- Ein Kryptosystem muss einfach bedienbar sein.

Ein Angreifer kennt das kryptographischen Verfahren, nur die privaten oder symmetrischen Schlüssel sind geheim.

No security by obscurity!

Symmetrische Verfahren

- *Symmetrische Verfahren:* A und B nutzen den selben Schlüssel
  - Schlüssel muss zw. A und B sicher ausgetauscht werden
    - vertraulich: E darf den Schlüssel nicht kennen
    - authentisch: A und B müssen wissen, wem sie vertrauliche Nachrichten schicken
  - Nachteile der symmetrischen Kryptographie: Jeder der verschlüsseln kann, kann auch entschlüsseln und beide Partner müssen einen gemeinsamen geheimen Schlüssel tauschen.
  - Vorteil der symmetrischen Kryptographie: Sehr effizient bzgl. Laufzeit.

Symmetrische Verschlüsselungsverfahren

Schutzziel: Vertraulichkeit

- Verschlüsselungsfunktion enc (encryption):
  - Klartext m, Schlüssel k, Ciphertext $c = \text{enc}(k, m)$
- Entschlüsselungsfunktion dec (decryption):
  - Ciphertext c, Schlüssel k, Klartext $m = \text{dec}(k, c)$
- Zusammenhang: Für alle Klartexte m und Schlüssel k: $\text{dec}(k, \text{enc}(k, m)) = m$

$\boxed{A}$ Schlüssel: k	$\boxed{B}$ Schlüssel: k
compute $c := \text{enc}(k, m)$	
$\xrightarrow{c}$	
	compute $m := \text{dec}(k, c)$

Beispiel: CAESAR-Verfahren

Eingesetzt von Gaius Iulius Caesar:
(römischer Kaiser 100 - 44 v. Chr.)

- Verschlüsselung: Verschiebe alle Buchstaben um einen festen Wert k nach rechts
  Für $k = 3$: A $\rightarrow$ D, B $\rightarrow$ E, C $\rightarrow$ F, ..., Z $\rightarrow$ C
- Entschlüsselung: Verschiebe alle Buchstaben um den selben Wert nach links
  Für $k = 3$: A $\rightarrow$ X, B $\rightarrow$ Y, C $\rightarrow$ Z, ..., Z $\rightarrow$ W
- Dann gilt:
  - $\text{enc}(3, \text{CAESAR}) = \text{FDHVDU}$
  - $\text{dec}(3, \text{FDHVDU}) = \text{CAESAR}$

Sicherheit CAESAR-Verfahren I

Um den Klartext zu erhalten, probieren wir alle Schlüssel aus:

Ciphertext:	`OLYY KLY YPUNL`
Verschiebung um 1 nach links:	`NKXX JKX XOTMK`
Verschiebung um 2 nach links:	`MJWW IJW WNSLJ`
Verschiebung um 3 nach links:	`LIVV HIV VMRKI`
Verschiebung um 4 nach links:	`KHUU GHU ULQJH`
Verschiebung um 5 nach links:	`JGTT FGT TKPIG`
Verschiebung um 6 nach links:	`IFSS EFS SJOHF`
Verschiebung um 7 nach links:	`HERR DER RINGE`

Sicherheit CAESAR-Verfahren II

- Es gibt 26 Buchstaben im deutschen Alphabet
  - Verschiebung um 27 ist das selbe wie Verschiebung um 1, ...
- Es gibt also nur 26 verschiedene Schlüssel
- Durchsuchen des Schlüsselraums möglich $\rightarrow$ *Brute Force*
- Für große Nachrichten ergibt sich in der Regel nur ein sinnvoller Text

Sicherheit CAESAR-Verfahren II

- Es gibt 26 Buchstaben im deutschen Alphabet
  - Verschiebung um 27 ist das selbe wie Verschiebung um 1, ...
- Es gibt also nur 26 verschiedene Schlüssel
- Durchsuchen des Schlüsselraums möglich $\rightarrow$ *Brute Force*
- Für große Nachrichten ergibt sich in der Regel nur ein sinnvoller Text

Erkenntnis

Für die Sicherheit eines Verschlüsselungsverfahrens muss der Schlüsselraum (Menge der möglichen Schlüssel) so groß sein, dass nicht alle Schlüssel durchprobiert werden können.

Modifiziertes CAESAR-Verfahren

- Ersetze Buchstaben nicht mittels fester Verschiebung, sondern beliebig (Monoalphabetische Verschlüsselung)

A	B	C	D	···	X	Y	Z
↓	↓	↓	↓	···	↓	↓	↓
X	E	S	U	···	M	H	P

- Wie viele Schlüssel gibt es?
  - A kann durch 26 Buchstaben ersetzt werden
  - B kann durch 25 Buchstaben ersetzt werden

Modifiziertes CAESAR-Verfahren

- Ersetze Buchstaben nicht mittels fester Verschiebung, sondern beliebig (Monoalphabetische Verschlüsselung)

A	B	C	D	⋯	X	Y	Z
↓	↓	↓	↓	⋯	↓	↓	↓
X	E	S	U	⋯	M	H	P

- Wie viele Schlüssel gibt es?
  - A kann durch 26 Buchstaben ersetzt werden
  - B kann durch 25 Buchstaben ersetzt werden
  - Insgesamt: $26 \cdot 25 \cdots 2 \cdot 1 = 26!$ verschiedene Schlüssel
- Frage: sind $26!$ Schlüssel zu viel, um alle durchzuprobieren?

Große Zahlen

Anzahl Schlüssel: $26! \approx 2^{88} \approx 10^{27}$

- Anzahl der Atome der Erde 2^{170}
- Anzahl der Atome der Sonne 2^{190}
- Anzahl der Atome in unserer Galaxis 2^{223}

Angenommen, ein Computer berechnet pro Sekunde $2 \cdot 10^9$ Entschlüsselungen:

$$\frac{\text{Anzahl möglicher Schlüssel}}{\text{Entschl. je Sek. * Sek. pro Jahr}} = \frac{10^{27}}{(2 * 10^9 s^{-1}) * (3.15 * 10^7 s/\text{Jahr})}$$

$$= \frac{10^{11}}{6.3} > 10^{10} = 10 \text{ Mrd. Jahre}$$

Sicherheit des modifizierten CAESAR-Verfahrens

- Angriff über relative Häufigkeiten in deutschen Texten

  1. Buchstabe E: 17.4 %
  2. Buchstabe N: 09,8 %
  3. Buchstabe I: 07.6 %
  4. Buchstabe S: 07,3 %
  5. Buchstabe R: 07,0 %
  6. Buchstabe A: 06,5 %
  7. Buchstabe T: 06,2 %
  8. Buchstabe D: 05,1 %

  ⋮

  26. Buchstabe Q: 00,02 %

- Richtige Wahl von E und N liefert schon ca. $1/4$ des Textes. Rest erhält man durch geschicktes Raten.

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of

sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt

yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts

sof Kdafopdtp xt Meyyxtpot uoxt Otso.
```

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e        e  e        e  e        e    e       e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
 e   e         e e   e  e  e    e       e  e         e
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e   e     e    e   e   e e    e       e  e e e
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
 e     e             e   e  E  e
```

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E							

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e          e  e  n   n  e  e  n   n  en   e       e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
 e n e          e e   e ne  e n n e      en  e          e n
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e  n e     e    e   e   e en    e      e   e e e   n
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
 e     e  n   n     n en  e n En e
```

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N						

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e          e  e  n  n  e  e  n  n  en   e       e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
 e n e          e e  e ne  e n n e      en  e         e n
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e  n e     e    e   e   e en    e       e   e e e   n
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
 e     e  n   n     n en  e n En e
```

Letztes Wort: Otso = Ende oder Ente

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N						

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e          e  e  n  n  e  e  nd  n  end   e  d    e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e          e e   e ne  e n nde      en  e          e n
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e  nde     e    e   e   e en    e      de   e ede   nd
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
de     e  n   n     n en  e n Ende
```

Letztes Wort: Otso = Ende oder Ente

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N				D		

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e          e  e  n   n  e  e   nd  n  end   e  d    e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e         e e   e ne  e n nde      en  e          e n
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e  nde     e    e   e   e en    e      de  e ede   nd
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
de      e  n   n     n en  e n Ende
```

Kurze Wörter: xt, oxt = in, ein

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N				D		

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e    i     e  e in   n  e  e   nd  n  endi  e  d    e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e          eie   eine  ein nde   i   en  e          ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e  nde      e   i e   e    e en     e      de   e ede   nd
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
de      e  n  in     in en  ein Ende
```

Kurze Wörter: xt, oxt = in, ein

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I			D		

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
    e    i     e  e in   n  e  e   nd  n  endi  e  d    e
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e          eie   eine  ein nde   i   en  e          ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 e  nde      e   i e   e    e en     e      de   e ede   nd
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
de      e  n  in     in en  ein Ende
```

Kurze Wörter: sof,soh = der,des

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I			D		

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
  s  err  i     e  e in  n  e  e s nd  n  endi  e  d ss er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e  s    r  eier seines ein nde  i s en  e  r s   s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  r e   i e   es   e en    e    r des  eredes  nd
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der    re  n  in     in en  ein Ende
```

Kurze Wörter: sof,soh = der,des

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R		D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
  s  err  i     e  e in  n  e  e s nd  n  endi  e  d ss er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e  s    r  eier seines ein nde  i s en  e  r s   s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  r e   i e   es   e en    e    r des  eredes  nd
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der    re  n  in     in en  ein Ende
```

dts = und

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R		D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
  s  err  i     eu e in   n  eu e s nd  n uendi  e  d ss er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e  s   ur  eier seines einunde  i s en  e ur s   s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  r e   i es  es   e en     e    r des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der  u re un  in     in en  ein Ende
```

dts = und

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
  s  err  i     eu e in   n  eu e s nd  n uendi  e  d ss er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de n e  s   ur  eier seines einunde  i s en  e ur s   s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  r e   i es  es   e en    e    r des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der  u re un  in     in en  ein Ende
```

skhh = dass

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
A s  err  i     eu e in  n  eu e sand an uendi  e  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae  s   ur  eier seines einunde  i s en  e ur s a s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  rae  i es  es  e en    e  ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in    in en  ein Ende
```

skhh = dass

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
A s  err  i     eu e in   n  eu e sand an uendi  e  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae  s   ur  eier seines einunde  i s en  e ur s a s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  rae   i es  es   e en     e   ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in     in en  ein Ende
```

Kvh = Als

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
Als  err  il    eu elin   n  eu elsand an uendi  e  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae  s   ur  eier seines einundel  i s en  e ur s a s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  rae   i es  es   e en   lle   ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in     in en  ein Ende
```

Kvh = Als

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
Als  err  il    eu elin   n  eu elsand an uendi  e  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae  s   ur  eier seines einundel  i s en  e ur s a s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  rae   i es  es   e en   lle   ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in     in en  ein Ende
```

G = T?

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
Als  err  il   eutelin  n  eutelsand an uendi te  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae  st  ur  eier seines einundel  i sten  e urtsta s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  rae  ti es  est  e en  lle  ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in    in en  ein Ende
```

G = T?

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	T

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
Als  err  il   eutelin  n  eutelsand an uendi te  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae  st  ur  eier seines einundel  i sten  e urtsta s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
 es nders  rae  ti es  est  e en  lle  ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in    in en  ein Ende
```

Moff = Herr, yohetsofh = besonders

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	T

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
Als Herr Bilbo Beutelin  on Beutelsand an uendi te  dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
de nae hst  ur  eier seines einundel  i sten  eburtsta s ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
besonders  rae hti es  est  eben  olle  ar des  eredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Au re un  in Hobbin en  ein Ende
```

Moff = Herr, yohetsofh = besonders

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	T

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens: Beispiel

Geheimtext:

