Kryptographie I
Symmetrische Kryptographie

Alexander May

Fakultät für Mathematik
Ruhr-Universität Bochum

Wintersemester 2009/10

Organisatorisches

- Vorlesung: **Mo 12-14** in HZO 80 (2+2 SWS, 6 CP)
- Übung: **Mo 16-18** und **Mi 10-12** in NA 5/99
- Assistent: **Thomas Dullien**, Korrektor: **Florian Giesen**
- Übungsbetrieb: jeweils abwechselnd alle 2 Wochen
  - Präsenzübung, Start 19. Oktober
  - Zentralübung, Start 26. Oktober
- Übungsaufgaben werden korrigiert.
- Gruppenabgaben bis 3 Personen
- Bonussystem:
  1/3-Notenstufe für 50%, 2/3-Notenstufe für 75%
- Klausur: Ende Februar

Literatur

Vorlesung richtet sich nach

- Jonathan Katz, Yehuda Lindell, “Introduction to Modern Cryptography”, Taylor & Francis, 2008

Weitere Literatur

- S. Goldwasser, M. Bellare, “Lecture Notes on Cryptography”, MIT, online, 1996–2008
- O. Goldreich, “Foundations of Cryptography – Volume 1 (Basic Tools)”, Cambridge University Press, 2001
- O. Goldreich, “Foundations of Cryptography – Volume 2 (Basic Applications)”, Cambridge University Press, 2004s
- A.J. Menezes, P.C. van Oorschot und S.A.Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1996

Effiziente Algorithmen

Ziel:

- Ver-/Entschlüsseln soll effizient möglich sein.
- Unser Berechnungsmodell ist die Turingmaschine (s. DiMa I+II)
- Verwenden polynomielle Algorithmen $A \in \mathcal{P}$.

Definition Polynomialzeit-Algorithmus

Sei A ein Algorithmus. A heißt *polynomial-Zeit* (*pt*), falls A bei Eingaben der Länge n in Laufzeit $\mathcal{O}(n^k)$ für ein festes k anhält.
A heißt *probabilistisch polynomial-Zeit* (*ppt*), falls A ein pt-Algorithmus ist, der Zufallsbits verwendet.

Verschlüsselungsverfahren

Definition Symmetrisches Verschlüsselungsverfahren

Sei n ein Sicherheitsparameter und $\mathcal{K}, \mathcal{M}, \mathcal{C}$ der Schlüssel-, Nachrichten- bzw. Chiffretextraum.
Ein *symmetrisches Verschlüsselungsverfahren* $\Pi = (Gen, Enc, Dec)$ besteht drei ppt-Algorithmen:

1. **Gen:** *Gen* liefert bei Eingabe 1^n einen Schlüssel $k \in_R \mathcal{K}$.
2. **Enc:** *Enc* liefert bei Eingabe k und Nachricht $m \in \mathcal{M}$ einen Chiffretext $c \in C$. D.h. *Enc* berechnet eine Funktion $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$. Wir schreiben $c \leftarrow Enc_k(m)$.
3. **Dec:** *Dec* liefert bei Eingabe k und $c = Enc_k(m) \in \mathcal{C}$ eine Nachricht mit der Eigenschaft

$$Dec_k(Enc_k(m)) = m \text{ für alle } k \in \mathcal{K}, m \in \mathcal{M}.$$

Wir schreiben $m \leftarrow Dec_k(c)$.

- $Enc_k(m)$ ist für jedes feste k injektiv.

Kerckhoffs' Prinzip (1883)

Forderung Kerckhoffs' Prinzip

Die Sicherheit eines Verschlüsselungsverfahrens $\Pi = (Gen, Enc, Dec)$ darf ausschließlich auf der Geheimhaltung des Schlüssels beruhen. D.h. Gen, Enc und Dec sind bekannt.

Anmerkungen:

- Schlüssel lassen sich besser geheimhalten als Algorithmen.
- Schlüssel lassen sich besser austauschen als Algorithmen.
- Schlüssel lassen sich besser verwalten als Algorithmen.
- Öffentliche Untersuchung von Π durch Experten ist erforderlich.

Typen von Angreifern

Definition Angreiferszenarien

Wir unterscheiden folgende vier Angriffe auf Verschlüsselungsverfahren in aufsteigender Stärke.

1. **Ciphertext Only Angriff (COA, passiver Angriff):**
   Angreifer erhält nur Chiffretexte.
2. **Known Plaintext Angriff (KPA, passiv):**
   Angreifer erhält Paare Klartext/Chiffretext.
3. **Chosen Plaintext Angriff (CPA, aktiv):**
   Angreifer erhält Chiffretexte von Klartexten seiner Wahl.
4. **Chosen Ciphertext Angriff (CCA, aktiv):**
   Angreifer erhält Entschlüsselung von Chiffetexten seiner Wahl.

Ziele eines Angreifers:

- Ziel bei 1: Bestimme zugrundeliegende Klartexte.
- Ziel bei 2-4: Bestimme Klartexte anderer Chiffretexte.
- Wir werden bald schon schwächere Ziele für Angreifer definieren.

Monoalphabetische Substitution – Verschiebechiffre

Idee der Verschiebe-Chiffre: Verschiebe jeden Buchstaben um k Position zyklisch im Alphabet. Identizieren $A, \ldots, Z$ mit $0, \ldots, 25$.

Definition Verschiebe-Chiffre (ca. 50 v. Chr.)

Es gilt $\mathcal{M} = \mathcal{C} = \mathbb{Z}_{26}^n$ und $\mathcal{K} = [25] := \{1, \ldots, 25\}$.

1. **Gen:** Ausgabe $k \in_R [25]$.
2. **Enc:** Verschlüssele $m = m_0 \ldots m_{n-1} \in \mathbb{Z}_{26}^n$ als $c := Enc_k(m_0) \ldots Enc_k(m_{n-1})$ mit
$$Enc_k(m_i) \leftarrow m_i + k \bmod 26 \text{ für } i = 0, \ldots, n-1.$$
3. **Dec:** Entschlüssele $c := c_0 \ldots c_{n-1}$ als $m_0 \ldots m_{n-1} := Dec_k(c_0) \ldots Dec_k(c_{n-1})$ mit
$$Dec_k(c_i) \leftarrow c_i - k \bmod 26 \text{ für } i = 0, \ldots, n-1.$$

- Beispiel: KRYPTO wird mit $k = 2$ als MTARVQ verschlüsselt.
- $|\mathcal{K}| = 25$, d.h. der Schlüsselraum kann leicht durchsucht werden.
- Benötigen Schlüsselräume mit mindestens 2^{80} Elementen.

Polyalphabetische Substitution – Vigenère Chiffre

Idee der Vigenère Chiffre:

- Verwende t hintereinandergeschaltete Verschiebungen.

Definition Vigenère Chiffre (1553)

Es gilt $\mathcal{M} = \mathcal{C} = \mathbb{Z}_{26}^n$ und $\mathcal{K} = \mathbb{Z}_{26}^t$.

1. **Gen:** Berechne $k = k_0 ... k_{t-1} \in_R \mathbb{Z}_{26}^t$.
2. **Enc:** Verschlüssele $m = m_0 \dots m_{n-1} \in \mathbb{Z}_{26}^n$ als $c := Enc_k(m_0) \dots Enc_k(m_{n-1})$ mit
$$Enc_k(m_i) \leftarrow m_i + k_{i \bmod t} \bmod 26 \text{ für } i = 0, \dots, n-1.$$
3. **Dec:** Entschlüssele $c := c_0 \dots c_{n-1}$ als $m_0 \dots m_{n-1} := Dec_k(c_0) \dots Dec_k(c_{n-1})$ mit
$$Dec_k(c_i) \leftarrow c_i - k_{i \bmod t} \bmod 26 \text{ für } i = 0, \dots, n-1.$$

- Sonderfall $t = 1$ liefert Verschiebechiffre.
- Sonderfall $t = n$ liefert perfekt sichere (!) Vernam-Chiffre (1918).
- Kryptanalyse mittels Häufigkeitsanalyse für $t \ll n$ möglich.

Prinzipien der modernen Kryptographie – Sicherheit

Prinzip 1 Sicherheitsziel

Die Sicherheitsziele müssen präzise definiert werden.

Beispiele für ungenügende Definitionen von Sicherheit

- *Kein Angreifer kann k finden.* Betrachte $Enc_k(\cdot)$, das die Identität berechnet: k wird zum Entschlüsseln nicht benötigt.
- *Kein Angreifer kann die zugrundeliegende Nachricht bestimmen.* Möglicherweise können 90% der Nachricht bestimmt werden.
- *Kein Angreifer kann einen Buchstaben des Klartexts bestimmen.* Möglicherweise kann der Angreifer zwischen zwei Nachrichten – z.B. JA und NEIN – unterscheiden.
- *Kein Angreifer erhält Information über den Klartext vom Chiffretext.* Gut, erfordert aber Spezifikation des Begriffs Information.

Alternative Definition: *Kein Angreifer kann für einen Chiffretext eine Funktion auf dem zugrundeliegenden Klartext berechnen.*

Prinzip 2 – Präzisierung der Annahmen

Prinzip 2 Komplexitätsannahme

Es muss spezifiziert werden, unter welchen Annahmen das System als sicher gilt.

Eigenschaften:

- Annahmen sollten unabhängig von der Kryptographie sein. Bsp: Das Faktorisierungsproblem ist nicht in polynomial-Zeit lösbar.
- Angriffsszenario KO, KP, CPA oder CCA muss definiert werden.
- Wir müssen das Berechnungsmodell des Angreifers definieren, z.B. eine Beschränkung auf ppt Angreifer.

Prinzip 3 – Reduktionsbeweis der Sicherheit

Prinzip 3 Beweis der Sicherheit

Wir beweisen, dass unter den gegebenen Annahmen *kein* Angreifer die Sicherheit brechen kann.

Anmerkungen:

- D.h. wir beweisen, dass das System gegen **alle** Angreifer sicher ist, unabhängig von der Herangehensweise des Angreifers!
- Typische Beweisaussage: “Unter Annahme X folgt die Sicherheit von Konstruktion Y bezüglich der Sicherheitsdefinition Z”.
- Der Beweis erfolgt per Reduktion: Ein erfolgreicher Angreifer $\mathcal{A}$ für Y bezüglich Z wird transformiert in einen Algorithmus $\mathcal{B}$, der Annahme X verletzt.

  Bsp: Angreifer $\mathcal{A}$ auf die CCA-Sicherheit einer Verschlüsselung liefert einen Algorithmus $\mathcal{B}$ zum Faktorisieren.

Perfekte Sicherheit

Szenario:

- Angreifer besitzt *unbeschränkte* Berechnungskraft.
- Seien $\mathcal{M}, \mathcal{K}, \mathcal{C}$ versehen mit Ws-Verteilungen.
- Sei M eine Zufallsvariable für die Ws-Verteilung auf $\mathcal{M}$, d.h. wir ziehen ein $m \in \mathcal{M}$ mit $\mathrm{Ws}[M = m]$.
- Analog definieren wir Zufallsvariablen K für $\mathcal{K}$ und C für $\mathcal{C}$.
- Es gelte oBdA $\mathrm{Ws}[M = m] > 0$ und $\mathrm{Ws}[C = c] > 0$ für alle $m \in \mathcal{M}, c \in \mathcal{C}$. (anderenfalls entferne m aus $\mathcal{M}$ bzw. c aus $\mathcal{C}$)

Definition Perfekte Sicherheit

Ein Verschlüsselungsverfahren $\Pi = (Gen, Enc, Dec)$ heißt *perfekt sicher*, falls $\mathrm{Ws}[M = m \mid C = c] = \mathrm{Ws}[M = m]$ für alle $m \in \mathcal{M}$, $c \in \mathcal{C}$.

Interpretation: c liefert dem Angreifer keine Informationen über m.

Verteilung auf Chiffretexten unabhängig vom Plaintext

Satz Chiffretext-Verteilung

Ein Verschlüsselungsverfahren Π ist perfekt sicher gdw $\text{Ws}[C = c \mid M = m] = \text{Ws}[C = c]$ für alle $m \in \mathcal{M}, c \in \mathcal{C}$.

Beweis:

- "$\Rightarrow$": Sei Π perfekt sicher. Nach dem Satz von Bayes gilt

$$\frac{\text{Ws}[C = c \mid M = m] \cdot \text{Ws}[M = m]}{\text{Ws}[C = c]} = \text{Ws}[M = m \mid C = c] = \text{Ws}[M = m].$$

- Daraus folgt $\text{Ws}[C = c \mid M = m] = \text{Ws}[C = c]$.
- "$\Leftarrow$": Aus $\text{Ws}[C = c \mid M = m] = \text{Ws}[C = c]$ folgt mit dem Satz von Bayes $\text{Ws}[M = m] = \text{Ws}[M = m \mid C = c]$.
- Damit ist Π perfekt sicher.

Ununterscheidbarkeit von Verschlüsselungen

Satz Ununterscheidbarkeit von Verschlüsselungen

Ein Verschlüsselungsverfahren Π ist perfekt sicher gdw für alle $m_0, m_1 \in \mathcal{M}$, $c \in \mathcal{C}$ gilt $\mathrm{Ws}[C = c \mid M = m_0] = \mathrm{Ws}[C = c \mid M = m_1]$.