```
Kvh Moff Yxvye Yodgovxt zet Yodgovhkts ktudotsxpgo, skhh of
Als Herr Bilbo Beutelin von Beutelsand ankuendigte, dass er
sortkonmhg ldf Aoxof hoxtoh oxtdtsovalxphgot Poydfghgkph oxt
demnaechst zur Feier seines einundelfzigsten Geburtstags ein
yohetsofh wfkonmgxpoh Aohg poyot cevvo, ckf soh Pofosoh dts
besonders praechtiges Fest geben wolle, war des Geredes und
sof Kdafopdtp xt Meyyxtpot uoxt Otso.
der Aufregung in Hobbingen kein Ende.
```

Häufigkeiten im Text (Plätze 1 - 8):

O	T	X	F	D	S	H	G
36	18	12	12	11	11	10	9
E	N	I	R	U	D	S	T

Relative Häufigkeiten (Plätze 1 - 8):

E	N	I	S	R	A	T	D
17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1

Sicherheit des modifizierten CAESAR-Verfahrens

Solche Angriffe heißen auch statistische Angriffe

Erkenntnis

- Zur Beurteilung der Sicherheit müssen alle kryptoanalytischen Methoden berücksichtigt werden
- Größe des Schlüsselraums ist nur eine notwendige (keine hinreichende) Bedingung für die Sicherheit.

Stromchiffren und Blockchiffren

- Symmetrische Verschlüsselungsverfahren werden grundlegend in zwei Klassen eingeteilt: *Stromchiffren* und *Blockchiffren*
- Eine Stromchiffre verschlüsselt den Klartext i.A. bitweise (manchmal aber auch in größeren Einheiten). Dieses geschieht in den meisten Fällen durch eine bitweise exklusive Veroderung (XOR) mit einem Schlüsselstrom
- Bei Blockchiffren findet (im Unterschied zu den Stromchiffren) die Verschlüsselung nicht bitweise sondern in Blöcken fester Größe statt (üblich sind im Allgemeinen 64 Bit)

Exklusive Veroderung (XOR)

- XOR ist eine logische Verknüpfung mehrerer Eingänge die genau dann "1" (wahr) wird wenn wenn an einer ungeraden Anzahl von Eingängen "1" ist und die restlichen "0" (falsch).
- Bei zwei Eingängen, A und B, müssen diese verschieden sein damit der Ausgang "1" wird:

A	B	A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

One-time Pad I

Wir betrachten Nachrichten als Bitstrings:

- Eine Nachricht $m = (m_1, \ldots, m_n)$ besteht nur aus 0 und 1
- Für eine Nachricht der Länge $n \in \mathbb{N}$ schreiben wir auch kurz: $m \in \{0,1\}^n$
- Beispiel: Kodierung der Buchstaben über ASCII-Kode $A = 01000001$, $B = 01000010$, $\ldots$

Als Schlüssel nutzen wir ebenfalls Bitstrings:

- Für eine Nachricht der Länge $n \in \mathbb{N}$ benötigen wir
- Schlüssel $k = (k_1, \ldots, k_n) \in \{0,1\}^n$ der Länge n.

One-time Pad II

- Verschlüsselung: Ciphertext $c = m \oplus k$.

  $c = (c_1, \ldots, c_n) = (m_1, \ldots, m_n) \oplus (k_1, \ldots, k_n) = (m_1 \oplus k_1, \ldots, m_n \oplus k_n)$

- Entschlüsselung: Klartext $m = c \oplus k$.

  $m = (m_1, \ldots, m_n) = (c_1, \ldots, c_n) \oplus (k_1, \ldots, k_n) = (c_1 \oplus k_1, \ldots, c_n \oplus k_n)$

One-time Pad II

- Verschlüsselung: Ciphertext $c = m \oplus k$.

  $c = (c_1, \ldots, c_n) = (m_1, \ldots, m_n) \oplus (k_1, \ldots, k_n) = (m_1 \oplus k_1, \ldots, m_n \oplus k_n)$

- Entschlüsselung: Klartext $m = c \oplus k$.

  $m = (m_1, \ldots, m_n) = (c_1, \ldots, c_n) \oplus (k_1, \ldots, k_n) = (c_1 \oplus k_1, \ldots, c_n \oplus k_n)$

- Korrektheit: Wegen $0 \oplus 0 = 0$ und $1 \oplus 1 = 0$ gilt

  $k \oplus k = (k_1, \ldots, k_n) \oplus (k_1, \ldots, k_n) = (k_1 \oplus k_1, \ldots, k_n \oplus k_n) = (0, \ldots, 0)$

One-time Pad II

- Verschlüsselung: Ciphertext $c = m \oplus k$.

  $c = (c_1, \ldots, c_n) = (m_1, \ldots, m_n) \oplus (k_1, \ldots, k_n) = (m_1 \oplus k_1, \ldots, m_n \oplus k_n)$

- Entschlüsselung: Klartext $m = c \oplus k$.

  $m = (m_1, \ldots, m_n) = (c_1, \ldots, c_n) \oplus (k_1, \ldots, k_n) = (c_1 \oplus k_1, \ldots, c_n \oplus k_n)$

- Korrektheit: Wegen $0 \oplus 0 = 0$ und $1 \oplus 1 = 0$ gilt

  $k \oplus k = (k_1, \ldots, k_n) \oplus (k_1, \ldots, k_n) = (k_1 \oplus k_1, \ldots, k_n \oplus k_n) = (0, \ldots, 0)$

  - Wir erhalten

    $$c \oplus k \quad = \quad (m \oplus k) \oplus k = m \oplus (k \oplus k) = m \oplus 0 = m$$

One-time Pad III

Klartext: ASCII:	G 01000111	E 01000101	H 01001000	E 01000101	I 01001001	M 01001101
			$\oplus$			
Schlüssel:	00111000	10011110	10011111	00010111	10100111	10011110
			=			
Ciphertext:	01111111	11011011	11010111	01010010	11101110	11010011
			$\oplus$			
Schlüssel:	00111000	10011110	10011111	00010111	10100111	10011110
			=			
ASCII: Klartext:	01000111 G	01000101 E	01001000 H	01000101 E	01001001 I	01001101 M

Einschub: Wahrscheinlichkeitstheorie I

Wir betrachten folgendes Experiment: Münzwurf

- Bei einmaligem Münzwurf gibt es nur zwei Ereignisse: Kopf oder Zahl
  - Die Wahrscheinlichkeit, Kopf bzw. Zahl zu werfen, ist $\frac{1}{2}$, kurz $\Pr(\text{Kopf}) = \frac{1}{2}$, $\Pr(\text{Zahl}) = \frac{1}{2}$

Einschub: Wahrscheinlichkeitstheorie I

Wir betrachten folgendes Experiment: Münzwurf

- Bei einmaligem Münzwurf gibt es nur zwei Ereignisse: Kopf oder Zahl
  - Die Wahrscheinlichkeit, Kopf bzw. Zahl zu werfen, ist $\frac{1}{2}$, kurz $\Pr(\text{Kopf}) = \frac{1}{2}$, $\Pr(\text{Zahl}) = \frac{1}{2}$

Identifiziere Kopf mit 0 und Zahl mit 1.

Einschub: Wahrscheinlichkeitstheorie I

Wir betrachten folgendes Experiment: Münzwurf

- Bei einmaligem Münzwurf gibt es nur zwei Ereignisse: Kopf oder Zahl
  - Die Wahrscheinlichkeit, Kopf bzw. Zahl zu werfen, ist $\frac{1}{2}$, kurz $\Pr(\text{Kopf}) = \frac{1}{2}$, $\Pr(\text{Zahl}) = \frac{1}{2}$

Identifiziere Kopf mit 0 und Zahl mit 1.

- Bei zweimaligem Münzwurf gibt es vier Ereignisse: 00, 01, 10 und 11
  - $\Pr(00) = \Pr(01) = \Pr(10) = \Pr(11) = \frac{1}{4}$

Einschub: Wahrscheinlichkeitstheorie I

Wir betrachten folgendes Experiment: Münzwurf

- Bei einmaligem Münzwurf gibt es nur zwei Ereignisse: Kopf oder Zahl
  - Die Wahrscheinlichkeit, Kopf bzw. Zahl zu werfen, ist $\frac{1}{2}$, kurz $\Pr(\text{Kopf}) = \frac{1}{2}$, $\Pr(\text{Zahl}) = \frac{1}{2}$

Identifiziere Kopf mit 0 und Zahl mit 1.

- Bei zweimaligem Münzwurf gibt es vier Ereignisse: 00, 01, 10 und 11
  - $\Pr(00) = \Pr(01) = \Pr(10) = \Pr(11) = \frac{1}{4}$
- Bei $n-$maligem Münzwurf gibt es 2^n Ereignisse
  - Die Wahrscheinlichkeit, das richtige Ereignis zu raten: $\frac{1}{2^n}$

Sicherheit One-time Pad I

Für Nachricht $m \in \{0,1\}^n$ benötigen wir Schlüssel $k \in \{0,1\}^n$

- Wir nennen k zufällig, wenn $\Pr(k) = \frac{1}{2^n}$ (Wahrscheinlichkeit, dass Angreifer den richtigen Schlüssel errät, ist $\frac{1}{2^n}$)
- Die zufällige Wahl der eingesetzten Schlüssel ist wesentlich für die Sicherheit:
  - Sind alle Schlüssel gleichwahrscheinlich, hat der Angreifer keinen Anhaltspunkt, welcher Schlüssel eingesetzt wurde

Sicherheit One-time Pad II

- Welche Angriffe gibt es gegen das One-time Pad?
  (Unter der Voraussetzung, dass der Schlüssel zufällig gewählt wurde)
- In unserem Beispiel:
  - Aus Klartext $m =$ GEHEIM (48 Bit in ASCII) wurde mittels Schlüssel $k = (k_1, \ldots, k_{48})$ ein Ciphertext $c = (c_1, \ldots, c_{48})$
  - Durchprobieren aller Schlüssel auf den Ciphertext führt zu 2^{48} Klartexten
    - Nicht sinnvolle Texte: üÄY:;- AAAAAA **LAls
    - Sinnvolle Texte: Berlin BOSTON Er ist
  - Ein Angreifer hat keine Information, welcher der sinnvollen Texte der richtige ist

Sicherheit One-time Pad III

$$\begin{array}{rcll} \text{Klartext:} & 001001110 & \longleftarrow & \text{Kein Zufall} \\ & \oplus & & \\ \text{Schlüssel:} & 100011100 & \longleftarrow & \text{Zufall} \\ & = & & \\ \text{Ciphertext:} & 101010010 & \longleftarrow & \text{Zufall} \end{array}$$

Aus Sicht des Angreifers ist der Ciphertext eine absolut zufällige Bitfolge

- Genauso zufällig wie der eingesetzten Schlüssel
- Unabhängig vom verschlüsselten Klartext

Erkenntnis

Beim One-time Pad Verfahren liefert der Ciphertext keinerlei Informationen über den Klartext (und damit auch nicht über den eingesetzte Schlüssel).