Beweis:

- "$\Rightarrow$": Mit dem Satz auf voriger Folie gilt für perfekt sichere Π
$$\mathrm{Ws}[C = c \mid M = m_0] = \mathrm{Ws}[C = c] = \mathrm{Ws}[C = c \mid M = m_1].$$
- "$\Leftarrow$": Sei $m' \in \mathcal{M}$ beliebig. Es gilt
$$\begin{aligned} \mathrm{Ws}[C = c] &= \sum_{m \in \mathcal{M}} \mathrm{Ws}[C = c \mid M = m] \cdot \mathrm{Ws}[M = m] \\ &= \mathrm{Ws}[C = c \mid M = m'] \cdot \sum_{m \in \mathcal{M}} \mathrm{Ws}[M = m] \\ &= \mathrm{Ws}[C = c \mid M = m']. \end{aligned}$$
- Die perfekte Sicherheit von Π folgt mit dem Satz auf voriger Folie.

Das One-Time Pad (Vernam Verschlüsselung)

Definition One-Time Pad (1918)

Sei $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^\ell$.

1. **Gen:** Ausgabe $k \in_R \{0,1\}^\ell$
2. **Enc:** Für $m \in \{0,1\}^\ell$ Ausgabe $c = Enc_k(m) := m \oplus k$.
3. **Dec:** Für $c \in \{0,1\}^\ell$ Ausgabe $m = Dec_k(c) := c \oplus k$.

Satz Sicherheit des One-Time Pads

Das One-Time Pad ist perfekt sicher.

Beweis:

- Wegen $C = M \oplus K$ gilt für alle $m_0, m_1 \in \mathcal{M}$ und $c \in \mathcal{C}$

$$\begin{aligned} \mathrm{Ws}[C = c \mid M = m_0] &= \mathrm{Ws}[M \oplus K = c \mid M = m_0] = \mathrm{Ws}[K = m_0 \oplus c] \\ &= \frac{1}{2^\ell} = \mathrm{Ws}[C = c \mid M = m_1]. \end{aligned}$$

- Damit ist das One-Time Pad perfekt sicher.

Beschränkungen perfekter Sicherheit

Satz Größe des Schlüsselraums

Sei Π perfekt sicher. Dann gilt $|\mathcal{K}| \geq |\mathcal{M}|$.

Beweis: Angenommen $|\mathcal{K}| < |\mathcal{M}|$.

- Für $c \in \mathcal{C}$ definiere $D(c) = \{m \mid m = Dec_k(c) \text{ für ein } k \in \mathcal{K}\}$.
- Es gilt $|D(c)| \leq |\mathcal{K}|$, da jeder Schlüssel k genau ein m liefert.
- Wegen $|\mathcal{K}| < |\mathcal{M}|$ folgt $|D(c)| < |\mathcal{M}|$. D.h. es gibt ein $m \in \mathcal{M}$ mit
$$\text{Ws}[M = m \mid C = c] = 0 < \text{Ws}[M = m].$$
- Damit ist Π nicht perfekt sicher.

Satz von Shannon (1949)

Satz von Shannon

Sei $\Pi = (Gen, Enc, Dec)$ mit $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$. Π ist perfekt sicher gdw

1. Gen wählt alle $k \in \mathcal{K}$ gleichverteilt mit Ws $\frac{1}{|\mathcal{K}|}$.
2. Für alle $m \in \mathcal{M}, c \in \mathcal{C}$ existiert genau ein $k \in \mathcal{K}$: $c = Enc_k(m)$.

Beweisidee:

- "$\Leftarrow$": Jedes $c \in \mathcal{C}$ korrespondiert zu genau einem $m \in \mathcal{M}$ via k.
- D.h. c wird zu m entschlüsselt, falls k verwendet wird.
- Dies geschieht gleichverteilt mit Ws $\frac{1}{|\mathcal{K}|}$. Damit gilt

$$\text{Ws}[C = c \mid M = m] = \frac{1}{|\mathcal{K}|} \text{ für alle } m \in \mathcal{M}.$$

- Es folgt $\text{Ws}[C = c \mid M = m_0] = \frac{1}{|\mathcal{K}|} = \text{Ws}[C = c \mid M = m_1]$.

Satz von Shannon (1949)

Beweisidee (Fortsetzung):

- "$\Rightarrow$": Sei Π perfekt sicher mit $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$.
- Ann: $\exists(m, c)$ mit $c \neq Enc_k(m)$ für alle $k \in \mathcal{K}$. Dann gilt $\text{Ws}[M = M|C = c] = 0 < \text{Ws}[M = m]$. (Widerspruch)
- Ann: $\exists(m, c)$ mit $c = Enc_k(m)$ für mehrere $k \in \mathcal{K}$. Dann existiert ein (m', c') mit $c' \neq Enc_k(m')$ für alle $k \in \mathcal{K}$. (Widerspruch)
- Damit gibt es für jedes feste c und jedes m genau einen Schlüssel k_m mit $c = Enc_{k_m}(m)$.
- Daraus folgt für alle m, m'

$$\begin{aligned} \text{Ws}[K = k_m] &= \text{Ws}[C = c \mid M = m] \\ &= \text{Ws}[C = c \mid M = m'] = \text{Ws}[K = k_{m'}]. \end{aligned}$$

- D.h. es gilt $\text{Ws}[K = k] = \frac{1}{|\mathcal{K}|}$ für alle $k \in \mathcal{K}$.

Ununterscheidbarkeit von Chiffretexten

Spiel Ununterscheidbarkeit von Chiffretexten $PrivK^{eav}_{\mathcal{A},\Pi}(n)$

Sei Π ein Verschlüsselungsverfahren und $\mathcal{A}$ ein Angreifer.

1. $(m_0, m_1) \leftarrow \mathcal{A}$.
2. $k \leftarrow Gen(1^n)$.
3. Wähle $b \in_R \{0,1\}$. $b' \leftarrow \mathcal{A}(Enc_k(m_b))$.
4. $PrivK^{eav}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{für } b = b' \\ 0 & \text{sonst} \end{cases}$.

Anmerkungen:

- $\mathcal{A}$ wählt die zu verschlüsselnden Nachrichten m_0, m_1 selbst.
- $\mathcal{A}$ gewinnt das Spiel, d.h. $b = b'$, durch Raten von b' mit Ws $\frac{1}{2}$.
- Wir bezeichnen $\text{Ws}[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2}$ als Vorteil von $\mathcal{A}$.

Raten ist optimal

Satz Perfekte Sicherheit und $PrivK^{eav}_{\mathcal{A},\Pi}$

Ein Verschlüsselungsverfahren Π ist perfekt sicher gdw für alle Angreifer $\mathcal{A}$ gilt $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2}$.