Absolute Sicherheit I

Das One-time Pad ist also sicher im Sinne der folgenden Definition.

Definition

Ein Verschlüsselungsverfahren heißt absolut sicher, wenn es auch gegen Angreifer mit unbeschränkten Ressourcen (Zeit, Rechenleistung) sicher ist.

Absolute Sicherheit II

Absolut sichere Verschlüsselungsverfahren haben einen entscheidenden Nachteil:

Claude Shannon, 1948

Ein Verschlüsselungsverfahren kann nur dann absolut sicher sein, wenn der Schlüssel genauso lang ist wie die zu verschlüsselnde Nachricht.

Zum Teil müssen aber große Datenmengen verschlüsselt werden:

- Bilder und Videos von Spionagesatelliten
- Sprach- und Videotelephonie über das Internet

Für die Verschlüsselung von 10 GB benötigen wir einen Schlüssel der Größe 10 GB.

Modifiziertes One-time Pad I

Idee: Verwende den Schlüssel mehrfach

- Wir wollen zwei Nachrichten $m^1 = (m_1^1, \ldots, m_n^1)$ und $m^2 = (m_1^2, \ldots, m_n^2)$ mit **einem** Schlüssel $k = (k_1, \ldots, k_n)$ verschlüsseln.
- Für die Ciphertexte $c^1 = m^1 \oplus k$ und $c^2 = m^2 \oplus k$ gilt dann

$$c^1 \oplus c^2 = (m^1 \oplus k) \oplus (m^2 \oplus k) = m^1 \oplus m^2 \oplus \underbrace{k \oplus k}_{=(0,\ldots,0)} = m^1 \oplus m^2$$

Wir kennen also die Summe der Klartexte, ohne den Schlüssel kennen zu müssen!

Modifiziertes One-time Pad II

Statistischer Angriff

- Bestimme alle sinnvollen Texte für m^1
- Überprüfe, ob die Wahl von m^1 auch einen sinnvollen Text für m^2 ergibt:
  Wegen $c^1 \oplus c^2 = m^1 \oplus m^2$ gilt

$$m^2 = c^1 \oplus c^2 \oplus m^1$$

  - Wenn ja: haben wir die Nachrichten m^1, m^2 gefunden
  - Wenn nein: wählen wir einen anderen Kandidaten für m^1
- Wahrscheinlichkeit, dass es hier mehrere Möglichkeiten gibt, ist klein
- Wahrscheinlichkeit sinkt mit Anzahl der Nachrichten

Modifiziertes One-time Pad III

Erkenntnis

Selbst auf den ersten Blick kleine Veränderungen eines kryptographsichen Verfahrens können zu einer massiven Senkung der Sicherheit führen.

Absolute und praktische Sicherheit

- Absolute Sicherheit: Resistent gegen Angreifer mit unbeschränkten Ressourcen
- Praktische Sicherheit: Resistent gegen Angreifer mit beschränkten Ressourcen:
  - Angreifer haben nur beschränkte Rechenkapazität und Zeit
  - Kryptoverfahren sind nur sicher in Bezug auf diese Ressourcen

Definition Sicherheitsniveau

Ein Kryptoverfahren hat ein Sicherheitsniveau von $n \in \mathbb{N}$ Bit, wenn ein Angreifer 2^n Versuche benötigt, dass Verfahren zu brechen. (Für Verschlüsselungsverfahren: den Klartext zu erhalten.)

Sicherheitsniveau I

- Für die Beurteilung des Sicherheitsniveaus müssen alle kryptoanalytischen Verfahren herangezogen werden:
  - Brute Force (Durchprobieren aller Schlüssel)
  - statistische Angriffe (wie beim modifizierten CAESAR-Verfahren)
  - ...

Sicherheitsniveau II

- Sicherheitsniveaus der bisherigen Verfahren:
  - CAESAR-Verfahren: $26 < 2^5$ Schlüssel, also Sicherheitsniveau < 5 Bit
  - modifiziertes CAESAR-Verfahren (monoalphabetische Verschlüsselung)
    - $26! \approx 2^{88}$ Schlüssel, also Sicherheitsniveau nicht größer als 88 Bit
    - statistischer Angriff: Schwer zu quantifizieren, ca. $2^6 > 50$ Versuche
    - Sicherheitsniveau: < 6 Bit
  - One-time Pad: Sicherheitsniveau ∞ Bit, abhängig von Schlüssellänge

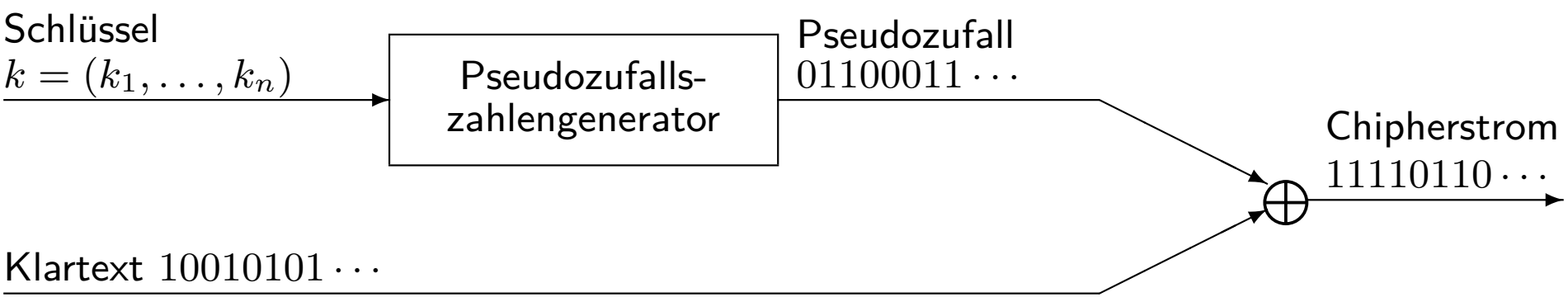
Sicherheitsniveau III

- Heutige Kryptoverfahren sollten ein Sicherheitsniveau ≥ 100 Bit haben
  - 2^{100} Versuche sind praktisch nicht durchführbar
  - Solche Verfahren gelten damit als **praktisch sicher**
- Erinnerung: Brute Force ist ein Angriff
  - Sicherheitsniveau ≥ 100 Bit bedeutet also mindestens 2^{100} Schlüssel
  - Für einen Schlüsselraum der Form $\{0,1\}^n$ also $n \geq 100$

Stromchiffren I

Nachbildung des One-time Pads

- Aus Schlüssel $k \in \{0,1\}^n$, $n \geq 100$ wird ein pseudozufälliger Schlüssel generiert (Schlüsselstrom)
- Der Schlüsselstrom wird komponentenweise mit dem Klartext addiert (XOR)



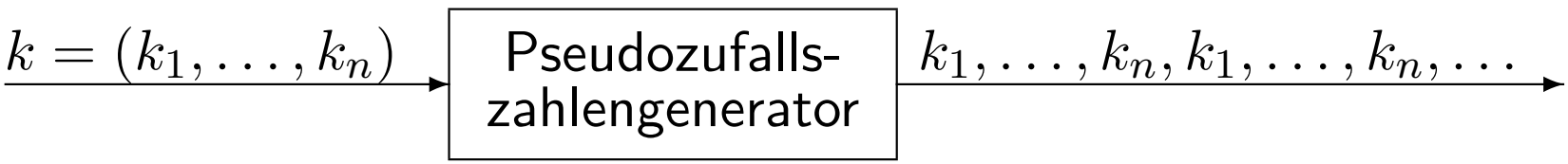
Stromchiffren II

- Zufallszahlen spielen in der IT-Sicherheit eine Wichtige Rolle da mit diesem u.a. Startwerte sog. “Seeds” für die erstellung von Kryptografischen Schlüsseln genutzt werden.
- Im Falle von PRNGs (Pseudo Random Number Generators) werden durch einen deterministischen Algorithmus bei gleichem Startwert (“seed”) immer den gleiche Output liefert.
- Die Entropie ist einfach gesagt die maximale Anzahl an verschiedenen Zuständen, welcher der RNG annehmen kann, bevor er sich wiederholt.
- Ist die Entropie in einem System zu gering, wäre es einem Angreifer möglich, die Schlüssel durch das “raten” von “seeds” selbst zu erstellen/durchzuprobieren (Brute Force Angriff)

Stromchiffren III

Welche Bedingungen muss der Pseudozufallszahlengenerator erfüllen?

- Erste Idee: Nutze Schlüssel k mehrmals



- Unsicher, siehe modifiziertes One-time Pad

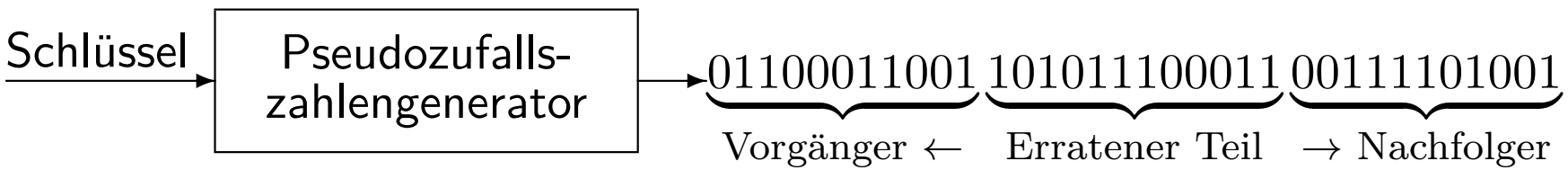
Stromchiffren IV

Welche Bedingungen muss der Pseudozufallszahlengenerator erfüllen?

- Aus 100 Bit Zufall (Schlüssel) lässt sich nicht z.B. 200 Bit Zufall (Schlüsselstrom) berechnen
  - 200 Bit Zufall hieße: Angreifer rät Schlüsselstrom mit Wahrscheinlichkeit $1/2^{200}$
  - Er muss aber nur Schlüssel raten (Wahrscheinlichkeit $1/2^{100}$), und dann den Schlüsselstrom berechnen
- Wir benötigen nur praktische Sicherheit
  - Angreifer (beschränkte Ressourcen) sollte keinen Unterschied feststellen zwischen
    - echtem Zufall
    - Pseudozufall, der von einem Pseudozufallszahlengenerator stammt

Stromchiffren V

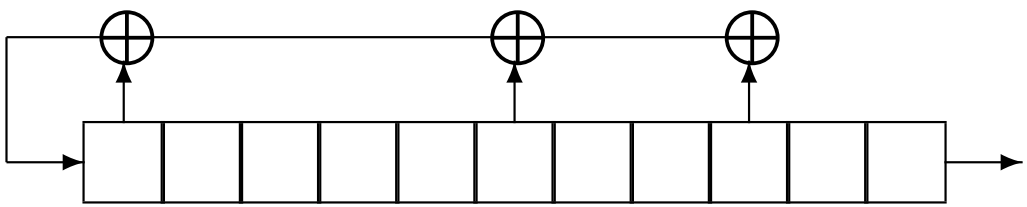
- Formale Definition von Pseudozufallszahlengenerator in den Vorlesungen
  - Kryptologie
  - Komplexitätstheorie
- Wichtig Bedingung für die Sicherheit von Stromchiffren:
  - Aus Teilen des Schlüsselstroms dürfen keine Nachfolger bestimmt werden können
  - Aus Teilen des Schlüsselstroms dürfen keine Vorgänger bestimmt werden können



- Ansonsten Angriffe wie beim modifizierten One-time Pad möglich

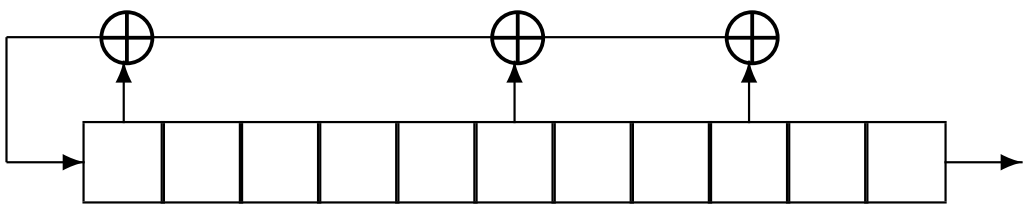
Stromchiffren: LFSR

Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



Stromchiffren: LFSR

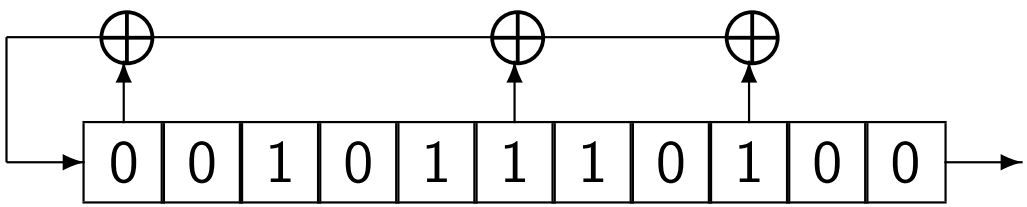
Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



- Im ersten Schritt wird das Register mit dem Schlüssel initialisiert

Stromchiffren: LFSR

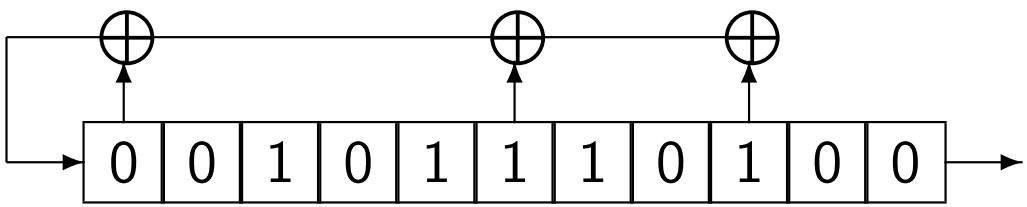
Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



- Im ersten Schritt wird das Register mit dem Schlüssel initialisiert

Stromchiffren: LFSR

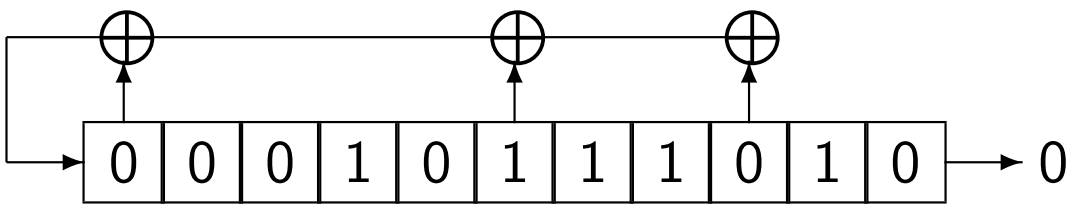
Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



- Im ersten Schritt wird das Register mit dem Schlüssel initialisiert
- In jedem Schritt (Takt), wird
  - ein Bit ausgegeben (jenes aus dem letzten Feld)
  - alles um eine Stelle nach rechts verschoben und
  - die ausgezeichneten Bits addiert und an die erste Stelle geschrieben

Stromchiffren: LFSR

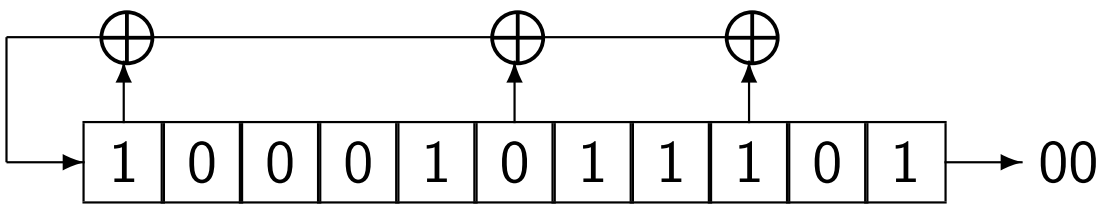
Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



- Im ersten Schritt wird das Register mit dem Schlüssel initialisiert
- In jedem Schritt (Takt), wird
  - ein Bit ausgegeben (jenes aus dem letzten Feld)
  - alles um eine Stelle nach rechts verschoben und
  - die ausgezeichneten Bits addiert und an die erste Stelle geschrieben

Stromchiffren: LFSR

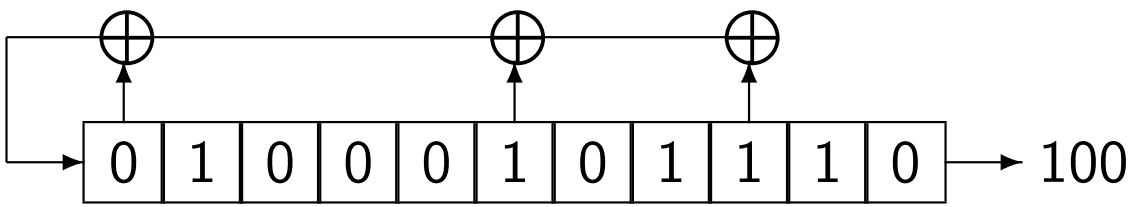
Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



- Im ersten Schritt wird das Register mit dem Schlüssel initialisiert
- In jedem Schritt (Takt), wird
  - ein Bit ausgegeben (jenes aus dem letzten Feld)
  - alles um eine Stelle nach rechts verschoben und
  - die ausgezeichneten Bits addiert und an die erste Stelle geschrieben