Beweis:

- "$\Leftarrow$": Sei Π nicht perfekt sicher. Dann existieren $m_0, m_1 \in \mathcal{M}$ und $c \in C$ mit $\mathrm{Ws}[C = c \mid M = m_0] \neq \mathrm{Ws}[C = c \mid M = m_1]$.
- OBdA $\mathrm{Ws}[C = c \mid M = m_0] > \mathrm{Ws}[C = c \mid M = m_1]$.
- Wir definieren den folgenden Angreifer $\mathcal{A}$ für das Spiel $PrivK^{eav}_{\mathcal{A},\Pi}$.

Algorithmus Angreifer $\mathcal{A}$

EINGABE: m_0, m_1, c

1. Versende Nachrichten m_0, m_1. Erhalte $c' \leftarrow Enc_k(m_b)$.
2. Falls $c' = c$, setze $b' = 0$. Sonst setze $b' \in_R \{0, 1\}$.

AUSGABE: b'

Nicht perfekt sicher $\Rightarrow$ Vorteil

Beweis (Fortsetzung):

- Es gilt $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi} = 1] = \mathrm{Ws}[\mathcal{A}(Enc(m_b) = b)]$

$$
\begin{aligned}
&= \frac{1}{2} \cdot \mathrm{Ws}[C \neq c] + \mathrm{Ws}[M = m_0 \mid C = c] \cdot \mathrm{Ws}[C = c] \\
&= \frac{1}{2}\big(1 - \mathrm{Ws}[C = c]\big) + \mathrm{Ws}[M = m_0 \mid C = c] \cdot \mathrm{Ws}[C = c].
\end{aligned}
$$

- Falls $\mathrm{Ws}[M = m_0 \mid C = c] > \frac{1}{2}$, so folgt $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi} = 1] > \frac{1}{2}$.

- Es gilt $\mathrm{Ws}[M = m_0 \mid C = c]$

$$
\begin{aligned}
&= \frac{\mathrm{Ws}[C = c \mid M = m_0] \cdot \overbrace{\mathrm{Ws}[M = m_0]}^{\frac{1}{2}}}{\sum_{i=0}^{1} \mathrm{Ws}[C = c \mid M = m_i] \cdot \underbrace{\mathrm{Ws}[M = m_i]}_{\frac{1}{2}}} \\
&= \frac{\mathrm{Ws}[C = c \mid M = m_0]}{\underbrace{\mathrm{Ws}[C = c \mid M = m_0] + \mathrm{Ws}[C = c \mid M = m_1]}_{< 2 \cdot \mathrm{Ws}[C = c \mid M = m_0]}} > \frac{1}{2}.
\end{aligned}
$$

Perfekt sicher $\Rightarrow$ kein Vorteil

Beweis (Fortsetzung): Perfekt sicher $\Rightarrow \mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$

- Sei Π perfekt sicher. Dann gilt für alle $m_0, m_1 \in \mathcal{M}$, $c \in \mathcal{C}$ $\mathrm{Ws}[C = c \mid M = m_0] = \mathrm{Ws}[C = c] = \mathrm{Ws}[C = c \mid M = m_1]$.
- D.h. es gilt $\{c \mid c \in Enc_k(m)\} = \mathcal{C}$ für alle $m \in M$.
- Daraus folgt $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi} = 1] = \mathrm{Ws}[\mathcal{A}(Enc(m_b) = b)]$

$$
\begin{aligned}
&= \mathrm{Ws}[b = 0] \cdot \mathrm{Ws}[\mathcal{A}(Enc(m_0)) = 0] + \mathrm{Ws}[b = 1] \cdot \mathrm{Ws}[\mathcal{A}(Enc(m_1) = 1] \\
&= \frac{1}{2} \cdot \left(\sum_{c \in Enc(m_0)} \mathrm{Ws}[\mathcal{A}(c) = 0 \mid C = c] \cdot \mathrm{Ws}[C = c] \right. \\
&\quad \left. + \sum_{c \in Enc(m_1)} \underbrace{\mathrm{Ws}[\mathcal{A}(c) = 1 \mid C = c]}_{1 - \mathrm{Ws}[\mathcal{A}(c)=0 \mid C=c]} \cdot \mathrm{Ws}[C = c] \right) \\
&= \frac{1}{2} \cdot \sum_{c \in \mathcal{C}} \mathrm{Ws}[C = c] = \frac{1}{2}.
\end{aligned}
$$

Computational Security

Perfekte Sicherheit:

- Liefert Sicherheit im informationstheoretischen Sinn, d.h. der Angreifer erhält nicht genügend Information, um zu entschlüsseln.
- Benötigen Schlüssel der Länge aller zu verschlüsselnden Nachrichten. Dies ist unpraktikabel in der Praxis.

Computational Security Ansatz:

- Wir verwenden kurze Schlüssel (z.B. 128 Bit).
- Liefert Sicherheit nur gegenüber ppt Angreifern.
- Unbeschränkte Angreifer können bei KPA-Angriff $\mathcal{K}$ durchsuchen.
- Seien $(m_1, c_1), \ldots, (m_n, c_n)$ die Plaintext/Chiffretext Paare.
- Mit hoher Ws existiert eindeutiges k mit $m_i = Dec_k(c_i)$, $i \in [n]$.
- Mit obigem KPA-Angriff kann der Angreifer in Polynomial-Zeit auch ein einzelnes $k \in \mathcal{K}$ raten, dieses ist korrekt mit Ws $\frac{1}{|\mathcal{K}|}$.
- D.h. ppt Angreifer besitzen nur vernachlässigbare Erfolgsws im Sicherheitsparameter.

Vernachlässigbare Wahrscheinlichkeit

Definition Vernachlässigbare Wahrscheinlichkeit

Eine Funktion $f : \mathbb{N} \to \mathbb{R}$ heißt *vernachlässigbar*, falls für jedes Polynom p ein $N \in \mathbb{N}$ existiert, so dass für alle $n \geq N$ gilt $f(n) < \frac{1}{p(n)}$.
Notation: $f(n) = \mathrm{negl}(n)$.

Bsp:

- Vernachlässigbare Funktionen: $\frac{1}{2^n}$, $\frac{1}{2^{\sqrt{n}}}$, $\frac{1}{2^{\log^2 n}}$, $\frac{1}{n^{\frac{\log n}{\log\log n}}}$.
- Nicht vernachlässigbare Funktionen: $\frac{1}{n^2}$, $\frac{1}{\log n}$, $\frac{1}{2^{\mathcal{O}(\log n)}}$.

Korollar Komposition vernachlässigbarer Funktionen

Seien f_1, f_2 vernachlässigbare Funktionen. Dann ist

1. $f_1 + f_2$ vernachlässigbar.
2. $q(n) \cdot f_1$ vernachlässigbar für jedes Polynom q.

Sicherheitsbeweis per Reduktion

Annahme: Problem X lässt sich in ppt nur mit Ws $\mathrm{negl}(n)$ lösen.

- Sei Π ein Krypto-Verfahren mit Sicherheitsparameter n.
- Sei $\mathcal{A}$ ein ppt Angreifer auf Π mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren eine polynomielle Reduktion $\mathcal{A}$' für $X \leq_p \mathcal{A}$. (Erinnerung: Diskrete Mathematik II)

Algorithmus Reduktion $\mathcal{A}'$ für $X \leq_p \mathcal{A}$

EINGABE: Instanz x des Problems X

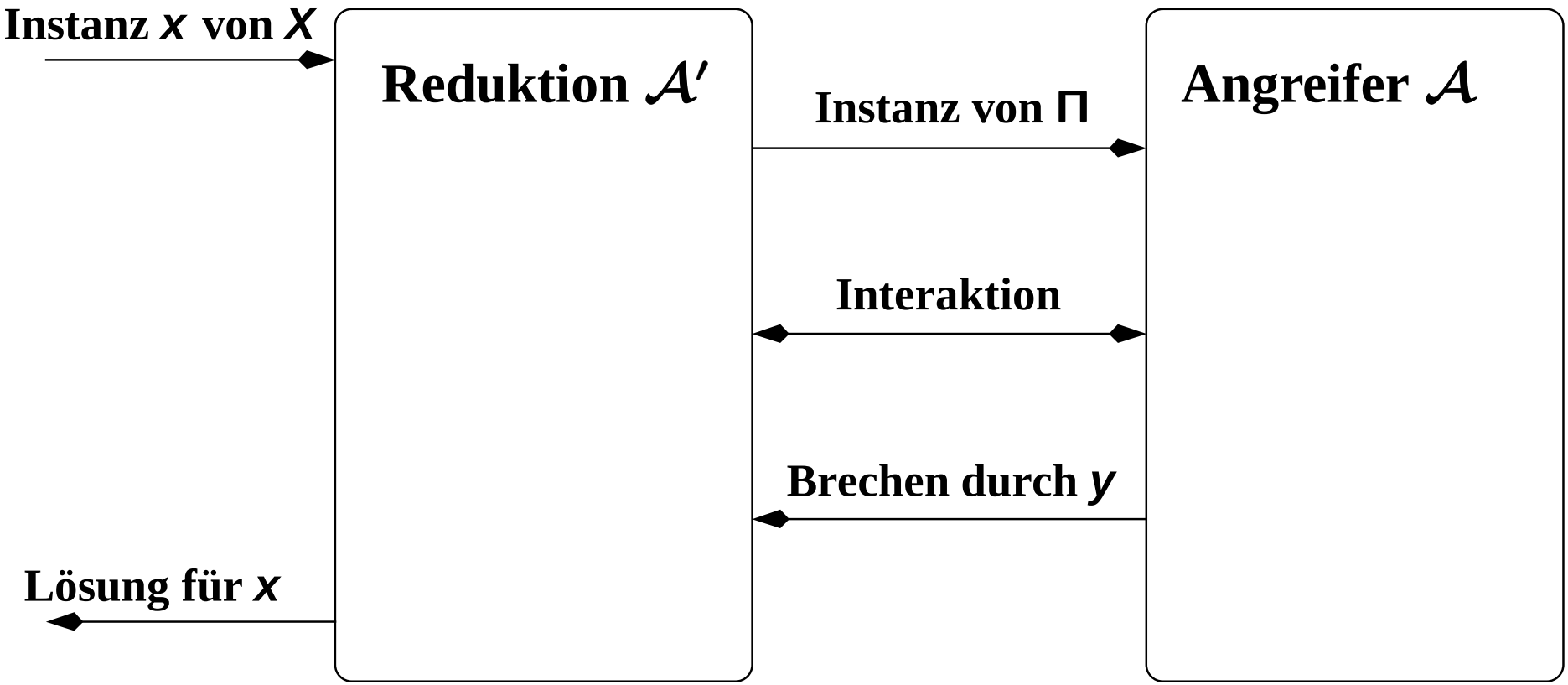
1. Konstruieren aus x Instanz von Π, senden diese an $\mathcal{A}$.
2. Sofern $\mathcal{A}$s Angriff eine Interaktion erfordert (z.B. bei CCA), wird diese von der Reduktion simuliert. $\mathcal{A}$s Sicht soll dabei identisch zu einem realen Angriff sein.
3. $\mathcal{A}$ bricht schließlich Π mittels Ausgabe y mit Ws $\epsilon(n)$.
4. Wir verwenden y, um eine Lösung für die Instanz x zu berechnen.

AUSGABE: Lösung für x

Sicherheitsbeweis per Reduktion

- Alle Schritte der Reduktion laufen in polynomial-Zeit.
- Angenommen Schritt 4 besitze Erfolgws $\frac{1}{p(n)}$ für ein Polynom $p(n)$.
- Dann besitzt die Reduktion insgesamt Erfolgsws $\frac{\epsilon(n)}{p(n)}$.
- Nach Annahme lässt sich X nur mit Ws $\mathrm{negl}(n)$ lösen.
- D.h. $\frac{\epsilon(n)}{p(n)} \leq \mathrm{negl}(n)$, und damit folgt $\epsilon(n) \leq \mathrm{negl}(n)$.
- Damit besitzt **jeder** Angreifer $\mathcal{A}$ vernachlässigbare Erfolgws.

Reduktionsbeweis bildlich



Computational Security gegenüber Lauschern

Definition Ununterscheidbare Chiffretexte

Ein Verschlüsselungsschema $\Pi = (Gen, Enc, Dec)$ besitzt *ununterscheidbare Chiffretexte gegenüber KPA* falls für alle ppt $\mathcal{A}$ gilt

$$\mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

Der Wsraum ist definiert über die Münzwürfe von *Gen* und $\mathcal{A}$.

Die Differenz $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2}$ bezeichnen wir als Vorteil von $\mathcal{A}$. Π heißt *KPA-sicher*, falls der Vorteil vernachlässigbar ist.

Chiffretext liefert kein einzelnes Bit des Klartextes

Notation: Sei m^i das i-te Bit einer Nachricht $m \in \{0,1\}^n$.

Satz

Sei Π KPA-sicher. Dann gilt für alle ppt $\mathcal{A}$ und alle $i \in [n]$:
$\mathrm{Ws}[\mathcal{A}(Enc_k(m) = m^i)] \leq \frac{1}{2} + \mathrm{negl}(n)$.