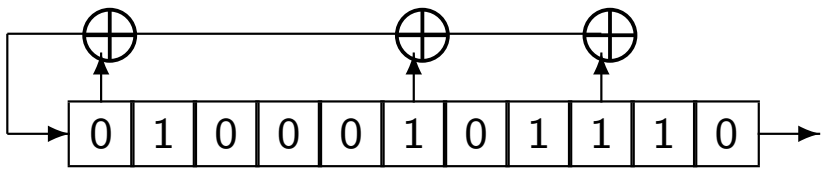
Stromchiffren: LFSR

Grundbaustein vieler Pseudozufallszahlengeneratoren für Stromchiffren: Linear Feedback Shift Register (LFSR) - Linear rückgekoppeltes Schieberegister



- Im ersten Schritt wird das Register mit dem Schlüssel initialisiert
- In jedem Schritt (Takt), wird
  - ein Bit ausgegeben (jenes aus dem letzten Feld)
  - alles um eine Stelle nach rechts verschoben und
  - die ausgezeichneten Bits addiert und an die erste Stelle geschrieben

Stromchiffren: Pseudozufallszahlengeneratoren

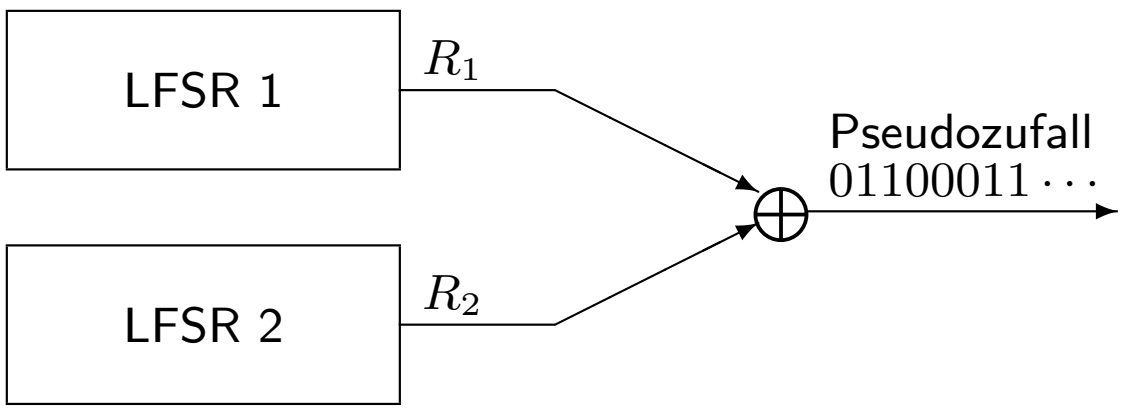


- LFSRs allein sind nicht geeignet für sichere Pseudozufallszahlengeneratoren
  (werden als Grundbausteine verwendet)
- Hier lassen sich bei bekanntem Teil des Schlüsselstroms Nachfolger berechnen:
  - Inhalt des obigen Schieberegisters ist nach 11 Takten Teil des Schlüsselstroms
  - Errät ein Angreifer diesen Teil, kann er alle Nachfolger berechnen

Kombination von LFSRs II

Es gibt mehrere Möglichkeiten, LFSRs so zu kombinieren, dass die (gravierende) Schwäche verhindert wird

- Summations-Generator:
  - Nutzung von zwei LFSRs (R_1, R_2)
  - Die Ausgaben von R_1 und R_2 werden addiert ($\oplus$)



Kombination von LFSRs II

Es gibt mehrere Möglichkeiten, LFSRs so zu kombinieren, dass die (gravierende) Schwäche verhindert wird

- Shrinking-Generator:
  - Nutzung von zwei LFSRs (R_1, R_2)
    - Wenn R_1 Bit 1 ausgibt, wird die Ausgabe von R_2 für Schlüsselstrom genutzt
    - Wenn R_1 Bit 0 ausgibt, wird die Ausgabe von R_2 verworfen
- Dadurch ist es einem Angreifer nicht möglich, aus dem Schlüsselstrom auf die internen Register der LFSRs zu schließen

Vorteile von Stromchiffren

- Statistisches Verhalten von LFSRs sind mathematisch sehr gut untersucht
- LFSRs sind sehr effizient in Hardware umzusetzen
  - Benötigen lediglich die Operationen $\oplus$ und Shift
- Ver- und Entschlüsselung sind sehr effizient

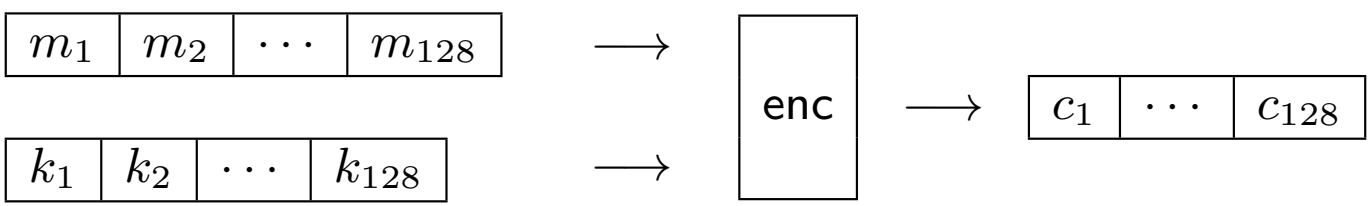
Blockchiffren I

- Blockchiffren bilden Bitstrings fester Länge auf Bitstrings der selben Länge ab:
  - Moderne Blockchiffren verarbeiten Bitstrings der Länge 128 (Klartextblock)
    (diese Länge heißt auch Blockgröße der Blockchiffre)
  - Genutzte Schlüssel müssen eine Bitlänge ≥ 100 haben
    (meist werden Schlüssel der Länge 128 Bit genutzt)

Blockchiffren I

- Blockchiffren bilden Bitstrings fester Länge auf Bitstrings der selben Länge ab:
  - Moderne Blockchiffren verarbeiten Bitstrings der Länge 128 (Klartextblock)
    (diese Länge heißt auch Blockgröße der Blockchiffre)
  - Genutzte Schlüssel müssen eine Bitlänge ≥ 100 haben
    (meist werden Schlüssel der Länge 128 Bit genutzt)

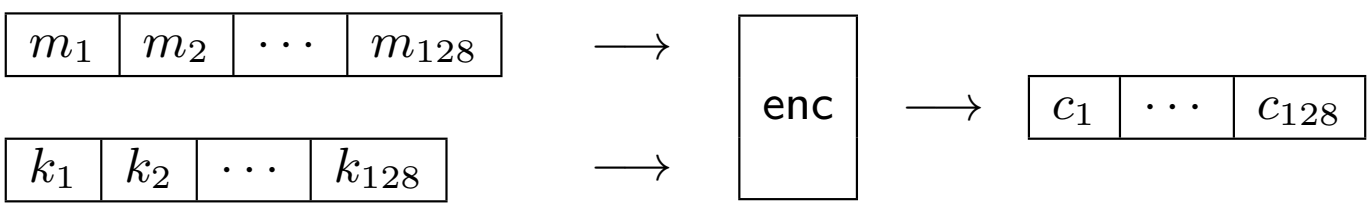
$$\text{enc} : \underbrace{\{0,1\}^{128}}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^{128}}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^{128}}_{\text{Chiphertext}}$$



Blockchiffren I

- Blockchiffren bilden Bitstrings fester Länge auf Bitstrings der selben Länge ab:
  - Moderne Blockchiffren verarbeiten Bitstrings der Länge 128 (Klartextblock)
    (diese Länge heißt auch Blockgröße der Blockchiffre)
  - Genutzte Schlüssel müssen eine Bitlänge ≥ 100 haben
    (meist werden Schlüssel der Länge 128 Bit genutzt)

$$\text{enc} : \underbrace{\{0,1\}^{128}}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^{128}}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^{128}}_{\text{Chiphertext}}$$



- Für längere Klartexte werden sogenannte Betriebsarten genutzt

Blockchiffren II

Wir behandeln zunächst nur die einzelne Blockchiffre

$$\text{enc} : \underbrace{\{0,1\}^{128}}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^{128}}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^{128}}_{\text{Chiphertext}}$$

Es gibt zwei etablierte Kontruktionsmethoden

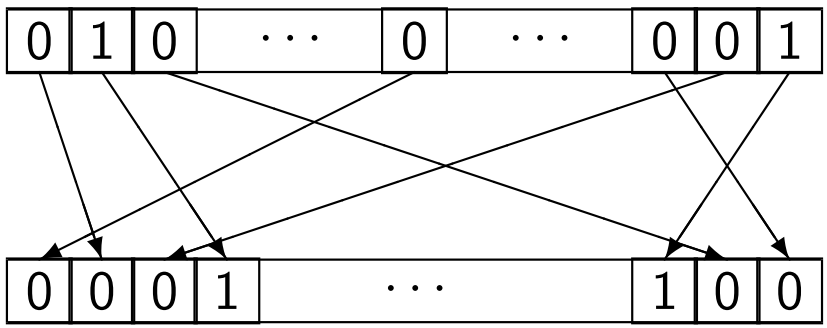
- Feistel-Chiffren:
  - Entwickelt von Horst Feistel in den 1970ger Jahren im IBM-Projekt Lucifer
  - Prominentes Beispiel: Data Encryption Standard (DES)
- Substitutions-Permutations-Netzwerk (SPN)
  - Grundidee geht auf Claude Shannon aus dem Jahr 1949 zurück
  - Prominentes Beispiel: Advanced Encryption Standard (AES), Nachfolger von DES

Blockchiffren III

Beide Konstruktionsmethoden nutzen die selben Ideen:

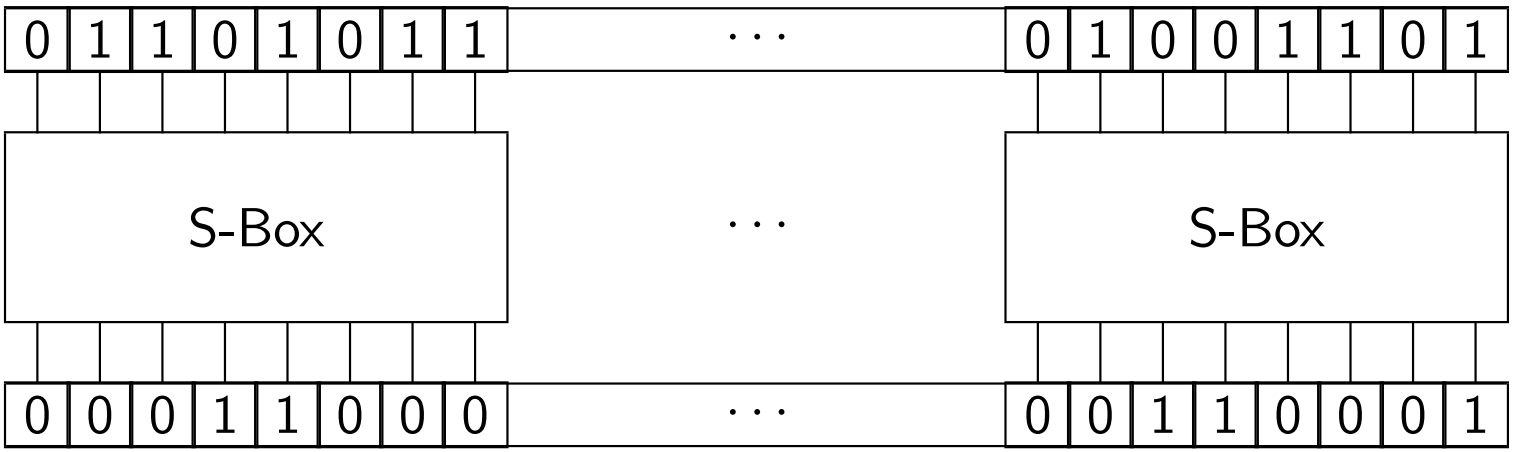
- Diffusion:
  - Verteilung der Informationen des Klartextes über den gesamten Chiffretext
    - Jedes Bit des Chiffretext hängt von jedem Bit des Klartext ab
    - Bei Änderung eines Klartextbits ändern sich ca. 50 % der Geheimtextbits
  - Realisiert durch Permutationen (spezielle lineare Abbildungen)
- Konfusion:
  - Zusammenhang zwischen Klartext und Chiffretext soll hochgradig komplex sein
    Gleiches gilt für den Zusammenhang zwischen Schlüssel und Chiffretext
  - Realisiert durch Substitutionen (nicht-lineare Abbildungen)
- Rundenbasiert: Wiederholte Anwendung von Diffusion, Konfusion und Schlüssel

Permutationen



- Reihenfolge der Bits werden über den gesamten Block (128 Bit) vertauscht
- Effekt der Diffusion ergibt sich erst nach mehreren Runden

Substitutionen



- S-Boxen sind nicht-lineare Abbildungen $\{0,1\}^n \longrightarrow \{0,1\}^n$
- Werden üblicherweise nur auf Bitstrings der Länge 8 oder 16 angewandt
- Damit effizient zu implementieren über Arrays der Länge 2^8 oder 2^{16}
  (Arrays der Länge 2^{128} sind nicht speicherbar)

Substitutions-Permutations-Netzwerk

Klartextblock (128 Bit)

Substitutions-Permutations-Netzwerk

Klartextblock (128 Bit)

1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab

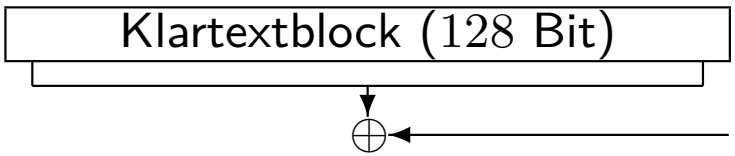
Substitutions-Permutations-Netzwerk

Klartextblock (128 Bit)

1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab

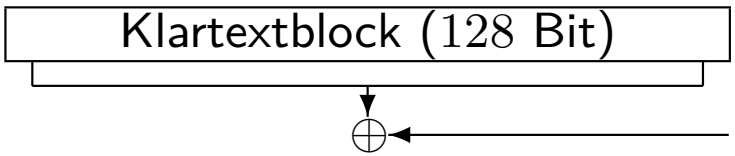
2) Runde 1: r_1 XOR Klartext

Substitutions-Permutations-Netzwerk



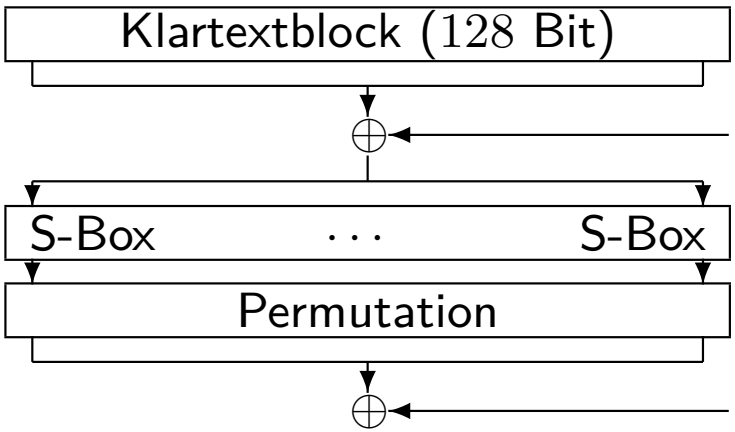
1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext

Substitutions-Permutations-Netzwerk



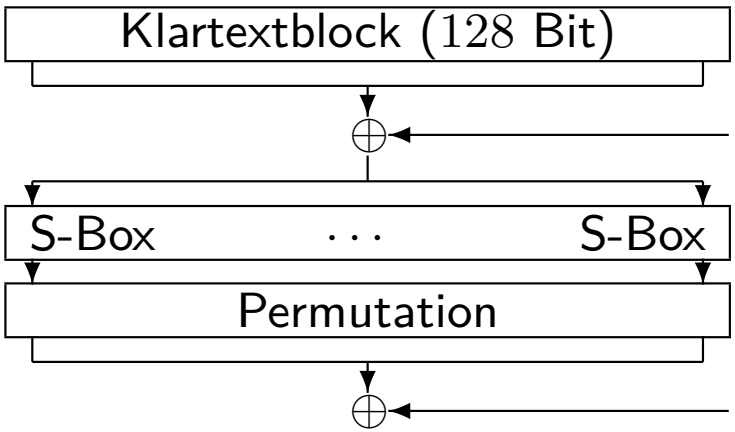
1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2

Substitutions-Permutations-Netzwerk



1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2

Substitutions-Permutations-Netzwerk

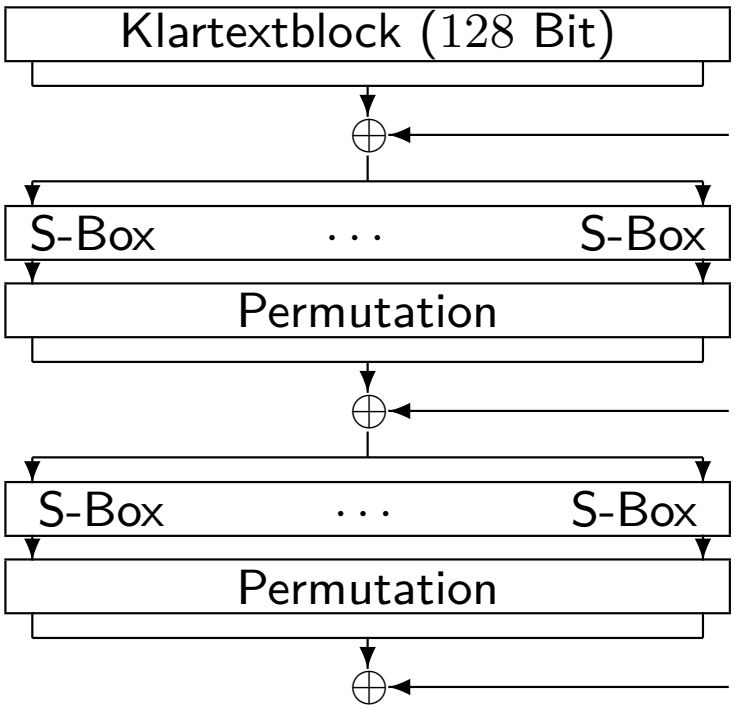


Runden-schlüssel 1

Runden-schlüssel 2

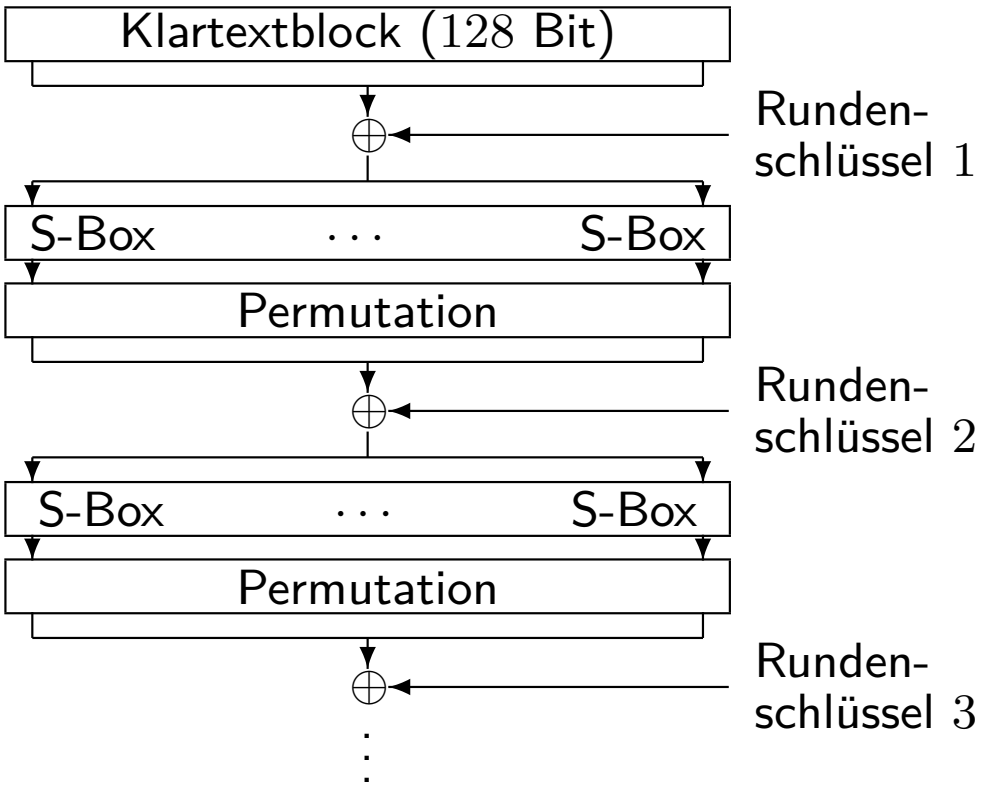
1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2
4) Runden 3 bis n laufen genauso

Substitutions-Permutations-Netzwerk



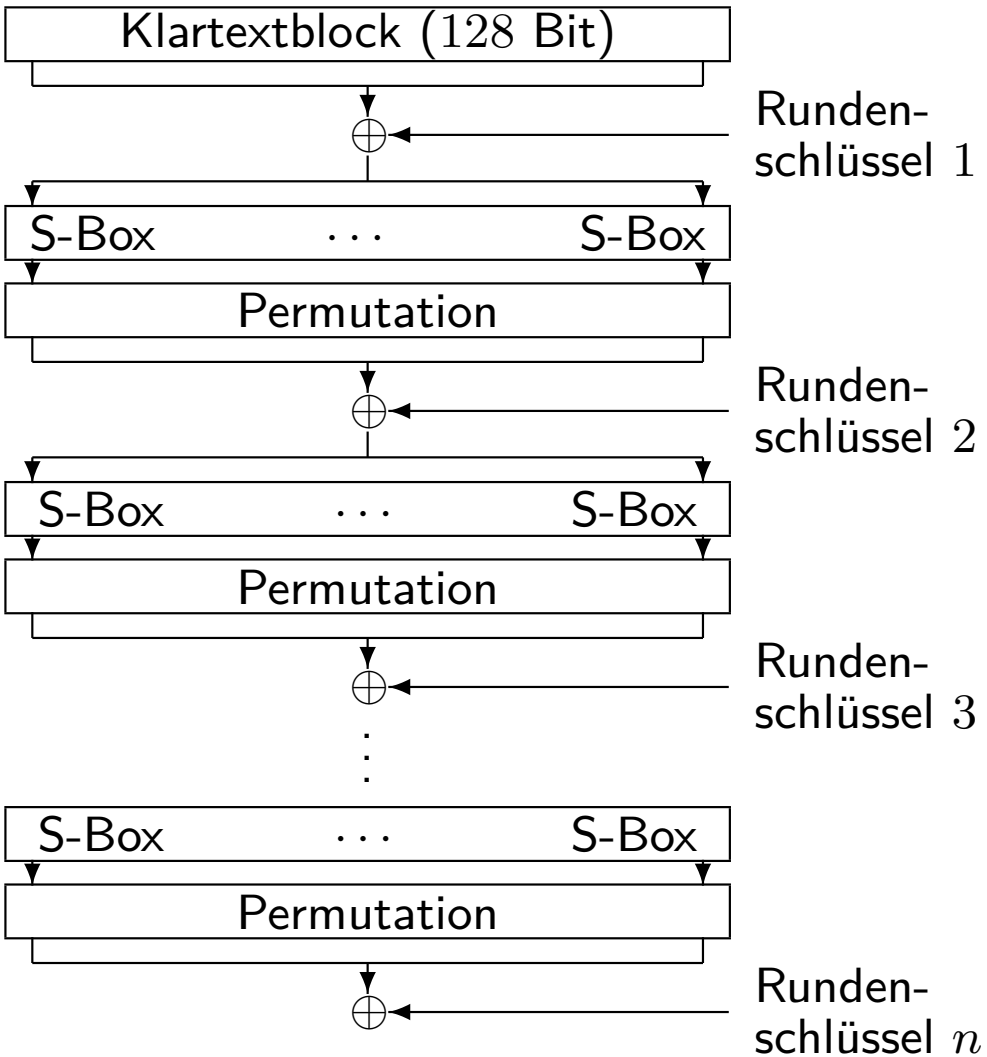
1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2
4) Runden 3 bis n laufen genauso

Substitutions-Permutations-Netzwerk



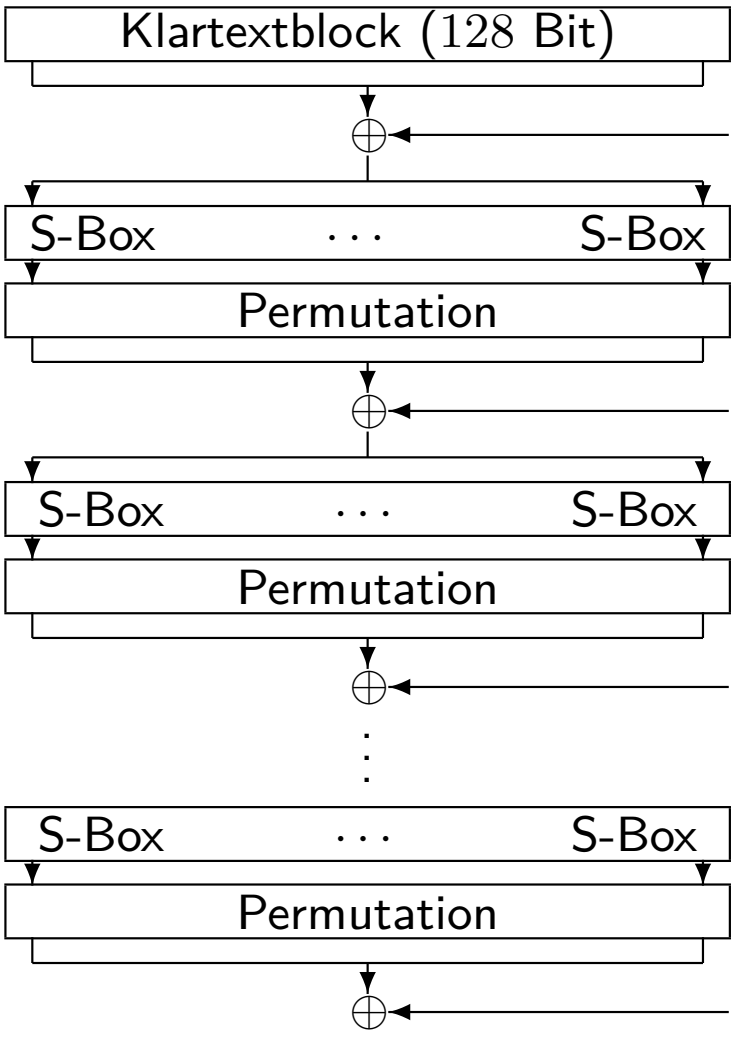
1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2
4) Runden 3 bis n laufen genauso

Substitutions-Permutations-Netzwerk



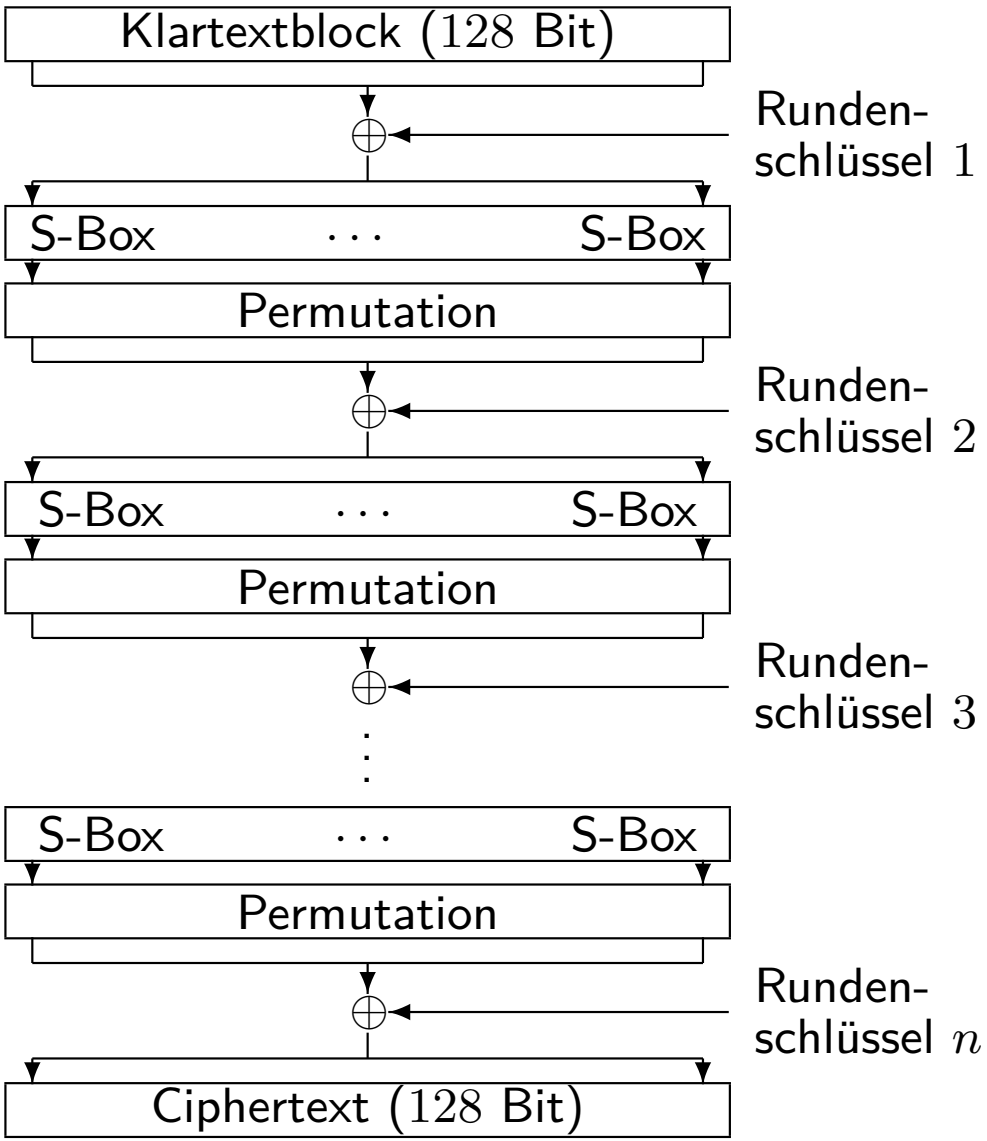
1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2
4) Runden 3 bis n laufen genauso

Substitutions-Permutations-Netzwerk



1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2
4) Runden 3 bis n laufen genauso
5) Runde n: Ciphertext

Substitutions-Permutations-Netzwerk



1) Aus $k \in \{0,1\}^{128}$ leitet man n Schlüssel $r_1, \ldots, r_n \in \{0,1\}^{128}$ ab
2) Runde 1: r_1 XOR Klartext
3) Runde 2: Substitution und Permutation, danach XOR r_2
4) Runden 3 bis n laufen genauso
5) Runde n: Ciphertext

Advanced Encryption Standard (AES)

AES ist eine Blockchiffre, die im Oktober 2000 vom National Institute of Standards and Technology (NIST) als Standard bekanntgegeben wurde.

- Schlüsselexpansion: 10,12 oder 14 Rundenschluessel müssen von dem Ersten Schluessel abgeleitet werden
  - Vorrunde: AddRoundKey(Schlüssel[0]): XOR mit Schlüssel