Beweis:

- Sei $I_0^n = \{m \in \{0,1\}^n \mid m^i = 0\}$ und $I_1^n = \{m \in \{0,1\}^n \mid m^i = 1\}$.
- Sei $\mathcal{A}$ ein Unterscheider für das i-te Bit mit Vorteil $\epsilon(n)$.
- Konstruieren Angreifer $\mathcal{A}'$, der $m_0 \in I_0$ und $m_1 \in I_1$ unterscheidet.

Algorithmus KPA-Angreifer $\mathcal{A}'$

EINGABE: 1^n

1. Wähle $m_0 \in_R I_0^n$, $m_1 \in_R I_1^n$.
2. Empfange $Enc_k(m_b)$ aus $PrivK_{\mathcal{A}',\Pi}(n)$-Spiel mit $b \in_R \{0,1\}$.
3. $b' \leftarrow \mathcal{A}(Enc_k(m_b))$

AUSGABE: $b' \in \{0,1\}$

Chiffretext liefert kein einzelnes Bit des Klartextes

Beweis: Es gilt $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A}',\Pi}(n) = 1] = \mathrm{Ws}[\mathcal{A}'(Enc_k(m_b) = b)]$

$$
\begin{aligned}
&= \sum_{i=0}^{1} \underbrace{\mathrm{Ws}[b = i]}_{\frac{1}{2}} \cdot \mathrm{Ws}[\mathcal{A}'(Enc_k(m_i)) = i] \\
&= \sum_{i=0}^{1} \mathrm{Ws}[m \in I_i^n] \cdot \mathrm{Ws}[\mathcal{A}(Enc_k(m)) = i \mid m \in I_i^n] \\
&= \mathrm{Ws}[\mathcal{A}(Enc_k(m)) = m^i] = \frac{1}{2} + \epsilon(n)
\end{aligned}
$$

- Da Π KPA-sicher ist, gilt $\mathrm{Ws}[PrivK^{eav}_{\mathcal{A}',\Pi}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$.
- Daraus folgt, dass $\mathcal{A}$ Vorteil $\epsilon \leq \mathrm{negl}(n)$ besitzt.

Chiffretext liefert ppt $\mathcal{A}$ keine Information

Ziel: $\mathcal{A}$ kann aus $Enc_k(m)$ keine Funktion $f(m)$ berechnen.

- Sei $\mathcal{M} \subseteq \{0,1\}^*$ und $S_n = \mathcal{M} \cap \{0,1\}^n$.
- Wir wählen ein $m \in_R S_n$.

Satz Nicht-Berechenbarkeit von Funktionen

Sei Π KPA-sicher. Für jeden ppt Angreifer $\mathcal{A}$ existiert ein ppt Algorithmus $\mathcal{A}'$, so dass für alle ppt-berechenbaren Funktionen f

$$|\mathrm{Ws}[\mathcal{A}(1^n, Enc_k(m)) = f(m)] - \mathrm{Ws}[\mathcal{A}'(1^n) = f(m)]| \leq \mathrm{negl}(n).$$

Wsraum: Zufällige Wahl von m, k, Münzwürfe von $\mathcal{A}, \mathcal{A}', Enc$.

Beweis:

- Wir zeigen zunächst, dass für alle ppt $\mathcal{A}$ gilt
$$|\mathrm{Ws}[\mathcal{A}(1^n, Enc_k(m)) = f(m)] - \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(1^n)) = f(m)]| \leq \mathrm{negl}(n).$$
- Wir konstruieren dazu KPA-Angreifer D auf Π mittels $\mathcal{A}$.

$\mathcal{A}$ kann $Enc_k(m)$ und $Enc_k(1^n)$ nicht unterscheiden.

Algorithmus Angreifer D im Spiel $PrivK^{eav}_{D,\Pi}(n)$

EINGABE: 1^n.

1. Wähle $m_0 = m \in_R S_n, m_1 = 1^n$. Erhalte $Enc_k(m_b)$ für $b \in_R \{0,1\}$.
2. Sende $(1^n, Enc_k(m_b))$ an $\mathcal{A}$. Erhalte Ausgabe $f(m_b)$.

AUSGABE: $b' = \begin{cases} 0 & \text{falls } f(m) = f(m_b) \\ 1 & \text{sonst} \end{cases}$.

Fall 1: $b = 0$, d.h. $\mathcal{A}$ erhält $Enc_k(m)$.

- Es gilt $\mathrm{Ws}[PrivK_{D,\Pi} = 1 \mid b = 0] = \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(m)) = f(m)]$.

Fall 2: $b = 1$, d.h. $\mathcal{A}$ erhält $Enc_k(1^n)$.

- Es gilt $\mathrm{Ws}[PrivK_{D,\Pi} = 1 \mid b = 1] = \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(1^n)) \neq f(m)]$
  $= 1 - \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(1^n) = f(m)]$.

$\mathcal{A}$ kann $Enc_k(m)$ und $Enc_k(1^n)$ nicht unterscheiden.

- Insgesamt folgt damit aus der KPA-Sicherheit von Π

$$\begin{aligned}
\mathrm{negl}(n) &\geq \left|\frac{1}{2} - \mathrm{Ws}[PrivK_{D,\Pi} = 1]\right| \\
&= \left|\frac{1}{2} - \sum_{i\in\{0,1\}} \mathrm{Ws}[PrivK_{D,\Pi} = 1 \mid b = i] \cdot \mathrm{Ws}[b = i]\right| \\
&= \Big|\frac{1}{2} - \frac{1}{2}\big(\mathrm{Ws}[\mathcal{A}(1^n, Enc_k(m)) = f(m)] \\
&\qquad +1 - \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(1^n)) = f(m)]\big)\Big|.
\end{aligned}$$

- Daraus folgt wie gewünscht
$|\mathrm{Ws}[\mathcal{A}(1^n, Enc_k(m)) = f(m)] - \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(1^n)) = f(m)]| \leq \underbrace{2\mathrm{negl}(n)}_{\mathrm{negl}(n)}.$

Konstruktion von $\mathcal{A}'$

Algorithmus $\mathcal{A}'$

EINGABE: 1^n

1. Berechne $k \leftarrow Gen(1^n)$ und $c \leftarrow Enc_k(1^n)$.
2. $f(m) \leftarrow \mathcal{A}(1^n, Enc_k(1^n))$.

AUSGABE: $f(m)$

Unser Satz zur Nicht-Berechenbarkeit von Funktionen folgt aus

$$\mathrm{Ws}[\mathcal{A}'(1^n) = f(m)] = \mathrm{Ws}[\mathcal{A}(1^n, Enc_k(1^n)) = f(m)].$$

Semantische Sicherheit

Semantische Sicherheit (informal): Erweiterung auf:

- Beliebige Verteilung anstatt Gleichverteilung $m \in_R S_n$.
- $\mathcal{A}$ und $\mathcal{A}'$ erhalten zusätzliche Information über den Klartext.