- Verschlüsselungsrunden (r = 1 bis R-1)
  - SubBytes(): vertausche Klartext Bytes mittels S-Box
  - ShiftRows(): shifte 4x4 bytes um einen festgesetzen wert
  - MixColumns(): vermische Spalten in 4x4 bytes
  - AddRoundKey(Schlüssel[r])
- Schlussrunde
  - SubBytes()
  - ShiftRows()
  - AddRoundKey(Schlüssel[R])

SPN: Sicherheitsdiskussion

Beide Grundbausteine eines SPN, P-Box und S-Box, werden für die Sicherheit benötigt:

- Wenn sich ein Bit des Klartextes oder Schlüssels ändert, ändern sich mehrere Bit der Ausgabe der S-Box
- Diese werden durch die P-Box auf verschiedene S-Boxen der nächsten Runde verteilt
- Lassen wir Substitutionen weg, dann ist die Blockchiffre eine lineaere Abbildung

Betriebsarten

- Blockchiffren verschlüsseln zunächst nur Klartexte einer bestimmten Blockgröße

$$\text{enc} : \underbrace{\{0,1\}^l}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^n}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^n}_{\text{Ciphertext}}$$

(Schlüssellänge $l \geq 100$, Blockgröße heute meist $n = 128$)

Betriebsarten

- Blockchiffren verschlüsseln zunächst nur Klartexte einer bestimmten Blockgröße

$$\text{enc} : \underbrace{\{0,1\}^l}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^n}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^n}_{\text{Ciphertext}}$$

  (Schlüssellänge $l \geq 100$, Blockgröße heute meist $n = 128$)
- Für die Verschlüsselung längerer Klartexte zerlegen wir den Klartext in Blöcke:

Betriebsarten

- Blockchiffren verschlüsseln zunächst nur Klartexte einer bestimmten Blockgröße

$$\text{enc} : \underbrace{\{0,1\}^l}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^n}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^n}_{\text{Ciphertext}}$$

  (Schlüssellänge $l \geq 100$, Blockgröße heute meist $n = 128$)
- Für die Verschlüsselung längerer Klartexte zerlegen wir den Klartext in Blöcke:

$$\underbrace{011000\cdots\cdots}_{\text{Block 1 (128 Bit)}}\ \underbrace{\cdots\cdots\cdots\cdots}_{\text{Block 2 (128 Bit)}}\ \cdots\cdots\cdots 1011110$$

Betriebsarten

- Blockchiffren verschlüsseln zunächst nur Klartexte einer bestimmten Blockgröße

$$\text{enc} : \underbrace{\{0,1\}^l}_{\text{Schlüssel}} \times \underbrace{\{0,1\}^n}_{\text{Klartexte}} \longrightarrow \underbrace{\{0,1\}^n}_{\text{Ciphertext}}$$

  (Schlüssellänge $l \geq 100$, Blockgröße heute meist $n = 128$)
- Für die Verschlüsselung längerer Klartexte zerlegen wir den Klartext in Blöcke:

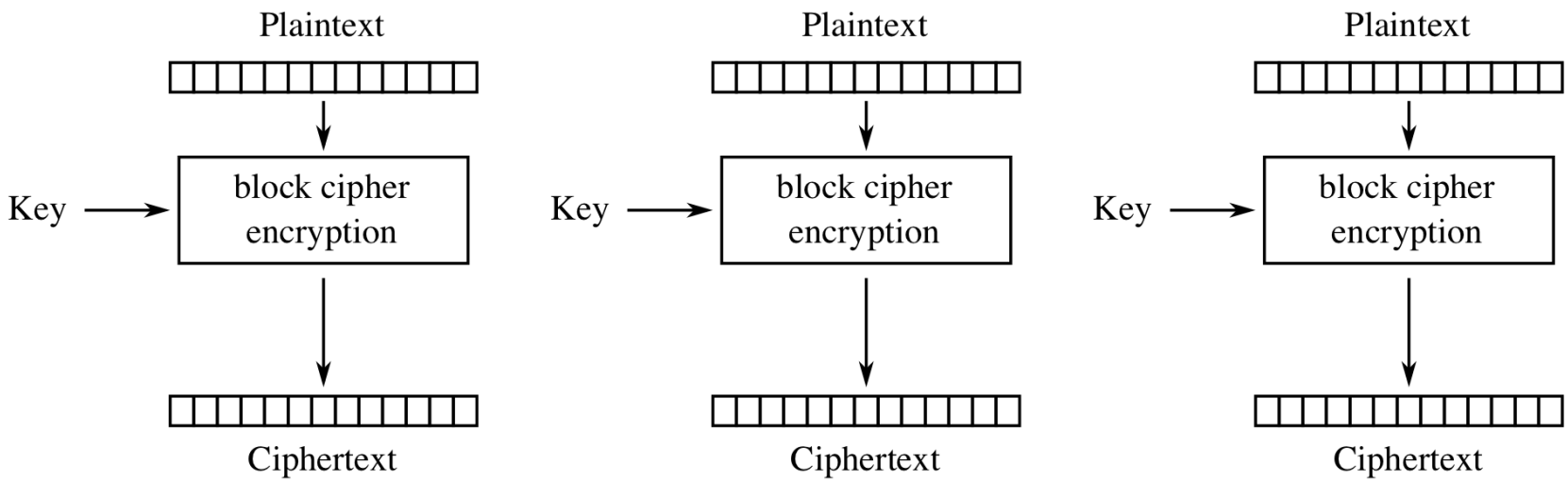
$$\underbrace{011000\cdots\cdots}_{\text{Block 1 (128 Bit)}}\ \underbrace{\cdots\cdots\cdots\cdots}_{\text{Block 2 (128 Bit)}}\ \cdots\cdots\cdots\underbrace{1011110\text{PADDING}}_{\text{Block s (128 Bit)}}$$

- Sollte der letzte Block nicht die Länge 128 haben, füllen wir entsprechend auf (mit 0er)

Betriebsart: Electronic Code Book (ECB)

Einfachste Betriebsart

- Der Klartext wird in Blöcke der benötigten Blocklänge zerlegt und Block für Block verschlüsselt

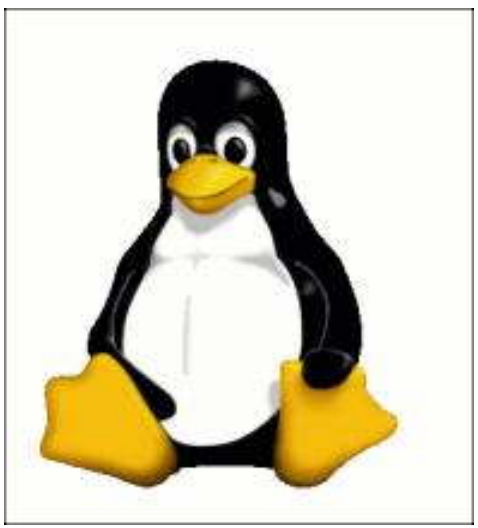


Electronic Codebook (ECB) mode encryption

Betriebsart: ECB

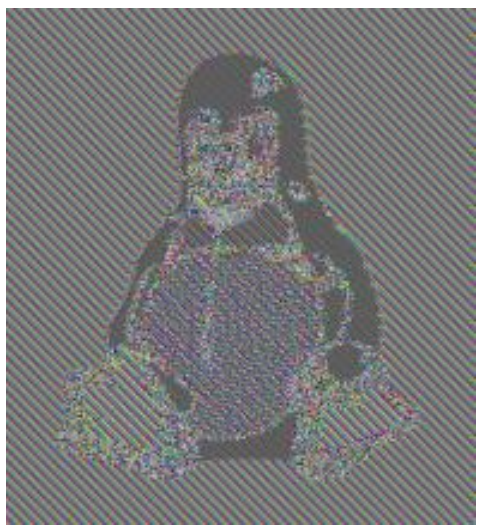
Probleme der Betriebsart ECB

- Gleiche Klartextblöcke werden zu gleichen Chiffretexten verschlüsselt (wird praktisch nicht verwendet)
- Ein Angreifer kann so die Struktur des Klartextes erkennen



Klartext

AES im ECB Mode



Chiffretext

Betriebsarten

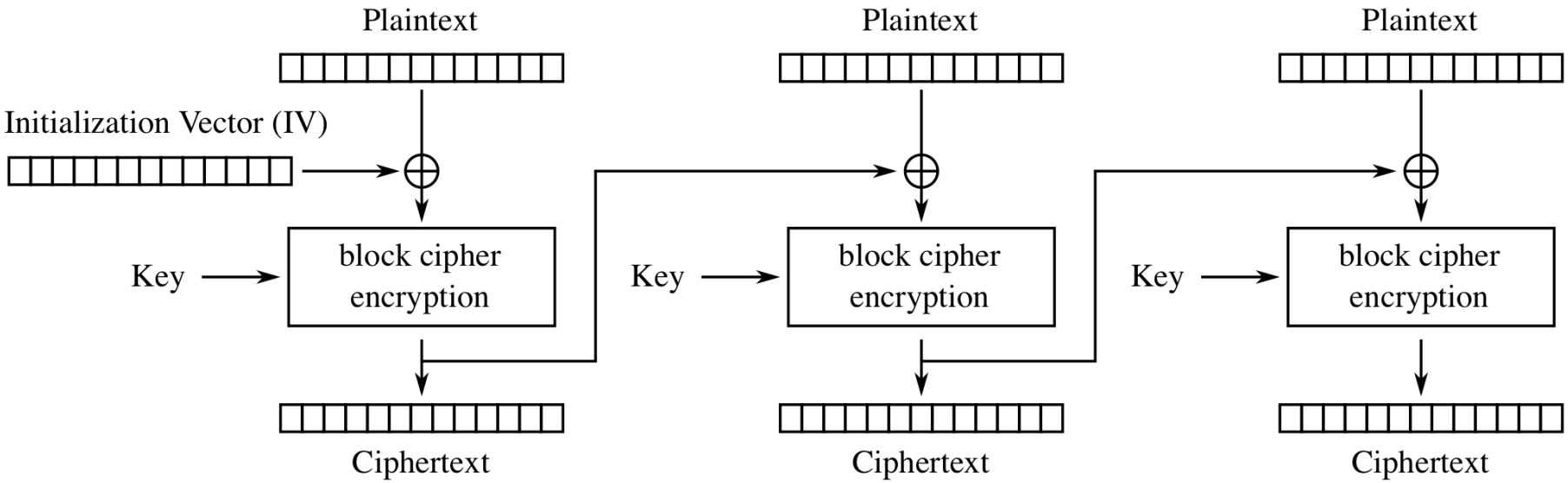
ECB ist nicht sicher

- Lösung:
  - Der Chiffretext sollte nicht nur von Schlüssel und Klartext abhängen,
  - sondern von einem weiteren Parameter
- Wir lernen zwei solcher Betriebsarten näher kennen
  - Cipher Block Chaining Mode (CBC)
  - Counter Mode (Ctr)
- Weitere bekannte Betreibsarten sind:
  - Cipher Feedback Mode (CFB)
  - Output Feedback Mode (OFB)

Betriebsart: Cipher Block Chaining I

Weiterer Parameter: Vorheriger Ciphertext

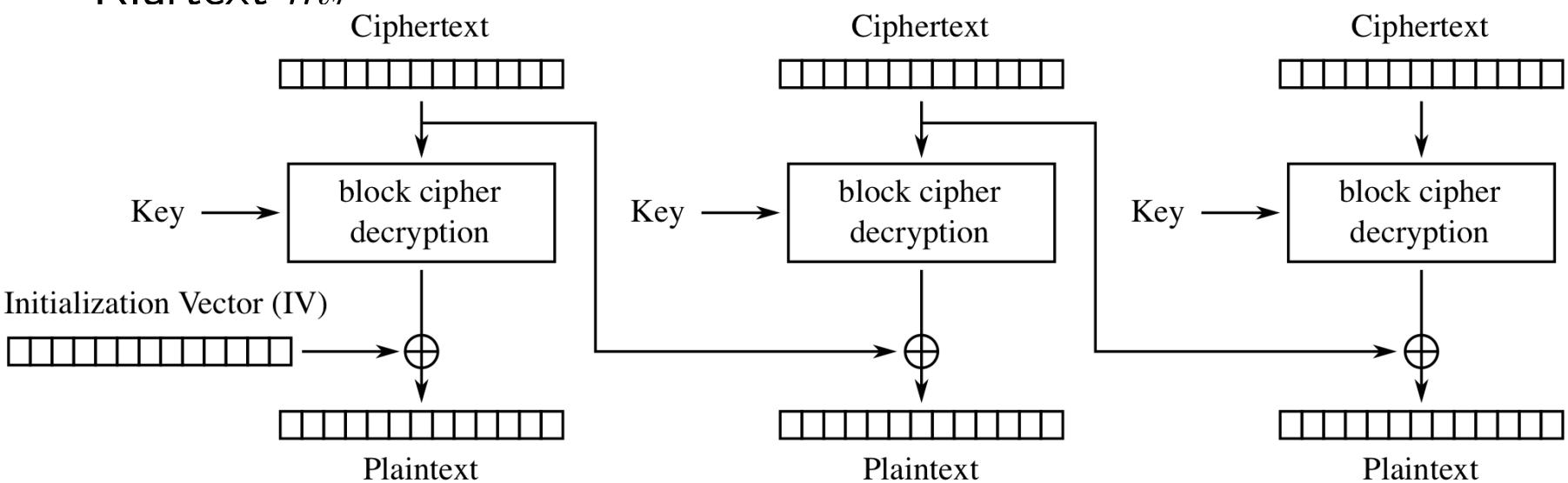
- Der $i-$te Klartext (Plaintext) m_i wird mit dem vorhergehenden Ciphertext c_{i-1} addiert und anschließend verschlüsselt: $c_i = \mathsf{enc}(k, m_i \oplus c_{i-1})$
- Für den ersten Block wird ein Initialisierungsvektor benötigt



Cipher Block Chaining (CBC) mode encryption

Betriebsart: Cipher Block Chaining II

- Für die Entschlüsselung wird der $i-$te Ciphertext c_i entschlüsselt und dann mit dem $(i-1)-$ten Ciphertext c_{i-1} addiert: $m_i = \mathsf{dec}(k, c_i) \oplus c_{i-1}$
- Wegen $c_i = \mathsf{enc}(k, m_i \oplus c_{i-1})$ erhalten wir tatsächlich den Klartext m_i

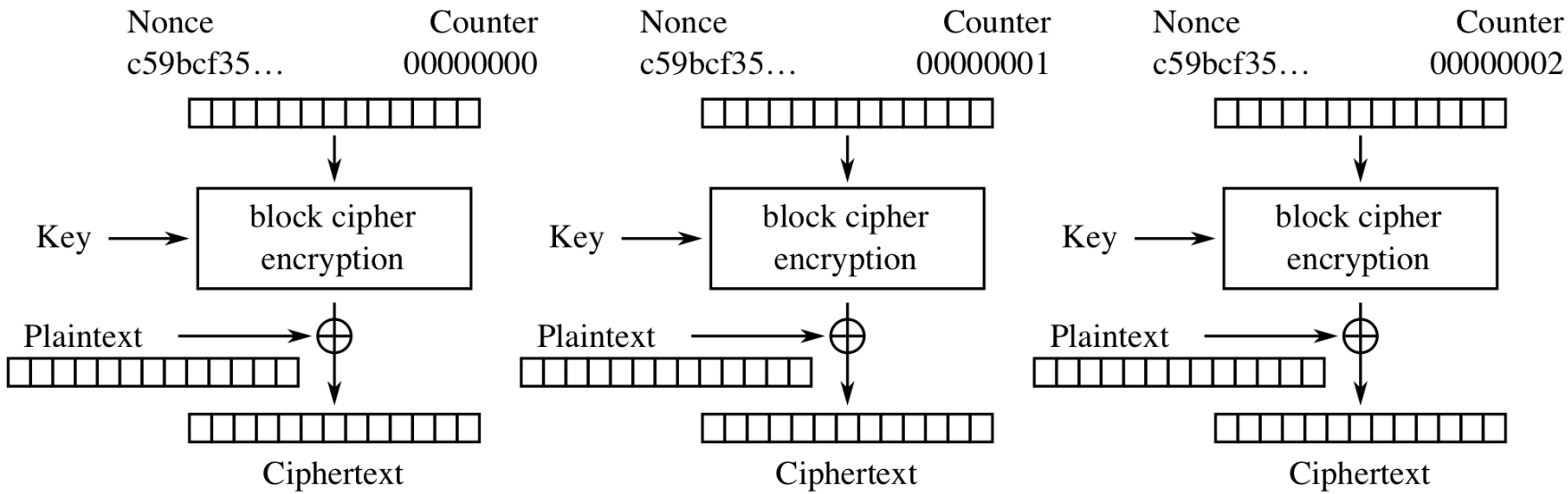


Cipher Block Chaining (CBC) mode decryption

Betriebsart: Counter Mode

Weiterer Parameter: Zähler (Counter)

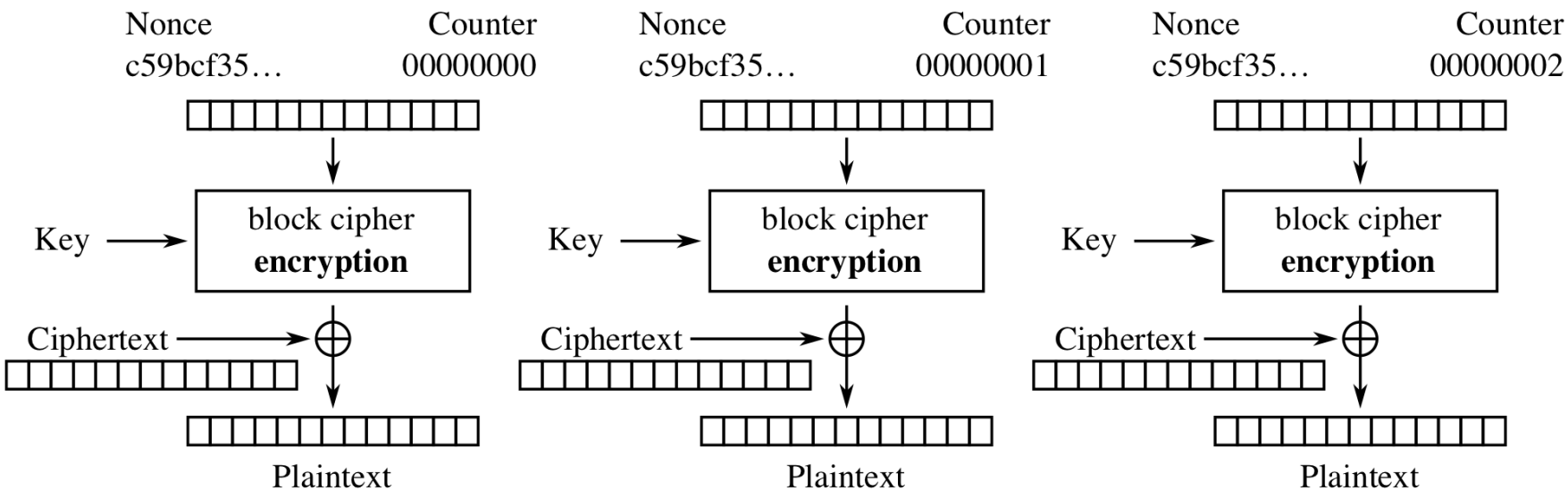
- Erzeuge Werte bestehend aus Nonce (Number Only Used Once) und Zähler
- Daraus wird ein Schlüsselstrom erzeugt und mit dem Klartext addiert (Die Blockchiffre wird also als PRNG genutzt)



Counter (CTR) mode encryption

Betriebsart: Counter Mode

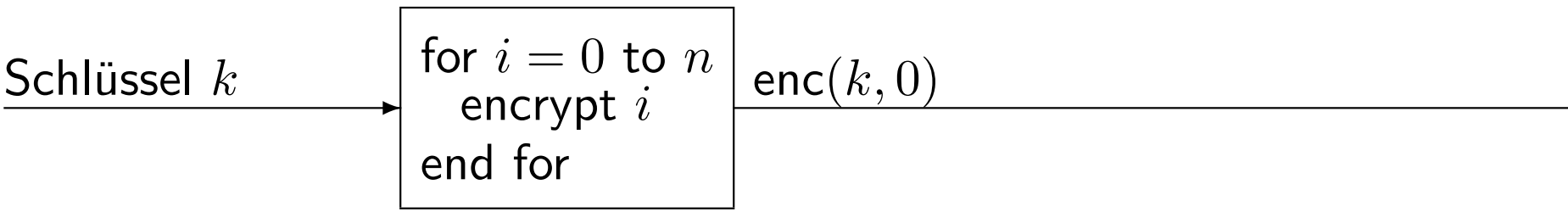
- Wie bei Stromchiffren üblich, erhalten wir Klartext zurück, indem wir den Schlüsselstrom auf den Ciphertext addieren (Schlüsselstrom $\oplus$ Schlüsselstrom $= (0, 0, 0, \dots)$)



Counter (CTR) mode decryption

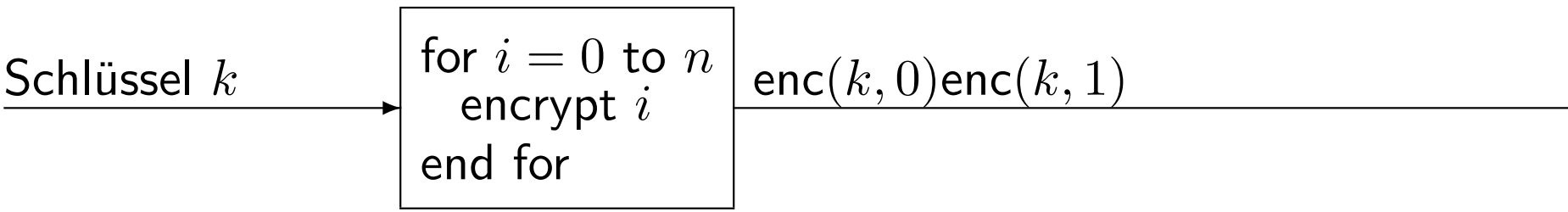
Sicherheit Counter Mode

Der Counter Mode nutzt die Blockchiffre als Pseudozufallszahlengenerator wie folgt:



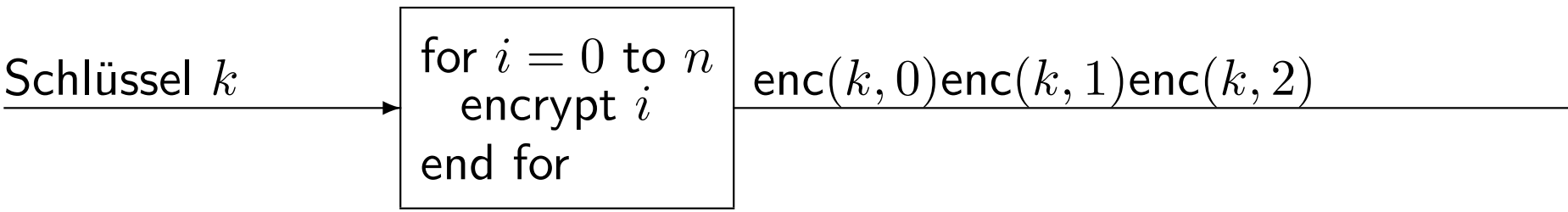
Sicherheit Counter Mode

Der Counter Mode nutzt die Blockchiffre als Pseudozufallszahlengenerator wie folgt:



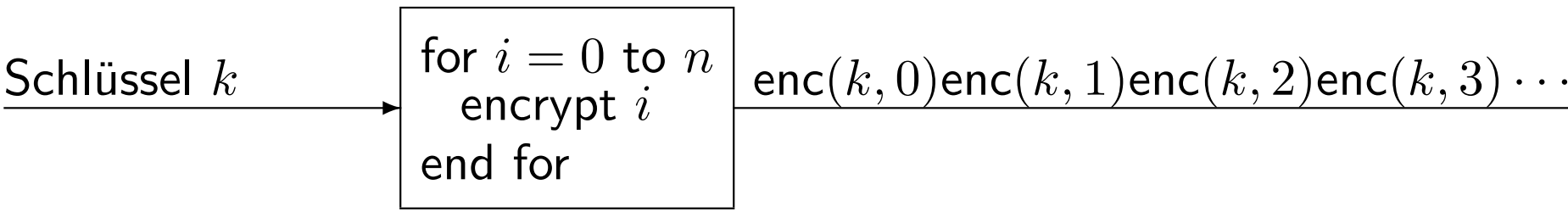
Sicherheit Counter Mode

Der Counter Mode nutzt die Blockchiffre als Pseudozufallszahlengenerator wie folgt:



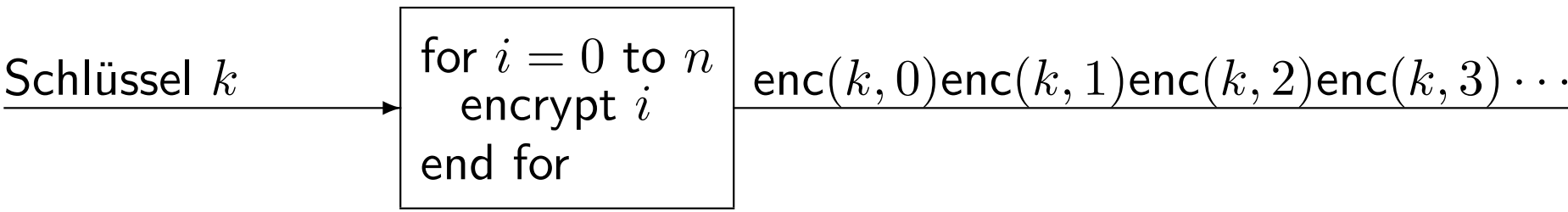
Sicherheit Counter Mode

Der Counter Mode nutzt die Blockchiffre als Pseudozufallszahlengenerator wie folgt:



Sicherheit Counter Mode

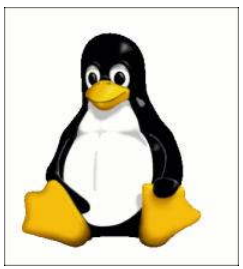
Der Counter Mode nutzt die Blockchiffre als Pseudozufallszahlengenerator wie folgt:



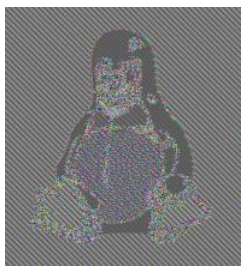
- Bedingungen für die Sicherheit:
  - Aus Teilen des Schlüsselstroms dürfen keine Nachfolger bestimmt werden können
  - Aus Teilen des Schlüsselstroms dürfen keine Vorgänger bestimmt werden können
- Wenn die Blockchiffre sicher ist (aus bekanntem Chiffetext kann der Schlüssel nicht berechnet werden), dann sind diese beiden Bedingungen erfüllt

ECB versus CBC und CTR

- ECB: Ciphertext lässt Struktur des Klartextes erkennen:



ECB Mode

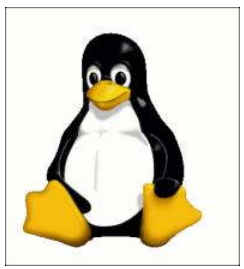


Klartext

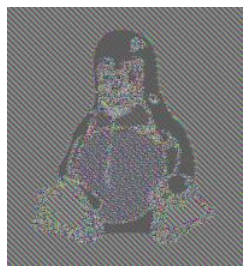
Chiffretext

ECB versus CBC und CTR

- ECB: Ciphertext lässt Struktur des Klartextes erkennen:



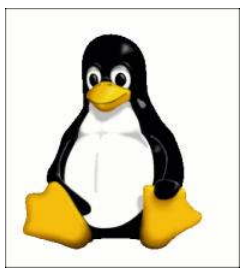
ECB Mode



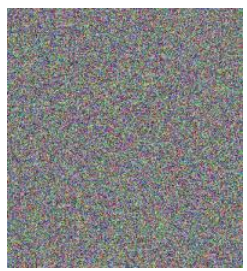
Klartext

Chiffretext

- Nutzung zusätzlicher Parameter (vorheriger Ciphertext (CBC), Zähler (CTR)):



CBC, CTR



Klartext

Chiffretext

Hashfunktionen I

Ziel: Integritätsschutz (erkennen, ob Daten verändert wurden)

Hashfunktionen bilden Bitstrings beliebiger Länge auf Bitstrings einer fester Länge l ab

- Hashfunktion: $H : \{0,1\}^* \longrightarrow \{0,1\}^l$
- Menge der Bitstrings beliebiger Länge: $\{0,1\}^*$ (Definitionsbereich von H)
- Menge der Bitstrings der Länge $l \in \mathbb{N}$: $\{0,1\}^l$ (Bildbereich von H)

Wie bei Verschlüsselungsverfahren werden häufig große Datenmengen verarbeitet

- Die Berechnung muss effizient sein, d.h. für gegebenes $m \in \{0,1\}^*$ muss $H(m)$ schnell berechnet werden können

Hashfunktionen II

Hashfunktionen werden für viele kryptographische Verfahren benötigt:

- Integritätsschutz
- Daten- und Instanzauthentisierung
- Elektronische Signatur
- Pseudozufallszahlengeneratoren

Beispiel:

```
md5sum Tux.jpg
28cd879dd7e59aa7fc8ac7b6c7939e61
```

MD5 erzeugt einen 512bit Hash einer Bild Datei

Hashfunktionen III

Definition kryptographische Hashfunktion

Wir nennen eine Hashfunktion $H : \{0,1\}^* \longrightarrow \{0,1\}^l$ **kryptographisch stark**, wenn sie die folgenden zwei Bedingungen erfüllt:

Preimage resistance: Für gegebenes $h \in \{0,1\}^l$ ist es praktisch nicht möglich einen Wert $m \in \{0,1\}^*$ mit $H(m) = h$ zu finden.

Collision resistance: Es ist praktisch nicht möglich zwei Werte $m, m' \in \{0,1\}^*$ so zu finden, dass $m \neq m'$ und $H(m) = H(m')$ gilt.

Hashfunktionen IV

Collision resistance: Es ist praktisch nicht möglich zwei Werte $m, m' \in \{0,1\}^*$ so zu finden, dass $m \neq m'$ und $H(m) = H(m')$ gilt.

Praktisch unmöglich:

- Bei einem geforderten Sicherheitsniveau von 100 Bit: Ein Angreifer benötigt ca. 2^{100} Versuche
  - einen Wert m mit $H(m) = h$ zu finden, oder
  - zwei Werte $m' \neq m$ mit $H(m) = H(m')$ zu finden

Hashfunktionen V

Preimage resistance: Für gegebenes $h \in \{0,1\}^l$ ist es praktisch nicht möglich einen Wert $m \in \{0,1\}^*$ mit $H(m) = h$ zu finden.

H soll effizient berechenbar sein, die Umkehrabbildung

(finde zu $h \in \{0,1\}^l$ Urbild $m \in \{0,1\}^*$ mit $H(m) = h$)

aber praktisch nicht berechenbar sein

Wir nennen solche Abbildungen auch Einwegfunktionen

Sicherheit von Hashfunktionen I

Collision resistance: Es ist praktisch nicht möglich zwei Werte $m, m' \in \{0,1\}^*$ so zu finden, dass $m \neq m'$ und $H(m) = H(m')$ gilt.

Zwei Werte $m, m' \in \{0,1\}^*$ mit $H(m) = H(m')$ nennen wir eine Kollision von H

Sicherheit von Hashfunktionen I

Collision resistance: Es ist praktisch nicht möglich zwei Werte $m, m' \in \{0,1\}^*$ so zu finden, dass $m \neq m'$ und $H(m) = H(m')$ gilt.

Zwei Werte $m, m' \in \{0,1\}^*$ mit $H(m) = H(m')$ nennen wir eine Kollision von H

- Kollisionen lassen sich nicht vermeiden:
  - H bildet $\{0,1\}^*$ (unendliche Menge) auf $\{0,1\}^l$ (endliche Menge, 2^l Elemente) ab
  - Damit muss es (sogar unendlich viele) Kollisionen geben

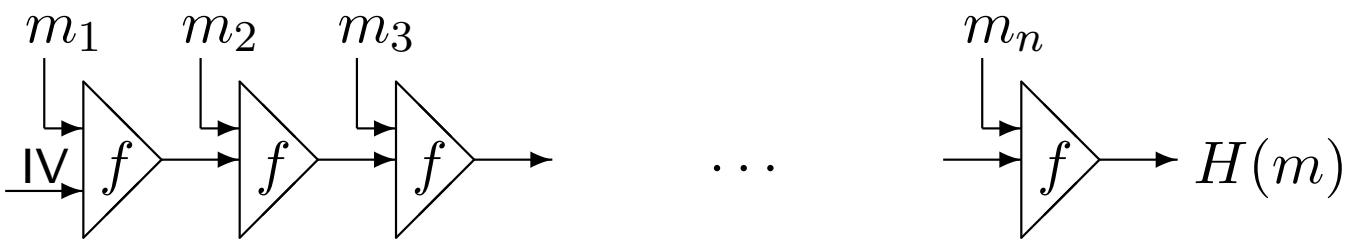
Sicherheit von Hashfunktionen II

Zwei Werte $m, m' \in \{0,1\}^*$ mit $H(m) = H(m')$ nennen wir eine Kollision von H

- Möglicher Angriff (Brute Force):
  - Wähle n zufällige Werte $m_1, \ldots, m_n \in \{0,1\}^*$
  - Prüfe, ob darunter zwei Werte sind, die eine Kollision von H bilden
- Praktisch nicht möglich (d.h. Sicherheitsniveau ≥ 100 Bit) bedeutet:
  - Ein Angreifer benötigt $\geq 2^{100}$ Versuche, bis er eine Kollision gefunden hat
  - d.h. muss mindestens 2^{100} Werte $m_1, \ldots, m_{2^{100}}$ wählen
- Um diesen Angriff auszuschließen, muss der Bildbereich von H groß sein
  - Mindestens 2^{200} Elemente, d.h. $H : \{0,1\}^* \longrightarrow \{0,1\}^l$ mit $l \geq 200$ (Beweis in der Vorlesung Kryptologie)

Konstruktion von Hashfunktionen I

Merkle-Damgard-Konstruktion:



- Nutze eine Kompressionsfunktion (basierend auf XOR)
  $f : \{0,1\}^l \times \{0,1\}^l \longrightarrow \{0,1\}^l$
  Diese muss sowohl eine Einwegfunktion, als auch kollisionsresistent sein
- m wird in Blöcke $m_1, \ldots, m_n$ der Länge l aufgeteilt
- Jeder Block wird zusammen mit dem vorherigen Ergebnis mittels f verarbeitet
- Für den ersten Schritt wird ein Initialsierungsvektor (IV) benötigt

Konstruktion von Hashfunktionen II

Bsp. für Kompressionsfunktion basierend auf einer Blockchiffre enc: (Matyas-Meyer-Oseas-Variante)

- Initialisierungsvektoren bei SHA2 sind immer gleich.
- Im ersten Schritt: $f_1 := f(\mathsf{IV}, m_1) = \mathsf{enc}(IV, m_1) \oplus m_1$
- Im zweiten Schritt: $f_2 := f(f_1, m_2) = \mathsf{enc}(f_1, m_2) \oplus m_2$
- ...
- Im letzten Schritt:
  $H(m) = f_n := f(f_{n-1}, m_n) = \mathsf{enc}(f_{n-1}, m_n) \oplus m_n$

Diese Variante wird sehr oft verwendet z.B. SHA-1, SHA-2, MD5

Hashfunktionen und Integritätsschutz I

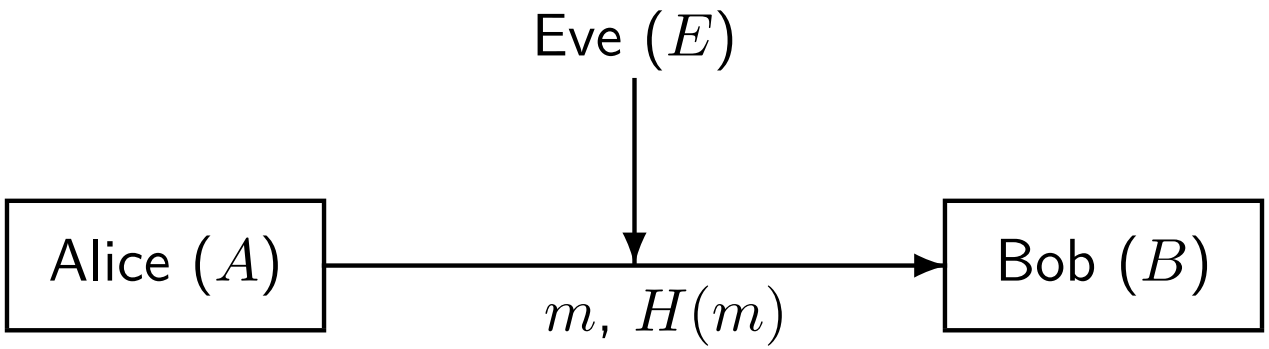
Ziel von Hashfunktionen: Integritätsschutz

- Speichern von Nachricht m und Hashwert $H(m)$ (auf verschiedenen Systemen)
- Prüfung, ob Nachricht m verändert wurde: Berechne Hashwert und vergleiche
- Hierfür wichtig: Kollisionsresistenz
  Ein Angreifer darf keine zweite Nachricht mit dem selben Hashwert erzeugen

Einsatzbeispiele:

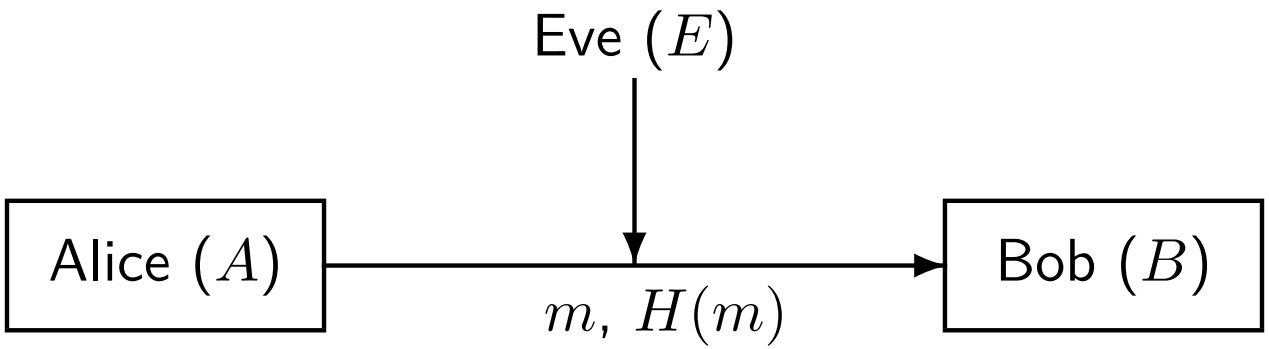
- Download von Software, z.B. OpenOffice von der Seite
  http://www.openoffice.org/de/downloads/index.html
  SHA256-Hashwert für
  Apache_OpenOffice_4.1.1_MacOS_x86-64_install_de.dmg:
  b905925d0d5bfb22e65910031cc1a17efce29498c4408a9deb368825ae8b236c

Hashfunktionen und Integritätsschutz II



Kann Eve die Nachricht trotzdem manipulieren?

Hashfunktionen und Integritätsschutz II



Kann Eve die Nachricht trotzdem manipulieren?

Es lässt sich aber nicht feststellen, wer die Nachricht erzeugt hat

- Hashfunktionen sind öffentlich
- Es wird kein kryptographischer Schlüssel verwendet

Hashfunktionen erfüllen also nicht das Schutzziel Datenauthentizität!

Message Authentication Codes (MAC) I

Schutzziel: Datenauthentizität
(B kann zweifelsfrei feststellen, dass die Nachricht von A stammt)

A Schlüssel: k		B Schlüssel: k
compute Prüfsumme $t :=$ $\mathsf{mac}(k, m)$		
	$\xrightarrow{m,t}$	
		compute $t' := \mathsf{mac}(k, m)$ if $t' = t$ then accept else reject

Message Authentication Codes (MAC) II

Schutzziel: Datenauthentizität

- Nur die Inhaber des Schlüssels k können korrekte Prüfsummen berechnen
- Erfüllen also das Schutzziel Datenauthentizität
  (wenn nur A und B den Schlüssel kennen)
- Erfüllen nicht das Schutzziel Nichtabstreitbarkeit
  - Beide kennen Schlüssel k, also können auch beide Prüfsumme berechnen
  - Gegenüber Dritten (z.B. einem Richter) kann B also nicht beweisen, dass nicht er, sondern A die Prüfsumme berechnet hat
  - Hierfür benötigen wir Signaturen (asymmetrische Verfahren)

Message Authentication Codes (MAC) III

Wie schon Verschlüsselungsverfahren und Hashfunktionen, müssen auch MAC-Verfahren zum Teil sehr große Datenmengen effizient verarbeiten können.

Heutige Verfahren basieren auf

- Blockchiffren (z.B. CBC-MAC, XOR-MAC)
- Kryptographisch starken Hashfunktionen (z.B. HMAC)

CBC-MAC I

Cipher Block Chaining Message Authentication Code (CBC-MAC)

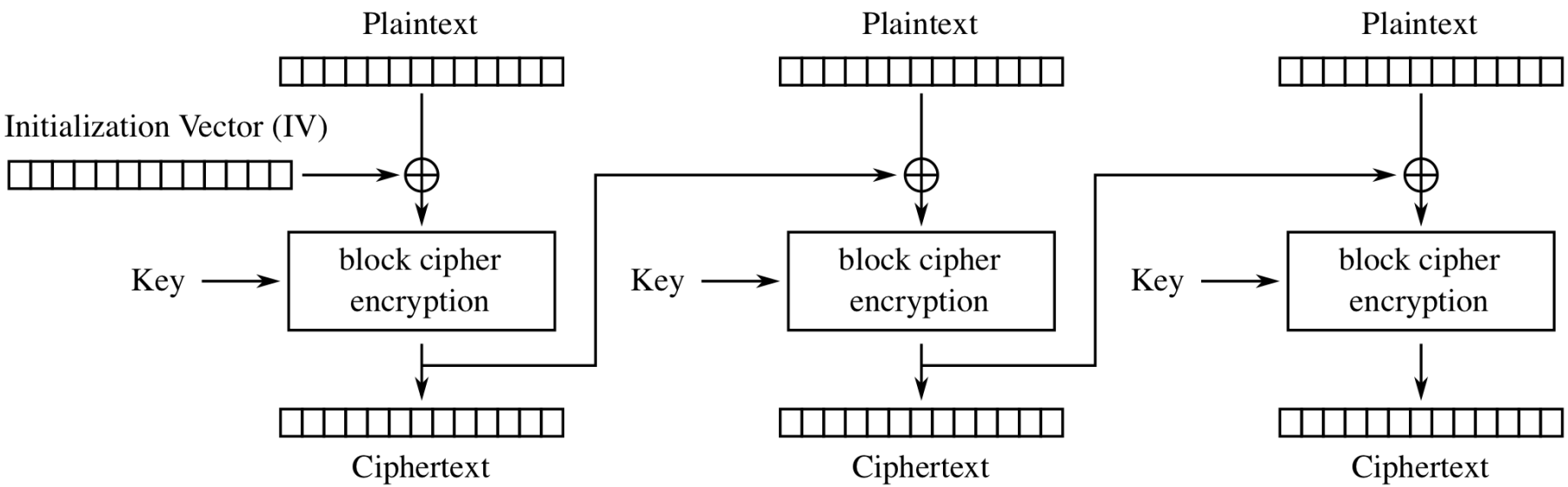
- Basiert auf einer sicheren Blockchiffre
- Nachricht m wird zunächst in Blöcke $m_1, \ldots, m_n$ zerlegt

CBC-MAC I

Cipher Block Chaining Message Authentication Code (CBC-MAC)

- Basiert auf einer sicheren Blockchiffre
- Nachricht m wird zunächst in Blöcke $m_1, \ldots, m_n$ zerlegt

Erinnerung Betriebsmode Cipher Block Chaining (CBC):



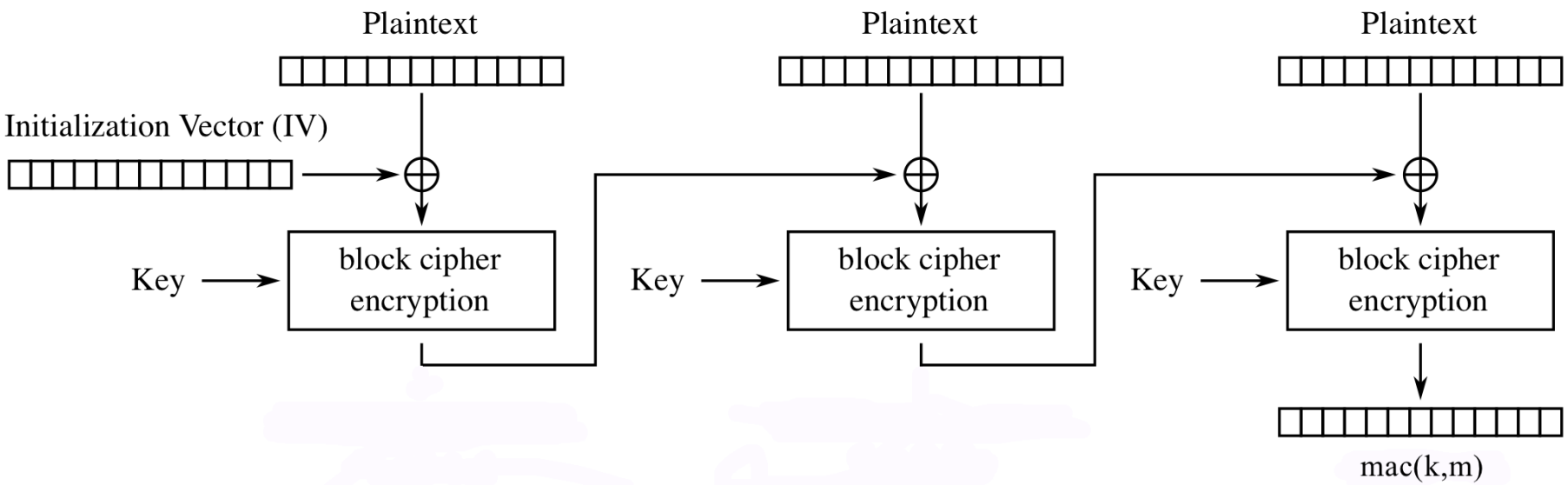
Cipher Block Chaining (CBC) mode encryption

CBC-MAC II

Cipher Block Chaining Message Authentication Code (CBC-MAC)

- Basiert auf einer sicheren Blockchiffre
- Nachricht m wird zunächst in Blöcke $m_1, \ldots, m_n$ zerlegt

Prüfsumme ist der letzte Ciphertext



Cipher Block Chaining Message Authentication Code (CBC-MAC)

HMAC

MAC-Verfahren basierend auf Hashfunktionen
$H : \{0,1\}^* \longrightarrow \{0,1\}^l$

$$\text{HMAC}(k, m) := H((k \oplus opad)||H((k \oplus ipad)||m))$$

Das Zeichen $||$ bedeutet Konkatenation (Verknüpfung)
Dabei sind

- $opad := 0x5C \cdots 0x5C$
- $ipad := 0x36 \cdots 0x36$

Konstanten (die Sicherheit hängt nicht von der konkreten Wahl der Konstanten ab)

Secure Messaging I

Ziel: Sicherer Kanal zwischen Alice und Bob (vertraulich und authentisch)

- Mehrere Umsetzungen denkbar:
  1) Verschlüsseln des Klartextes und MAC über den Geheimtext (z.B. bei IPsec)
  2) MAC über den Klartext und danach Verschlüsseln des Klartextes (z.B. bei SSH)
  3) MAC über den Klartext und Verschlüsseln von Klartext und MAC (z.B. bei SSL)
- Nur Umsetzung $1)$ ist sicher

Secure Messaging II

Erst Klartext verschlüsseln, dann MAC über den Geheimtext

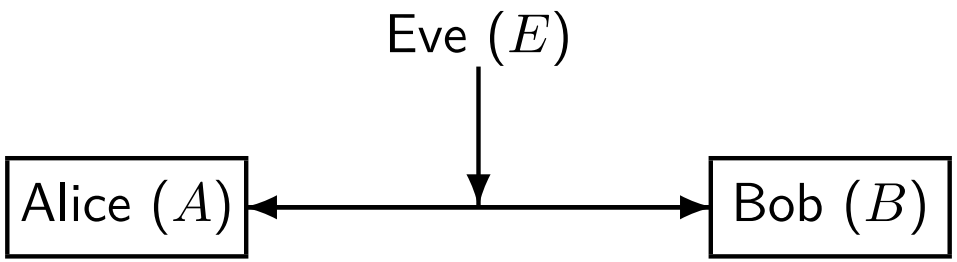
- Vertraulichkeit ist ein anderes Schutzziel als Datenauthentizität: MAC-Verfahren müssen nicht die Vertraulichkeit garantieren
- Weiterverarbeitung von Daten nur, wenn Sender bekannt ist: Verhindert z.B. Entschlüsselung von Schadsoftware

Weitere Bedingung:

- Für Verschlüsselung und MAC sollten verschiedene Schlüssel genutzt werden
  - Wesentliches kryptographisches Grundprinzip: Trenne wo du trennen kannst
  - Wird der Schlüssel für ein Verfahren kompromittiert, ist nicht automatisch das zweite Verfahren betroffen

Instanzauthentisierung

Schutzziel Instanzauthentisierung: B kann die Identität von A zweifelsfrei feststellen



Beispiele zur Instanzauthentisierung sind:

- Nachweis eines Geheimnisses (z.B. Benutzername/Passwort)
- Nachweis von Eigenschaften (z.B. biometrische Merkmale wie Fingerabdruck)
- Nachweis eines Geheimnisses, ohne dieses offen legen zu müssen (Challenge-Response Verfahren)

Challenge-Response Verfahren I

Ziel: Nachweis eines Geheimnisses (hier Schlüssel k), ohne dieses offen zu legen
Als Basis dienen Message Authentication Codes:

$\boxed{A}$ Schlüssel: k		$\boxed{B}$ Schlüssel: k
choose random c		
(challenge)	$\xrightarrow{c}$	
		compute $r := \mathsf{mac}(k, c)$
	$\xleftarrow{r}$	(response)
compute $r' := \mathsf{mac}(k, c)$		
if $r' = r$ then accept		
else reject		

Challenge-Response Verfahren II

Ziel: Nachweis eines Geheimnisses (hier Schlüsssel k), ohne dieses offen zu legen

- Nur die Inhaber des Schlüssels k können
  - aus der Challenge c
  - eine korrekte Response r

  berechnen
- Erfüllen also das Schutzziel Instanzauthentizität (wenn nur A und B den Schlüssel kennen)

Asymmetrische Verfahren I

Bei symm. Verfahren wird immer der selbe Schlüssel genutzt

- Schlüssel zum Ver- und Entschlüsseln sind gleich
- Schlüssel für Berechnung und Verifikation von Prüfsummen sind gleich
- Schlüssel zur Berechnung und Verifikation der Response sind gleich

Bei asymmetrischen Verfahren gibt es ein Schlüsselpaar (pk, sk)
pk : Public Key (ist öffentlich), sk : Secret Key (ist geheim)

Asymmetrische Verfahren I

Bei symm. Verfahren wird immer der selbe Schlüssel genutzt

- Schlüssel zum Ver- und Entschlüsseln sind gleich
- Schlüssel für Berechnung und Verifikation von Prüfsummen sind gleich
- Schlüssel zur Berechnung und Verifikation der Response sind gleich

Bei asymmetrischen Verfahren gibt es ein Schlüsselpaar (pk, sk)
pk : Public Key (ist öffentlich), sk : Secret Key (ist geheim)

- Verschlüsselung mit pk: jeder kann verschlüsseln
  Entschlüsselung mit sk: nur Inhaber des geheimen Schlüssels kann entschlüsseln
- Berechnung der Prüfsumme mit sk: nur Inhaber von sk kann berechnen
  Verifikation der Prüfsumme mit pk: jeder kann prüfen

Asymmetrische Verfahren II

Grundidee asymmetrischer Verfahren: Einwegfunktionen mit Falltür

- Einwegfunktion f (Begriff im Abschnitt Hashfunktionen kennen gelernt):
  - f ist effizient berechenbar
  - Umkehrfunktion von f ist praktisch nicht berechenbar
- Falltür:
  - Mit zusätzlichem Wissen lässt sich auch die Umkehrfunktion effizient berechnen

Kandidaten sind:

- Faktorisierungsproblem großer natürlicher Zahlen
  Multiplikation ist deutlich einfacher als Faktorisieren
- Das diskrete Logarithmusproblem
  Potenzieren ist deutlich einfacher als Logarithmus

Asymmetrische Verschlüsselung: RSA

RSA-Verfahren

- publiziert 1977
- benannt nach Rivest, Shamir und Adleman
- gilt als das erste bekannte asymmetrische Verfahren

Sicherheit beruht auf der vermuteten Schwierigkeit des Faktorisierens großer Zahlen

Faktorisierungsproblem:
Gegeben: Zusammengesetzte natürliche Zahl $n = p \cdot q \in \mathbb{N}$, p, q Primzahlen.
Lösung: Finde die beiden Primfaktoren p und q.

Einschub: Mathematische Grundlagen I

Für das RSA-Verfahren benötigen wir einige mathematische Grundlagen

- Die Menge $\mathbb{Z}_n$ (Die Menge aller Restklassen modulo n) besteht aus allen natürlichen Zahlen $< n$
  $\mathbb{Z}_n := \{0, 1, 2, \ldots, n-1\}$
- Für eine natürliche Zahl $a \in \mathbb{N}$ sei $t = a \bmod n$ diejenige eindeutige Zahl aus $\mathbb{Z}_n$, die aus a durch (ev. mehrmaliges) Subtrahieren von n entsteht
- $a \mod n = t \Rightarrow a - k \cdot n = t$ bzw. $a = k \cdot n + t$, mit $k \in \mathbb{Z}$
- Beispiel: $28 \mod 13 = 2, 28 - 2 \cdot 13 = 2$ bzw. $28 = 2 \cdot 13 + 2$

Einschub: Mathematische Grundlagen II

Für das RSA-Verfahren benötigen wir einige mathematische Grundlagen

- Die Menge $\mathbb{Z}_n$ besteht aus allen natürlichen Zahlen $< n$
  $\mathbb{Z}_n := \{0, 1, 2, \ldots, n-1\}$
- Wir rechnen in dieser Menge wie folgt:
  - Für Addition und Multiplikation in $\mathbb{Z}_n$ gilt nun:

$$\begin{aligned} a + b &:= a + b \bmod n \\ a \cdot b &:= a \cdot b \bmod n \end{aligned}$$

  - Beispiel in $\mathbb{Z}_4 = \{0, 1, 2, 3\} : 2 + 3 \bmod 4 = 5 \bmod 4 = 1$

Einschub: Mathematische Grundlagen III

Weitere Eigenschaften der Menge $\mathbb{Z}_n$:
Inverse Elemente bzgl. Addition und Multiplikation in $\mathbb{Z}_n$

- $a \in \mathbb{Z}_n$ heißt additiv invers (modulo n) zu $b \in \mathbb{Z}_n$, wenn

$$a + b = 0 \bmod n$$

- $a \in \mathbb{Z}_n$ heißt multiplikativ invers (modulo n) zu $b \in \mathbb{Z}_n$, wenn

$$a \cdot b = 1 \bmod n$$

Einschub: Mathematische Grundlagen III

Weitere Eigenschaften der Menge $\mathbb{Z}_n$:
Inverse Elemente bzgl. Addition und Multiplikation in $\mathbb{Z}_n$

- $a \in \mathbb{Z}_n$ heißt additiv invers (modulo n) zu $b \in \mathbb{Z}_n$, wenn

$$a + b = 0 \bmod n$$

- $a \in \mathbb{Z}_n$ heißt multiplikativ invers (modulo n) zu $b \in \mathbb{Z}_n$, wenn

$$a \cdot b = 1 \bmod n$$

- Beispiele in $\mathbb{Z}_4$:
  - $3 + 1 = 4 = 0 \bmod 4$ (1 ist additiv invers zu 3)
  - $3 \cdot 3 = 9 = 1 \bmod 4$ (3 ist multiplikativ invers zu sich selbst)

Einschub: Mathematische Grundlagen IV

Welche Elemente in $\mathbb{Z}_n$ haben Inverse?

- Bzgl. Addition alle Elemente: für alle $a \in \mathbb{Z}_n$ gilt mit $b = n - a$:

$$a + b = a + (n - a) = n = 0 \bmod n$$

Einschub: Mathematische Grundlagen IV

Welche Elemente in $\mathbb{Z}_n$ haben Inverse?

- Bzgl. Addition alle Elemente: für alle $a \in \mathbb{Z}_n$ gilt mit $b = n - a$:

$$a + b = a + (n - a) = n = 0 \bmod n$$

- Bzgl. Multiplikation nicht alle Elemente: z.B. in $\mathbb{Z}_4$
  - 1 ist zu sich selbst invers $(1 \cdot 1 = 1)$
  - 3 ist zu sich selbst invers (hatten wir schon)
  - zu 2 gibt es kein inverses Element
    $(2 \cdot 1 = 2 \neq 1,\ 2 \cdot 2 = 4 = 0 \bmod 4 \neq 1,$
    $2 \cdot 3 = 6 = 2 \bmod 4 \neq 1)$

Einschub: Mathematische Grundlagen V

Welche Elemente in $\mathbb{Z}_n$ haben multiplikativ Inverse?

- Für $a \in \mathbb{N}$ heißt $t \in \mathbb{N}$ *Teiler* von a (in Zeichen $t|a$), wenn $\frac{a}{t} \in \mathbb{N}$, $1, 2, 3, 4, 6$ sind Teiler von 12
- $t \in \mathbb{N}$ heißt *gemeinsamer Teiler* von $a, b \in \mathbb{N}$, wenn $t|a$ und $t|b$
  $1, 2, 4$ sind gemeinsame Teiler von 12 und 20
- $a, b \in \mathbb{N}$ heißen *teilerfremd*, wenn 1 einziger gemeinsamer Teiler von a und b ist, ggT$(a,b)=1$