Man kann zeigen:

> Semantische Sicherheit ist äquivalent zu KPA-Sicherheit.

Pseudozufälligkeit

Motivation: Pseudozufallsgenerator

- One-Time Pad: Sicherheit von $m \oplus k$ für $m \in \{0,1\}^n$, $k \in_R \{0,1\}^n$.
- D.h. wir benötigen einen echten Zufallsstring $k \in \{0,1\}^n$.
- Sei G ein Algorithmus, der eine Verteilung $\mathcal{D}$ auf $\{0,1\}^n$ liefert.
- Falls es für ppt D unmöglich ist, $\mathcal{D}$ von der Gleichverteilung auf $\{0,1\}^n$ zu unterscheiden, so können wir k mittels G wählen.

Definition Pseudozufallsgenerator

Sei G ein pt Algorithmus, der eine Funktion $\{0,1\}^n \to \{0,1\}^{\ell(n)}$ mit $\ell(n) > n$ berechne. G heißt *Pseudozufallsgenerator* falls für alle ppt D

$$|\mathrm{Ws}[D(r) = 1] - \mathrm{Ws}[D(G(s)) = 1]| \leq \mathrm{negl}(n),$$

wobei $r \in_R \{0,1\}^{\ell(n)}$ und $s \in_R \{0,1\}^n$, die sogenannte *Saat*.
Wsraum: Zufällige Wahl von r, s, Münzwürfe von D.

Anmerkung: G expandiert die echt zufällige Saat $s \in \{0,1\}^n$ in ein pseudozufälliges $G(s) \in \{0,1\}^{\ell(n)}$ mit Expansionsfaktor $\ell(n)$.

Unterscheider D mit beliebiger Laufzeit

Satz Unterscheider D mit beliebiger Laufzeit

Sei $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ ein Pseudozufallsgenerator. Dann existiert ein Unterscheider D mit Laufzeit $\mathcal{O}(2^n \cdot \text{Laufzeit}(G))$ und Erfolgsws

$$\text{Ws}[D(r) = 1] - \text{Ws}[D(G(s)) = 1] \geq \tfrac{1}{2}.$$

Beweis:

- D prüft, ob w mittels G generiert werden kann.

Algorithmus Unterscheider D

EINGABE: $w \in \{0,1\}^{\ell(n)}$

1. Berechne $G(s)$ für alle $s \in \{0,1\}^n$.

AUSGABE: $= \begin{cases} 1 & \text{falls } G(s) = w \text{ für ein } s \in \{0,1\}^n \\ 0 & \text{sonst} \end{cases}$.

Unterscheider D mit beliebiger Laufzeit

1. Fall: $w \in G(s)$, d.h. w wurde mittels G generiert

- Dann gilt $Ws[D(G(s)) = 1] = 1$.

2.Fall: $w = r \in_R \{0,1\}^{\ell(n)}$, d.h. w ist echt zufällig.

- Es gilt $|\{y \in \{0,1\}^{\ell(n)} \mid y = G(s) \text{ für ein } s \in \{0,1\}^n\}| \leq 2^n$.
- Damit ist $r \in_R \{0,1\}^{\ell(n)}$ mit Ws $\leq 2^{n-\ell(n)}$ im Bildraum von G.
- D.h. $Ws[D(r) = 1] \leq 2^{n-\ell(n)} \leq \frac{1}{2}$ wegen $\ell(n) > n$.

Daraus folgt insgesamt $\text{Ws}[D(r) = 1] - \text{Ws}[D(G(s)) = 1] \geq \frac{1}{2}$.

Stromchiffre

Algorithmus Stromchiffre

Sei G ein Pseudozufallsgenerator mit Expansionsfaktor $\ell(n)$. Wir definieren $\Pi_s = (Gen, Enc, Dec)$ mit Sicherheitsparameter n für Nachrichten der Länge $\ell(n)$.

1. **Gen:** Wähle $k \in_R \{0,1\}^n$.
2. **Enc:** Bei Eingabe $k \in \{0,1\}^n$ und $m \in \{0,1\}^{\ell(n)}$, berechne
$$c := G(k) \oplus m.$$
3. **Dec:** Bei Eingabe $k \in \{0,1\}^n$ und $c \in \{0,1\}^{\ell(n)}$, berechne
$$m := G(k) \oplus c.$$

Anmerkung:

- Π_s verwendet $G(k)$ anstatt $r \in \{0,1\}^{\ell(n)}$ wie im One-Time Pad.
- D.h. wir benötigen nur n statt $\ell(n)$ echte Zufallsbits.
(Bsp: n 128 Bit, $\ell(n)$ mehrere Megabyte)

Sicherheit unserer Stromchiffre

Satz Sicherheit von Π_s

Sei G ein Pseudozufallsgenerator. Dann ist Π_s KPA-sicher.

Beweis:

- Idee: Erfolgreicher Angreifer $\mathcal{A}$ liefert Unterscheider für G.
- Sei $\mathcal{A}$ ein KPA-Angreifer auf Π_s mit Vorteil $\epsilon(n)$.
- Wir konstruieren mittels $\mathcal{A}$ folgenden Unterscheider U für G.

Algorithmus Unterscheider U

EINGABE: $w \in \{0,1\}^{\ell(n)}$

1. Erhalte $(m_0, m_1) \leftarrow \mathcal{A}(1^n)$
2. Wähle $b \in_R \{0,1\}$ und berechne $c := w \oplus m_b$.
3. Erhalte $b' \leftarrow \mathcal{A}(c)$.

$$\text{AUSGABE:} = \begin{cases} 1 & \text{falls } b' = b, \text{ Interpretation: } w = G(k), k \in_R \{0,1\}^n \\ 0 & \text{sonst,} \qquad\quad \text{Interpretation: } w \in_R \{0,1\}^{\ell(n)} \end{cases}.$$

Sicherheit von Π_s

Fall 1: $w = r \in_R \{0,1\}^{\ell(n)}$, d.h. w ist ein echter Zufallsstring.

- Dann ist die Verteilung von c identisch zur Verteilung beim One-Time Pad Π_{otp}.
- Damit folgt aus der perfekten Sicherheit des One-Time Pads

$$\text{Ws}[D(w) = 1] = \text{Ws}[PrivK^{eav}_{\mathcal{A},\Pi_{\text{otp}}}(n) = 1] = \tfrac{1}{2}.$$

Fall 2: $w = G(k)$ für $k \in_R \{0,1\}^n$, d.h. w wurde mittels G generiert.