  8 und 15 sind teilerfrend

Ein Element $a \in \mathbb{Z}_n$ besitzt genau dann ein multiplikativ Inverses modulo n, wenn a und n teilerfremd sind

- 1 und 3 teilerfremd zu 4, 2 ist nicht teilerfremd zu 4

Einschub: Eulersche Phi-Funktion

Die Eulersche Phi-Funktion gibt zu jeder natürliche Zahl a an, wie viele zu a teilerfremde natürliche Zahlen es gibt, die nicht größer als a sind: $\phi(a) := |\{n \in \mathbb{N} | 1 \leq n < a \wedge \text{ ggT}(n, a) = 1\}|$

Beispiel:

- $\phi(6) =$

Einschub: Eulersche Phi-Funktion

Die Eulersche Phi-Funktion gibt zu jeder natürliche Zahl a an, wie viele zu a teilerfremde natürliche Zahlen es gibt, die nicht größer als a sind: $\phi(a) := |\{n \in \mathbb{N} | 1 \leq n < a \wedge \text{ ggT}(n, a) = 1\}|$

Beispiel:

- $\phi(6) = |\{1, 5\}| = 2$
- $\phi(8) =$

Einschub: Eulersche Phi-Funktion

Die Eulersche Phi-Funktion gibt zu jeder natürliche Zahl a an, wie viele zu a teilerfremde natürliche Zahlen es gibt, die nicht größer als a sind: $\phi(a) := |\{n \in \mathbb{N} | 1 \leq n < a \wedge \text{ ggT}(n, a) = 1\}|$

Beispiel:

- $\phi(6) = |\{1, 5\}| = 2$
- $\phi(8) = |\{1, 3, 5, 7\}| = 4$
- $\phi(19) =$

Einschub: Eulersche Phi-Funktion

Die Eulersche Phi-Funktion gibt zu jeder natürliche Zahl a an, wie viele zu a teilerfremde natürliche Zahlen es gibt, die nicht größer als a sind: $\phi(a) := |\{n \in \mathbb{N} | 1 \leq n < a \wedge \text{ ggT}(n, a) = 1\}|$

Beispiel:

- $\phi(6) = |\{1, 5\}| = 2$
- $\phi(8) = |\{1, 3, 5, 7\}| = 4$
- $\phi(19) = |\{1, 2, \ldots, 18\}| = 18$

Einschub: Eulersche Phi-Funktion

Die Eulersche Phi-Funktion gibt zu jeder natürliche Zahl a an, wie viele zu a teilerfremde natürliche Zahlen es gibt, die nicht größer als a sind: $\phi(a) := |\{n \in \mathbb{N} | 1 \leq n < a \wedge \text{ ggT}(n, a) = 1\}|$

Beispiel:

- $\phi(6) = |\{1, 5\}| = 2$
- $\phi(8) = |\{1, 3, 5, 7\}| = 4$
- $\phi(19) = |\{1, 2, \ldots, 18\}| = 18$

$\Rightarrow$ sei p eine Primzahl, dann gilt $\phi(p) = p - 1$

Einschub: Satz von Euler I

$$\text{ggT(a,b)=1} \Rightarrow a^{\phi(b)} \mod b = 1$$

- Betrachte alle zu b teilerfremden Zahlen in $\mathbb{Z}_b$: $k_1, k_2, \ldots, k_{\phi(b)}$
- Multipliziere nun alle k_i mit a: $ak_1, ak_2, \ldots, ak_{\phi(b)}$
- Dadurch können keine neuen Restklassen entstehen!
  $\Rightarrow \text{ggT}(ak_i, b) = 1$

Einschub: Satz von Euler I

$$\text{ggT(a,b)=1} \Rightarrow a^{\phi(b)} \mod b = 1$$

- Betrachte alle zu b teilerfremden Zahlen in $\mathbb{Z}_b$: $k_1, k_2, \ldots, k_{\phi(b)}$
- Multipliziere nun alle k_i mit a: $ak_1, ak_2, \ldots, ak_{\phi(b)}$
- Dadurch können keine neuen Restklassen entstehen!
  $\Rightarrow \text{ggT}(ak_i, b) = 1$

Beispiel:
$b = 8$, $\phi(8) = 4$
$k_1, k_2, \ldots, k_{\phi(b)}=$

Einschub: Satz von Euler I

$$\text{ggT(a,b)=1} \Rightarrow a^{\phi(b)} \mod b = 1$$

- Betrachte alle zu b teilerfremden Zahlen in $\mathbb{Z}_b$: $k_1, k_2, \ldots, k_{\phi(b)}$
- Multipliziere nun alle k_i mit a: $ak_1, ak_2, \ldots, ak_{\phi(b)}$
- Dadurch können keine neuen Restklassen entstehen!
  $\Rightarrow \text{ggT}(ak_i, b) = 1$

Beispiel:
$b = 8,\ \phi(8) = 4$
$k_1, k_2, \ldots, k_{\phi(b)}=1, 3, 5, 7$
Wähle $a = 3 \Rightarrow$
$3 \cdot 1 \mod 8, 3 \cdot 3 \mod 8, 3 \cdot 5 \mod 8, 3 \cdot 7 \mod 8 =$

Einschub: Satz von Euler I

$$\text{ggT(a,b)=1} \Rightarrow a^{\phi(b)} \mod b = 1$$

- Betrachte alle zu b teilerfremden Zahlen in $\mathbb{Z}_b$: $k_1, k_2, \ldots, k_{\phi(b)}$
- Multipliziere nun alle k_i mit a: $ak_1, ak_2, \ldots, ak_{\phi(b)}$
- Dadurch können keine neuen Restklassen entstehen!
  $\Rightarrow \text{ggT}(ak_i, b) = 1$

Beispiel:
$b = 8$, $\phi(8) = 4$
$k_1, k_2, \ldots, k_{\phi(b)}$=$1, 3, 5, 7$
Wähle $a = 3 \Rightarrow$
$3 \cdot 1 \mod 8, 3 \cdot 3 \mod 8, 3 \cdot 5 \mod 8, 3 \cdot 7 \mod 8 = 3, 1, 7, 5$

Einschub: Satz von Euler II

$$k_1 \cdot k_2 \cdot \dots \cdot k_{\phi(b)} = ak_1 \cdot ak_2 \cdot \dots \cdot ak_{\phi(b)} \mod b$$

Einschub: Satz von Euler II

$$k_1 \cdot k_2 \cdot \dots \cdot k_{\phi(b)} = ak_1 \cdot ak_2 \cdot \dots \cdot ak_{\phi(b)} \mod b$$
$$1 = a^{\phi(b)} \mod b$$

Einschub: Satz von Euler II

$$k_1 \cdot k_2 \cdot \dots \cdot k_{\phi(b)} = ak_1 \cdot ak_2 \cdot \dots \cdot ak_{\phi(b)} \mod b$$
$$1 = a^{\phi(b)} \mod b$$

Der kleine fermatsche Satz, kurz “der kleine Fermat” besagt:
Sei p eine Primzahl dann gilt,

$$a = a^p \mod p.$$

Einschub: Satz von Euler II

$$k_1 \cdot k_2 \cdot \dots \cdot k_{\phi(b)} = ak_1 \cdot ak_2 \cdot \dots \cdot ak_{\phi(b)} \mod b$$
$$1 = a^{\phi(b)} \mod b$$

Der kleine fermatsche Satz, kurz "der kleine Fermat" besagt: Sei p eine Primzahl dann gilt,

$$a = a^p \mod p.$$

Dieser Satz lässt sich aus dem Satz von Euler ableiten da,

$$1 = a^{\phi(p)} \mod p = a^{p-1} \mod p$$
$$a = a \cdot 1 = a \cdot a^{p-1} \mod p = a^p \mod p$$

RSA-Verschlüsselungsverfahren

Schlüsselgenerierung:

- Wähle zwei Primzahlen: p, q. Wir nennen $n = p \cdot q$ das *Modul*
- $\phi(n) = (p-1) \cdot (q-1)$ heißt auch *Euler-Zahl* von n
- Verschlüsselungsschlüssel (pk): Wähle e teilerfremd zu $\phi(n)$
- Entschlüsselungsschlüssel (sk): Wähle d so, dass $1 = e \cdot d \bmod \phi(n)$
  (Solch ein d existiert, da e teilerfremd zu $\phi(n)$ ist)

Verschlüsselung einer Nachricht $m \in \mathbb{Z}_n$

- Berechne $c = m^e \bmod n$

Entschlüsselung des Ciphertextes c

- Berechne $c^d \bmod n$

Korrektheit von RSA

Verschlüsselung: $c = m^e \bmod n$, Entschlüsselung: $c^d \bmod n$

- Warum gilt $m = c^d \bmod n$?
- Es gilt $e \cdot d \bmod \phi(n) = 1$

$$
\begin{aligned}
c^d &= (m^e \bmod n)^d \\
&= (m^e)^d \bmod n \\
&= m^{e \cdot d} \bmod n \\
&= m^{k \cdot \phi(n)+1} \bmod n \\
&= m^{k \cdot \phi(n)} \cdot m \bmod n \\
&= (m^{\phi(n)})^k \cdot m \bmod n \\
&= 1^k \cdot m \bmod n \\
&= m \bmod n
\end{aligned}
$$

RSA Beispiel

Schlüsselgenerierung:

- Primzahlen: $p = 11, q = 13$
- Modul: $n = p \cdot q = 143$, Eulerzahl: $\phi(n) = (p-1) \cdot (q-1) = 120$
- Verschlüsselungsschlüssel: $e = 23$ (ist teilerfremd zu $\phi(n)$)
- Entschlüsselungsschlüssel: $23 \cdot d - k \cdot 120 = 1 \Rightarrow d = 47$ (es gilt $e \cdot d = 1081 = 1 \bmod 120$)

Verschlüsselung der Nachricht $m = 7$

- $c = m^e \bmod n = 7^{23} \bmod 143 = 2$

Entschlüsselung des Ciphertextes $c = 2$

- $c^d \bmod n = 2^{47} \bmod 143 = 7$

In Wirklichkeit werden Primzahlen mit Größe 1000 Bit verwendet!

Sicherheit RSA I

- Angreifer kennt
  - den öffentlichen Verschlüsselungsschlüssel e und
  - das Modul n (also die Menge $\mathbb{Z}_n$, in der gerechnet wird)
- Ziel des Angreifers: Bestimme den Klartext m zum Ciphertext $c = m^e \bmod n$
- Einziger Angriff heute: Bestimme den geheimen Entschlüsselungsschlüssel d
  - Zusammenhang zwischen e und d: es gilt $e \cdot d \bmod \phi(n) = 1$
    Erinnerung: Für $n = p \cdot q$, p, q Primzahlen, gilt $\phi(n) = (p-1) \cdot (q-1)$
- Aber: Der Angreifer kennt $\phi(n)$ nicht, kann also d nicht aus e berechnen
- Um $\phi(n)$ zu bestimmen, kann er versuchen, p, q zu berechnen (n faktorisieren)

Sicherheit RSA II

Faktorisierungsproblem:
Gegeben: Zusammengesetzte natürliche Zahl $n = p \cdot q \in \mathbb{N}$, p, q Primzahlen.
Lösung: Finde die beiden Primfaktoren p und q.

- Erste Idee: Probedivision:

Sicherheit RSA II

Faktorisierungsproblem:

Gegeben: Zusammengesetzte natürliche Zahl $n = p \cdot q \in \mathbb{N}$, p, q Primzahlen.

Lösung: Finde die beiden Primfaktoren p und q.

- Erste Idee: Probedivision:
  - for $i = 2$ to $\sqrt{n}$
  - if $n/i \in \mathbb{N}$ stopp
  - return $i, n/i$

Sicherheit RSA II

Faktorisierungsproblem:
Gegeben: Zusammengesetzte natürliche Zahl $n = p \cdot q \in \mathbb{N}$, p, q Primzahlen.
Lösung: Finde die beiden Primfaktoren p und q.

- Erste Idee: Probedivision:
  - for $i = 2$ to $\sqrt{n}$
  - if $n/i \in \mathbb{N}$ stopp
  - return $i, n/i$
- Aber: Es gibt bessere Algorithmen zum Faktorisieren:
  - Für Sicherheitsniveau 100 Bit werden Primzahlen der Größe ca. 2^{1000} benötigt
  - Nachweis in der Vorlesung Kryptologie

Asymmetrische Verschlüsselungsverfahren

Weitere bekannte Verfahren

- Rabin-Verfahren (basiert auch auf Schwierigkeit des Faktorisierungsproblems)
- ElGamal-Verfahren (basiert auf Schwierigkeit des Logarithmusproblems)
- Merkle-Hellman und McEliece (basieren auf Schwierigkeit anderer Probleme)

Nachteil von asymmetrischen Verschlüsselungsverfahren:

Asymmetrische Verschlüsselungsverfahren

Weitere bekannte Verfahren

- Rabin-Verfahren (basiert auch auf Schwierigkeit des Faktorisierungsproblems)
- ElGamal-Verfahren (basiert auf Schwierigkeit des Logarithmusproblems)
- Merkle-Hellman und McEliece (basieren auf Schwierigkeit anderer Probleme)

Nachteil von asymmetrischen Verschlüsselungsverfahren:

- Sind deutlich ineffizienter als symmetrische Verschlüsselungsverfahren
- Verwendung daher in der Praxis nur für den Austausch symmetrischer Schlüssel

Hybridverfahren

Kombination von asymmetrischen (für Schlüsselaustausch) und symmetrischen (für die eigentliche Verschlüsselung) Verschlüsselungsverfahren

- A will B eine vertrauliche Nachricht m übermitteln
- B besitzt ein Schlüsselpaar (pk, sk) (z.B. für das RSA-Verfahren)
- A vertraut dem öffentlichen Schlüssel pk (weiß, dass dieser B gehört)

$\boxed{A}$ öffentl. Schlüssel: $pk = e$		$\boxed{B}$ geheimer Schlüssel: $sk = d$
choose random key k compute $k' = k^e \bmod n$ and $c = \text{enc}(k, m)$	$\xrightarrow{k', c}$	compute $k = k'^d \bmod n$ and $m = \text{dec}(k, c)$

Signaturverfahren

Schutzziel: Nichtabstreitbarkeit
(B kann auch gegenüber Dritten nachweisen, dass die Nachricht von A stammt)

$\boxed{A}$ geheimer Schlüssel: sk	$\boxed{B}$ öffentlicher Schlüssel: pk
compute Signatur $s := \text{sig}(sk, m)$	
$\xrightarrow{m,s}$	
	compute $b := \text{verify}(pk, m, s)$ if $b = \text{true}$ then accept else reject

Wir benötigen Verfahren:

- sig: Signieren von m mit einem geheimen Schlüssel sk
- verify: Prüfen von Signaturen mit dem öffentl. Schlüssel pk

Signaturverfahren RSA

Schlüsselgenerierung:

- Wähle zwei Primzahlen: p, q, setze $n = p \cdot q$ und $\phi(n) = (p-1) \cdot (q-1)$
- Verifikationsschlüssel (pk): Wähle e teilerfremd zu $\phi(n)$
- Signaturschlüssel (sk): Wähle d so, dass $e \cdot d = 1 \bmod \phi(n)$
- Zusätzlich benötigen wir eine Hashfunktion H

Signaturverfahren RSA

Schlüsselgenerierung:

- Wähle zwei Primzahlen: p, q, setze $n = p \cdot q$ und $\phi(n) = (p-1) \cdot (q-1)$
- Verifikationsschlüssel (pk): Wähle e teilerfremd zu $\phi(n)$
- Signaturschlüssel (sk): Wähle d so, dass $e \cdot d = 1 \bmod \phi(n)$
- Zusätzlich benötigen wir eine Hashfunktion H

Signatur einer Nachricht $m \in \mathbb{Z}_n$ (Signaturfunktion sig)

- Berechne $h = H(m)$
- Berechne $s = h^d \bmod n$ (Verschl. von $h = H(m)$ mit dem geheimen Schlüssel)

Signaturverfahren RSA

Schlüsselgenerierung:

- Wähle zwei Primzahlen: p, q, setze $n = p \cdot q$ und $\phi(n) = (p-1) \cdot (q-1)$
- Verifikationsschlüssel (pk): Wähle e teilerfremd zu $\phi(n)$
- Signaturschlüssel (sk): Wähle d so, dass $e \cdot d = 1 \bmod \phi(n)$
- Zusätzlich benötigen wir eine Hashfunktion H

Signatur einer Nachricht $m \in \mathbb{Z}_n$ (Signaturfunktion sig)

- Berechne $h = H(m)$
- Berechne $s = h^d \bmod n$ (Verschl. von $h = H(m)$ mit dem geheimen Schlüssel)

Signaturverifikation (Verifikationsfunktion verify)

- Berechne $h = H(m)$
- Berechne $v = s^e \bmod n$ (Entschl. von s mit öffentl. Schlüssel)
- Wenn $h = v$, akzeptiere die Signatur, sonst nicht

Signaturverfahren

Signaturverfahren erfüllen auch das Schutzziel Nichtabstreitbarkeit

- Die Signatur wird mit dem geheimen Schlüssel erzeugt
- Also kann nur der Inhaber des geheimen Schlüssels die Signatur berechnet haben

Weitere Signaturverfahren

- Digital Signature Algorithm (DSA), El-Gamal, Schnorr-Signatur:
  basieren auf dem diskreten Logarithmusproblem
- Merkle-Signatur: basiert auf Hashbäume
- McEliece-Niederreiter-Signatur: basiert auf lineare Codes

Challenge-Response Verfahren (asymmetrisch)

Ziel: Nachweis eines Geheimnisses (hier Schlüssel sk), ohne dieses offen zu legen
Als Basis können auch Signaturverfahren dienen
(ein symmetrisches Verfahren haben wir bereits gesehen)

$\boxed{A}$ Bs öff. Schlüssel: pk		$\boxed{B}$ Schlüsselpaar: (pk, sk)
choose random c		
(challenge)	$\xrightarrow{c}$	
		compute $r := \text{sig}(sk, c)$
	$\xleftarrow{r}$	(response)
compute $b := \text{verify}(pk, c, r)$		
if $b =$ true then accept		
else reject		

Zusammenfassung RSA

Ein RSA-Schlüsselpaar kann genutzt werden für

- Verschlüsselung
- Datenauthentisierung (Signaturverfahren)
- Instanzauthentisierung (Challenge-Response-Verfahren)

Für alle drei Verfahren müssen verschiedene Schlüssel eingesetzt werden

- Wichtiges kryptographisches Grundprinzip: Trenne wo du trennen kannst
- Es gib auch Angriffe, wenn die selben Schlüssel genutzt werden (nächste Folie)

Schlüsseltrennung

Schlüssel für Daten- und Instanzauthentisierung müssen verschieden sein

Andernfalls ist folgender Angriff möglich:

- Angreifer erzeugt Nachricht m und bildet $H(m) = c$ (challenge)
- Angreifer fordert B zur Authentisierung auf und sendet c
- B berechnet Prüfsumme r
  (B sieht nicht, ob c Zufall oder der Hashwert einer Nachricht ist)
- Dem Angreifer liegt eine von B korrekt signierte Nachricht vor

Schlüsseleinigungsverfahren

Ziel: Sicherer Austausch von Schlüsseln über einen unsicheren Kanal

- Diffie-Hellman-Schlüsseleinigungsverfahren
  (Entwickelt von Diffie, Hellman und Merkle 1976)
- Bekannt seit 1997: Ähnliches Verfahren entwickelt Anfang 1970 vom britischen Geheimdienst (Ellis, Cocks, Williamson)
- Sicherheit beruht auf der Schwierigkeit des Diskreten Logarithmusproblems

Problem Diskreter Logarithmus (DL):

$g^x \mod p = h$

Bei gegebenen g, h und p ist es schwierig x zu berechnen!

Beispiel: $2^x \mod 11 = 5$

Effizientes Exponieren

Problem Diskreter Logarithmus (DL):

$g^x \mod p = h$

Exponentiation kann effizient ohne große Zwischenergebnisse durchgeführt werden!

Beispiel:

$$g^8 \mod p = g \cdot g \cdot g \cdot g \cdot g \cdot g \cdot g \cdot g \mod p$$

Effizientes Exponieren

Problem Diskreter Logarithmus (DL):
$g^x \mod p = h$

Exponentiation kann effizient ohne große Zwischenergebnisse durchgeführt werden!

Beispiel:

$$g^8 \mod p = g \cdot g \cdot g \cdot g \cdot g \cdot g \cdot g \cdot g \mod p$$

Besser:

$$g^8 \mod p = ((g^2)^2)^2 \mod p$$

$g^x \mod p$, wenn $x \neq 2^n$, Beispiel: $x = 25 = 2^4 + 2^3 + 2^0$

$$g^{25} \mod p = (g \cdot g^{24}) \mod p = (g \cdot g^8 \cdot g^{16}) \mod p$$
$$= (g \cdot ((g)^2)^2)^2 \cdot (((g^2)^2)^2)^2) \mod p = ((((g^2 \cdot g)^2)^2)^2 \cdot g) \mod p$$

Bei Speicherung der Zwischenergebnisse nur sechs Multiplikationen!

Mathematische Grundlagen I

Wir betrachten wieder $\mathbb{Z}_p := \{0, 1, 2, \ldots, p-1\}$

Eigenschaften:

- Ist p eine Primzahl, so sind alle $x \in \mathbb{Z}_p \backslash \{0\}$ teilerfremd zu p (Def. Primzahl)
- Also besitzen alle $x \in \mathbb{Z}_p \backslash \{0\}$ multiplikativ Inverse in $\mathbb{Z}_p$
- Sei $\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\} = \{1, \ldots, p-1\}$
  (Menge aller multiplikativ invertierbaren Elemente in $\mathbb{Z}_p$)

Mathematische Grundlagen II

Eigenschaften (Fortsetzung):

- $\mathbb{Z}_p^*$ ist zyklisch, d.h. es gibt $g \in \mathbb{Z}_p^*$ mit $\{g^n \bmod p; n \in \mathbb{N}\} = \{1, 2, \ldots, p-1\}$
  - g heißt dann Erzeuger der Gruppe (Primitivwurzel)
  - Insbesondere gilt also $\mathbb{Z}_p^* = \{g^n \bmod p; 1 \leq n \leq p-1\}$
- Beispiel $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$:
  - 3 ist Erzeuger: $3^1 = 3 \bmod 5, 3^2 = 9 = 4 \bmod 5$, $3^3 = 27 = 2 \bmod 5, 3^4 = 81 = 1 \bmod 5$.
  - 4 nicht: $4^1 = 4, 4^2 = 16 = 1 \bmod 5, 4^3 = 64 = 4 \bmod 5, \ldots$ Also $\{4^n; n \in \mathbb{N}\} = \{1, 4\}$.
- Ist $g \in \mathbb{Z}_p^*$ Erzeuger, dann existiert für alle $h \in \mathbb{Z}_p^*$ ein $x \in \mathbb{N}$ mit $g^x = h.$, d.h. $\log_g h$ ist lösbar $(= x)$
- $\log_4 3$ ist nicht lösbar in $\mathbb{Z}_5^*$

Diffie-Hellman-Schlüsseleinigungsverfahren I

- Wir rechnen in $\mathbb{Z}_p^*$, p sehr große Primzahl ($\simeq$ 2000 bit)
- Wir benötigen einen Erzeuger $g \in \mathbb{Z}_p^*$
  (es gibt effiziente Verfahren diese zu finden)

A Parameter (g, p)		B Parameter (g, p)
choose random x		
compute $X = g^x \bmod p$	$\xrightarrow{X}$	
		choose random y
	$\xleftarrow{Y}$	compute $Y = g^y \bmod p$
compute $Y^x = (g^y)^x = g^{xy} \bmod p$		compute $X^y = (g^x)^y = g^{xy} \bmod p$

Diffie-Hellman-Schlüsseleinigungsverfahren II

Aus g^{xy} lassen sich z.B. Schlüssel für Secure Messaging ableiten:

- $k_E := \text{Hash}(g^{xy}||0x00)$ Schlüssel für Verschlüsselung
- $k_A := \text{Hash}(g^{xy}||0x01)$ Schlüssel für MAC

Sicherheit des Diffie-Hellman-Schlüsseleinigungsverfahrens

Angreifer kennt Parameter (g, p) und sieht $g^x \bmod p$ und $g^y \bmod p$
Ziel: Bestimmung von g^{xy} (das gemeinsame Geheimnis)

- Einzige derzeit bekannte Möglichkeit: Bestimme x oder y (d.h. berechne $\log_g g^x$ oder $\log_g g^y$)
- Erste Idee hierfür: Probepotenzieren
  - Wähle x', berechne $g^{x'}$ und vergleiche: $g^x \stackrel{?}{=} g^{x'}$.
  - Für $p > 2^{100}$ ist $|\mathbb{Z}_p^*| > 2^{100}$, also W'keit für Gleichheit: $1/2^{100}$.
  - D.h. ca. 2^{100} Versuche, x zu finden (Sicherheitsniveau 100 Bit)
- Aber: Es gibt bessere Algorithmen zum Bestimmen des Logarithmus
  - Für Sicherheitsniveau von 100 Bit werden deutlich größere Zahlen benötigt
  - Nachweis in der Vorlesung Kryptologie

Man-in-the-Middle Angriff

Aber folgender Angriff möglich:

A		E (Angreifer)		B
choose random x		choose random z		
$X = g^x \bmod p$	$\xrightarrow{X}$	$Z = g^z \bmod p$	$\xrightarrow{Z}$	choose random y
	$\xleftarrow{Z}$		$\xleftarrow{Y}$	$Y = g^y \bmod p$
$Z^x = g^{xz} \bmod p$		$X^z = g^{xz} \bmod p$		
		$Y^z = g^{yz} \bmod p$		$Z^y = g^{yz} \bmod p$

- Nicht A und B einigen sich auf ein Geheimnis, sondern A mit E und B mit E
- Nachfolgend kann E durch Umschlüsselung die Kommunikation zwischen A und B belauschen, ohne das dies auffällt

Diffie-Hellman-Schlüsseleinigungsverfahrens

Die Schlüsselanteile $g^x \bmod p$ und $g^y \bmod p$ müssen den jeweiligen Kommuniaktionspartnern zugeordnet werden können

- Mögliche Lösungen sind:
  - Authentisierung der Schlüsselanteile (MAC oder Signatur)
  - Authentisierung der Kommunikationspartner (Instanzauthentisierung)
- In jedem Fall benötigen A und B vorab einen Vertrauensanker, z.B.:
  - öffentlicher Schlüssel des jeweils anderen
  - gemeinsamer symmetrischer Schlüssel
- Dieser Vertrauensanker muss vorab sicher ausgetauscht werden:
  - öffentliche Schlüssel: authentisch
  - symmetrischer Schlüssel: vertraulich und authentisch

Vertrauensmodelle I

Problem: Zuordnung eines Schlüssels zum Schlüsselinhaber (wem gehört der Schlüssel)

- Beispiel 1: Verschlüsselung mit öffentlichem Schlüssel, Entschlüsselung mit geheimem Schlüssel:
  - Bei Nutzung des öffentlichen Schlüssels einer falschen Person kann diese die Nachricht entschlüsseln (Verlust der Vertraulichkeit)
- Beispiel 2: Signaturerzeugung mit geheimem Schlüssel, Verifikation mit öffentlichem Schlüssel:
  - Bei falscher Zuordnung des öffentlichen Schlüssels zu einer Person vertraue ich fälschlicherweise auf die Authentizität der Daten (Verlust der Datenauthentizität)

Vertrauensmodelle II

Wir unterscheiden drei Vertrauensmodelle

- Direct Trust: Nutzer erhält den öff. Schlüssel direkt vom Schlüsselinhaber
  - kleine Virtual Private Networks
- Web of Trust: Nutzer signieren gegenseitig ihre öff. Schlüssel
  - PGP, GNU-PG (Key Signing Parties)
- Hierarchical Trust: Öff. Schüssel werden von einer zentralen Instanz verwaltet
  - Qualifizierte elektronische Zertifikate nach Signaturgesetz
  - Public Key Infrastruktur des Deutschen Forschungsnetzwerkes
  - Server-Authentisierung via https

Direct Trust

Direct Trust ist nur für kleine Gruppen anwendbar (Paarweiser Schlüsselaustausch notwendig)

- 2 Personen: ein Austausch
- 3 Personen: 3
- 4 Personen: 6

Direct Trust

Direct Trust ist nur für kleine Gruppen anwendbar (Paarweiser Schlüsselaustausch notwendig)

- 2 Personen: ein Austausch
- 3 Personen: 3
- 4 Personen: 6
- n Personen: $n(n-1)/2$
- Kommt neuer Partner hinzu: Austausch mit n Personen

Web of Trust I

Idee: Nutzer signieren öffentliche Schlüssel andere Nutzer (und garantieren so für die Authentizität des Schlüssels).

- Beispiel (Alice, Bob und Carl):
  - Alice signiert Bobs öffentlichen Schlüssel
  - Bob signiert Carls öffentlichen Schlüssel
  - Alice vertraut Carls öffentlichem Schlüssel
- Warum vertraut Alice Carls Schlüssel?

Web of Trust I

Idee: Nutzer signieren öffentliche Schlüssel andere Nutzer (und garantieren so für die Authentizität des Schlüssels).

- Beispiel (Alice, Bob und Carl):
  - Alice signiert Bobs öffentlichen Schlüssel
  - Bob signiert Carls öffentlichen Schlüssel
  - Alice vertraut Carls öffentlichem Schlüssel
- Warum vertraut Alice Carls Schlüssel?
  - Alice prüft mit Bobs öff. Schlüssel Bobs Signatur von Carls öff. Schlüssel
    (Ist aber nur dann sicher, wenn Alice Bob vertraut, nur authentische Schlüssel zu signieren, also vor der Signierung die Bindung zwischen Schlüssel und Schlüsselinhaber zu prüfen (z.B. über Direct Trust))

Web of Trust II

Jeder Nutzer hat einen Schlüsselbund (key ring) mit öff. Schlüsseln anderer Nutzer

- Jedem öffentlichen Schlüssel sind zugeordnet
  - Name des Schlüsselinhabers
  - Owner Trust (5 Stufen, Grad des Vertrauens in die Prüffähigkeit des Schlüsselinhabers)
  - Key Legitimacy (Zulässigkeit) (3 Stufen, Grad des Vertrauens in den öffentlichen Schlüssel)
    leitet sich ab aus den Owner Trusts der Signierer und der Anzahl der Signaturen
  - Signaturen des öffentlichen Schlüssels
- Vorgehen in RFC 2440 (OpenPGP Message Format) beschrieben

Web of Trust: Owner Trust

- Wert für Owner Trust (OT) legt jeder Nutzer selbst fest.
- Gibt an, wie der Schlüsselinhaber öffentliche Schlüssel anderer Nutzer prüft
- Fünf Level:
  - unbekannt (unknown) für Nutzer, über die man keine weiteren Informationen hat
  - kein Vertrauen (not trusted) für Nutzer, denen nicht vertraut wird
  - geringes Vertrauen (marginal) für Nutzer, denen nicht voll vertraut wird
  - volles Vertrauen (complete) für Nutzer, denen voll vertraut wird
  - absolutes Vertrauen (ultimate) für Nutzer, deren privater Schlüssel sich im privaten Schlüsselbund befindet (üblicherweise nur die eigenen privaten Schlüssel)

Web of Trust: Key Legitimacy

Vertrauen in Authentizität eines öffentlichen Schlüssels

- $x =$ Anz. Signaturen von Nutzern, deren OT marginal ist
- $X =$ Anz. Signaturen mit einem OT marginal, die erforderlich ist, damit ein Schlüssel als authentisch eingestuft wird
- $y =$ Anz. Signaturen von Nutzern, deren OT complete ist
- $Y =$ Anz. Signaturen mit einem OT complete, die erforderlich ist, damit ein Schlüssel als authentisch eingestuft wird
- Vertrauenslevel $L := x/X + y/Y$. Man unterscheidet 3 Level:
  - nicht authentisch $(L = 0)$, teilweise authentisch $(0 < L < 1)$, authentisch $(L \geq 1)$
- Üblich sind die Werte $X = 2$ und $Y = 1$, d.h., Schlüssel ist authentisch, wenn
  - der Schlüssel von einer vertrauenswürdigen (complete) Person, oder
  - von zwei teilweise vertrauenswürdigen (marginal) Personen signiert wurde

Web of Trust: Eigenschaften

Nachteile:

- Einstufung (Owner Trust) erfordert hohes Wissen der Nutzer
- Die Signaturen sind juristisch nicht bindend (vgl. Signaturgesetz)
- Zurückziehen von Zertifikaten nicht einfach umsetzbar

Vorteile:

- Gegenüber Direct Trust eine deutliche Verbesserung
- Umgesetzt in PGP (Pretty Good Privacy) und GnuPG (Open Source)
- Es gibt zahlreiche Schlüsselserver, auf denen öffentliche Schlüssel und zugehörige Signaturen hochgeladen werden können
- Einige Institutionen bieten einen Certification Service für PGP- und GPG-Schlüssel an

Hierarchical Trust I

Ziele:

- Zuordnung eines öffentlichen Schlüssels zum Schlüsselinhaber
- Festlegung der Schlüsselnutzung (Verschlüsselung, Authentisierung, Signatur)
- Etablierung einer gemeinsamen Sicherheitsinfrastruktur
  - Wie sicher sind die Schlüssel gelagert, erzeugt usw.
  - Wie stark ist die Bindung zwischen Schlüssel und Schlüsselinhaber
    (z.B. wie wird geprüft)

Hierarchical Trust II

Public Key Infrastrukturen:

- Öffentliche Schlüssel werden von einer vertrauenswürdigen Instanz verwaltet
- Nutzer erhalten Zertifikate C_i für ihren öffentlichen Schlüssel pk_i
- Vertrauensanker bildet der öffentliche Schlüssel pk_{CA} der CA

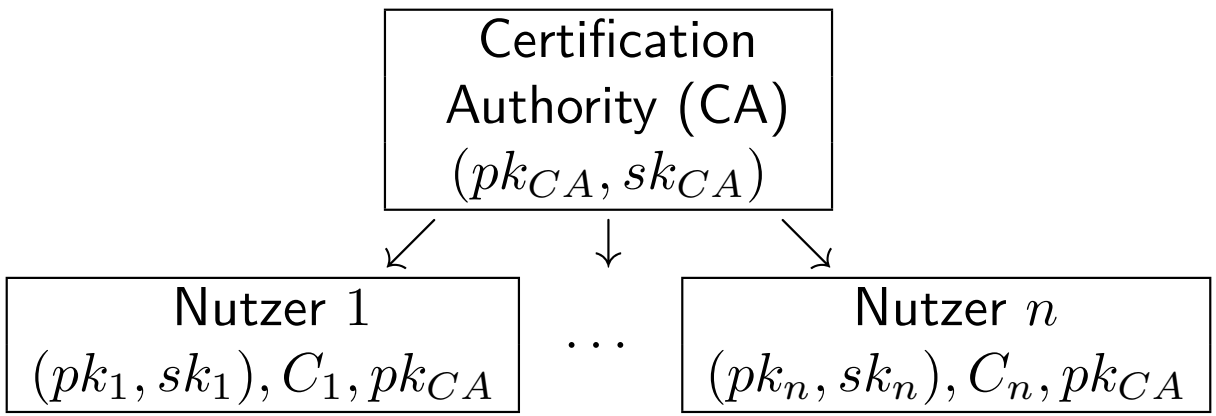


Abbildung: Schematischer Aufbau einer Public Key Infrastruktur

Public Key Infrastrukturen

$C_i =$

Angaben zum Nutzer Name, Organisation, usw.
öffentlicher Schlüssel pk_i
Verwendeter Algorithmus z.B. RSA, DSA
Gültigkeitszeitraum
Angaben zur Certification Authority Name, Kontaktdaten usw.
Schlüsselnutzung z.B. Signatur, Authentisierung, Verschlüsselung
Signatur der CA (mit sk_{CA})

Abbildung: Zertifikat für Nutzer i (ausgestellt von der Certification Authority)

Beispiel Zertifikat

Ausgestellt für

Allgemeiner Name (CN)	www.h-da.de
Organisation (O)	Hochschule Darmstadt
Organisationseinheit (OU)	IT Dienste und Anwendungen
Seriennummer	1F:C1:79:5A:DD:21:E3:A7:60:B6:D8:18

Ausgestellt von

Allgemeiner Name (CN)	DFN-Verein Global Issuing CA
Organisation (O)	Verein zur Foerderung eines Deutschen Forschungsnetzes e. V.
Organisationseinheit (OU)	DFN-PKI

Gültigkeitsdauer

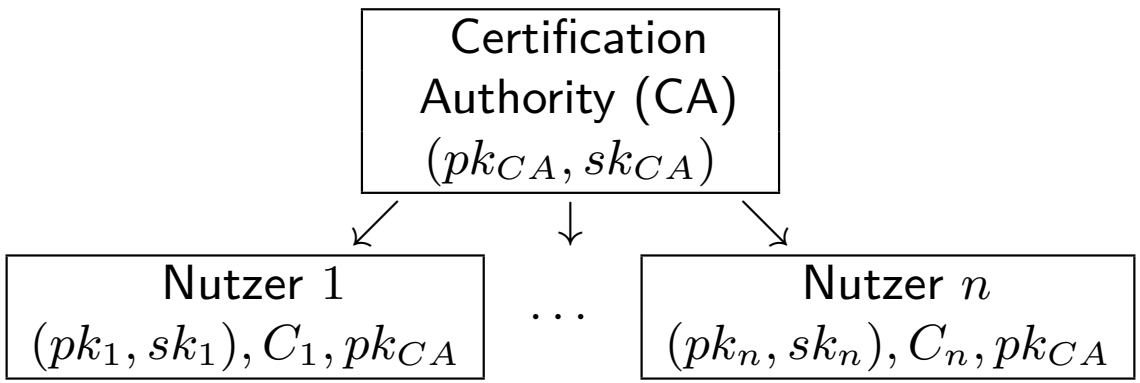
Beginnt mit	Mittwoch, 19. September 2018
Gültig bis	Montag, 21. Dezember 2020

Fingerabdrücke

SHA-256-Fingerabdruck	84:44:AB:22:3B:62:43:80:46:FC:96:7C:A0:20:07:11: 7E:CB:4D:6F:C8:25:50:C2:5A:36:B9:3B:38:C9:CD:AF
SHA1-Fingerabdruck	AA:33:D4:56:B2:79:FE:CF:5B:29:17:3F:29:34:07:29:04:F1:C5:1B

PKCS #1 RSA-Verschlüsselung
Modulus (4096 Bits):
d3 cb 5f 74 38 34 10 35 09 ff e1 fc a6 2c e3 98 f6 58 23 f0 6b f3 ae e8 da ad b9 b6 43 83 0d 24 09 cf 09 6b 40 04 bb 59 f3 9f 40
9e 1f 8f 36 11 2f 44 a0 00 2d 7c b8 a2 10 68 be c6 5d d1 5e 89 ae f6 c4 06 2a fe 6d c7 9d c6 64 cc af 52 82 a1 cb f7 4e b4 21 32
4e ff 48 7f 6a f6 dc e9 ad db 89 23 d6 eb d6 82 c0 df 10 90 ef 4b ae db 2e 54 7d 64 52 09 50 79 55 1c 34 16 d2 59 87 ec c3 af 0a
06 8d fa 0d af 06 f1 37 67 a3 87 62 d0 13 26 38 9e 94 1f 28 b7 34 2c 5a 04 7d a8 99 39 b8 95 8b 24 46 e3 d7 9f e0 4e bf 2c a3 19
28 e5 75 3f ba f7 b8 dc 0a dd 87 f8 8a e5 cb ef f5 cf cb 61 83 d2 3c 38 54 16 fa 53 a4 90 45 66 ad f3 8e e8 64 eb 49 92 3a 5d 9d
b9 c1 35 e0 d8 3b 9e 91 16 ed 8b a5 23 e2 ad 47 d7 5b b6 0c 6d a8 b5 f3 25 c1 f2 9f af 0c f9 78 44 9c 30 8e 1a cb ff 75 c7 51 0e
39 95 5b a5 98 63 55 a8 4a f0 ab b3 d8 e1 b2 f1 dd 2f 16 83 df 59 c4 2f 80 bc d1 af fc a3 b3 84 08 10 d3 18 72 1a 9a 0e 51 93 85
e3 00 51 7f 15 eb 60 83 0b f5 0f d6 f2 94 f0 e9 d6 74 96 b3 ea 15 2e 40 7d 7d df a7 55 c6 07 5c 48 0a d1 a9 ed 31 40 fe 2f c9 9c
62 21 c7 bf 15 74 c9 f3 74 85 e1 2c 40 e1 25 14 6b 3e b7 0b 96 a3 1e 49 1b 47 9f b1 cb ae cf 26 4f c3 09 41 fa c3 77 89 53 5b c7
6a e5 b9 91 29 6f 0a c7 14 0a 98 d8 fe 55 dc 85 2a 4e 52 3e 04 77 c7 76 eb 4b 64 92 e7 8d b2 76 25 81 8f f8 3a e6 af c9 3f b9 a8
ab 9d 97 90 5e d7 34 fb e7 9c 4c 63 18 13 1e 71 f6 9c 27 8f 7d 5d f0 ad de 4c 7e cc 0a e2 53 57 10 09 42 7d 0b 50 5a 5a 79 87 51
8e c6 f7 b0 ef b9 e7 0a 66 f5 a2 c0 8d e2 bd 03 80 c1 52 76 c9 88 9d f7 67 be 8d 25 14 90 fd 0b 57 22 36 b1 88 98 9b
Exponent (24 Bits):
65537
Signatur:
Größe: 256 Bytes / 2048 Bits
63 7a 5f 99 6a 64 91 1f 1b 9a a0 aa e2 1d 0b ba 9b 34 e2 14 17 46 58 f7 bc bd fd 56 50 05 7d 6f c8 42 dd 7d bb eb 3c db 59 7e
41 de c8 18 fd 41 6e 04 b7 b0 6e 93 df 2e 65 ae f4 59 a1 d1 79 e7 3c 11 a2 e1 97 9c 68 0f a2 6d 46 9d 3d 62 aa 97 f6 c6 77 5b 0e
0b 0c 49 5e 45 18 e1 ae 92 99 c3 bd e4 f0 c1 6e cc 23 4e cb be 8f f9 81 67 28 4c fc 85 0a 88 b0 3f 21 c4 e2 9d 0e 9e b7 35 d8 1e
f9 ba 92 6c 32 20 d6 23 1b 00 87 52 8a f3 57 85 e3 04 0c ec 66 0c 9c 04 2c 03 84 cd 2b 02 0a a5 05 32 79 01 5c 52 fa f6 5e 53 9d
b8 ee 27 f9 44 33 3f bb 2a cd 92 6c 0a 08 98 54 e4 f8 ec bc d6 4d 65 06 16 4e 49 29 aa ee f2 76 1d 4b 64 81 5a 53 86 d6 aa 62 9b
cb d4 57 37 35 58 5c 7f 6e 71 95 ae 00 a7 75 22 6d 22 36 00 d7 ec d3 46 a4 96 76 8a 58 df 8e f7 07 f0 33 91 52 d9 e6 8d 3a 8e

Public Key Infrastrukturen: Beispiel Signatur



Nutzer 1 signiert Dokument m, Nutzer 2 prüft Signatur

Nutzer 1	Schlüssel:	Nutzer 2	Schlüssel: pk_{CA}
$(pk_1, sk_1), C_1$			
compute $s := \text{sig}(sk_1, m)$	$\xrightarrow{s, C_1, m}$	verify(pk_{CA}, C_1) verify(pk_1, m, s)	

Public Key Infrastrukturen

Häufig weitere Aufteilung:

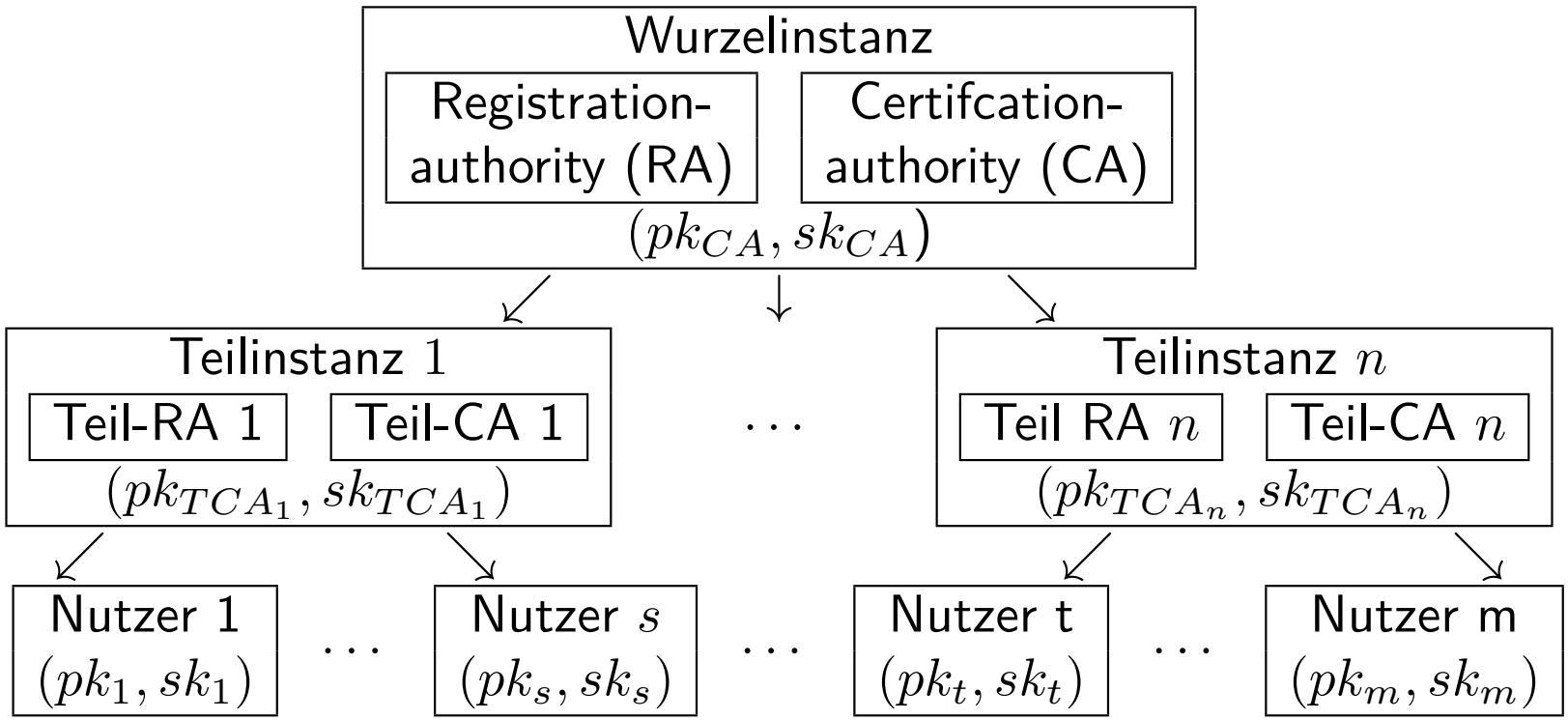


Abbildung: Schematische Darstellung einer 3-stufigen Public Key Infrastruktur

Public Key Infrastrukturen

Aufgabenverteilung:

- Registration Authority
  - Legt Sicherheitsrichtlinien fest
  - Prüft Zertifizierungsanträge
  - Legt Inhalte der Zertifikate fest
  - Leitet Anträge an Certification Authority weiter
- Certification Authority:
  - Stellt die eigentlichen Zertifikate aus

Zurückziehen von Zertifikaten:

- Bei Sicherheitsvorfällen (z.B. Schlüsselkompromittierung)
- Eine Instanz hält sich nicht an Sicherheitsvorgaben
- Hierzu: Führen von Certificate Revocation Lists (CRLs)
- Zusätzlich: Prüfung, ob Zertifikate in CRL enthalten ist

Public Key Infrastrukturen

Alle Instanzen einer PKI (Root-CA, Teil-CAs, Endnutzer) müssen ein definiertes Maß an Sicherheitsstandards einhalten.
Sicherheitsvorgaben werden in zwei Dokumenten festgelegt:

- Certificate Policy: welche Sicherheitsvorgaben müssen eingehalten werden
- Certificate Practise Statement: wie werden die Sicherheitsvorgaben umgesetzt
- RFC 3647: Internet X.509 Public Key Infrastructure Certificate Policy an Certificate Practise Framework
  (beschreibt wird Aufbau und Inhalt beider Dokumente im Detail)