- Damit ist die Verteilung von c identisch zur Verteilung in Π_s.
- Es folgt $\text{Ws}[D(w) = 1] = \text{Ws}[PrivK^{eav}_{\mathcal{A},\Pi_s}(n) = 1] = \frac{1}{2} + \epsilon(n)$.

Aus der Pseudozufälligkeit von G folgt insgesamt

$$\text{negl}(n) \geq \Big| \underbrace{\text{Ws}[D(r) = 1]}_{\frac{1}{2}} - \underbrace{\text{Ws}[D(G(k)) = 1]}_{\frac{1}{2}+\epsilon(n)} \Big| = \epsilon(n).$$

Damit ist der Vorteil jedes Angreifers $\mathcal{A}$ vernachlässigbar.

Generator mit variabler Ausgabelänge

Ziel: Um Nachricht beliebiger Länge $\ell(n)$ mit Algorithmus Π_s zu verschlüsseln, benötigen wir ein G mit variabler Ausgabelänge $\ell(n)$.

Definition Pseudozufallsgenerator mit variabler Ausgabelänge

Ein pt Algorithmus G heißt *Pseudozufallsgenerator mit variabler Ausgabelänge* falls

1. Für eine Saat $s \in \{0,1\}^n$ und eine Länge $\ell \in N$ berechnet $G(s, 1^\ell)$ einen String der Länge ℓ.
2. Für jedes Polynom $\ell(\cdot)$ ist $G_\ell(s) := G(s, 1^{\ell(n)})$, $s \in \{0,1\}^n$ ein Pseudozufallsgenerator mit Expansionsfaktor $\ell(n)$.
3. Für alle s, ℓ, ℓ' mit $\ell \leq \ell'$ ist $G(s, 1^\ell)$ ein Präfix von $G(s, 1^{\ell'})$.

Anmerkungen:

- Für Nachricht m erzeugen wir Chiffretext $c := G(k, 1^{|m|}) \oplus m$.
- Bedingung 3 ist technischer Natur, um im KPA-Spiel Verschlüsselungen von m_0, m_1 beliebiger Länge zuzulassen.

Existenz Zufallsgenerator mit/ohne variable Länge

Fakt Existenz von Zufallsgeneratoren

1. Die Existenz von Pseudozufallsgeneratoren folgt unter der Annahme der Existenz von sogenannten Einwegfunktionen.
2. Pseudozufallsgeneratoren variabler Ausgabelänge können aus jedem Pseudozufallsgenerator fixer Länge konstruiert werden.

Vereinfacht:

$$\begin{aligned} \text{Einwegfunktion} &\Rightarrow \text{Pseudozufallsgenerator fixer Länge} \\ &\Rightarrow \text{Pseudozufallsgenerator variabler Länge} \end{aligned}$$

(mehr dazu im Verlauf der Vorlesung)

Diskussion Stromchiffren

Stromchiffre:

- Pseudozufallsgeneratoren mit variabler Ausgabelänge liefern Strom von Zufallsbits.
- Wir nennen diese Stromgeneratoren auch Stromchiffren.

Stromchiffren in der Praxis:

- Beispiele: LFSRs, RC4, SEAL, A5/1 und Bluetooth.
- Viele Stromchiffren in der Praxis sind sehr schnell, allerdings sind die meisten leider ad hoc Lösungen ohne Sicherheitsbeweis.
- Schwächen in RC4 führten zum Brechen des WEP Protokolls.
- LFSRs sind kryptographisch vollständig gebrochen worden.
- Seit 2004: Ecrypt-Projekt eStream zur Etablierung sicherer Standard-Stromchiffren.
- Derzeitige Kandidaten: HC-128, Rabbit, Salsa20/12, SOSEMANUK, Grain v1, MICKEY v2 und Trivium.

Sicherheit mehrfacher Verschlüsselung

Bisher: Angreifer $\mathcal{A}$ erhält nur *eine* Verschlüsselung. Nachrichten müssen aber sicher bleiben, falls $\mathcal{A}$ mehrere Chiffretexte erhält.

Spiel Mehrfache Verschlüsselung $PrivK^{mult}_{\mathcal{A},\Pi}(n)$

Sei Π ein Verschlüsselungsverfahren und $\mathcal{A}$ ein Angreifer.

1. $(M_0, M_1) \leftarrow \mathcal{A}(1^n)$ mit $M_0 = (m_0^1, \ldots, m_0^t)$, $M_1 = (m_1^1, \ldots, m_1^t)$ und $|m_0^i| = |m_1^i|$ für alle $i \in [t]$.
2. $k \leftarrow Gen(1^n)$.
3. Wähle $b \in_R \{0,1\}$. $b' \leftarrow \mathcal{A}((Enc_k(m_b^1), \ldots, Enc_k(m_b^t))$.
4. $PrivK^{mult}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{für } b = b' \\ 0 & \text{sonst} \end{cases}$.

Mult-KPA Sicherheit

Definition Mult-KPA Sicherheit

Ein Verschlüsselungsschema $\Pi = (Gen, Enc, Dec)$ besitzt *ununterscheidbare mehrfache Chiffretexte gegenüber KPA* falls für alle ppt $\mathcal{A}$:

$$\mathrm{Ws}[PrivK^{mult}_{\mathcal{A},\Pi}(n) = 1] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

Der Wsraum ist definiert über die Münzwürfe von $\mathcal{A}$ und $PrivK^{mult}_{\mathcal{A},\Pi}$.

Notation: Wir bezeichnen Π als *mult-KPA sicher*.

KPA Sicherheit vs. mult-KPA Sicherheit

Satz KPA Sicherheit vs. mult-KPA Sicherhei

KPA Sicherheit impliziert **nicht** mult-KPA Sicherheit.

Beweis:

- Π_s ist KPA-sicher. Wir betrachten folgendes mult-KPA Spiel.

Algorithmus Angreifer $\mathcal{A}$ für Π_s

EINGABE: Sicherheitsparameter n

1. $(M_0, M_1) \leftarrow \mathcal{A}(1^n)$ mit $M_0 = (0^n, 0^n)$, $M_1 = (0^n, 1^n)$.
2. Erhalte $C = (c_1, c_2)$.

AUSGABE: $b' = \begin{cases} 1 & \text{falls } c_1 = c_2 \\ 0 & \text{sonst} \end{cases}$.

- Da *Enc* von Π_s deterministisch ist, gilt $\mathrm{Ws}[PrivK^{mult}_{\mathcal{A},\Pi_s}(n) = 1] = 1$.

Unsicherheit deterministischer Verschlüsselung

Korollar Unsicherheit deterministischer Verschlüsselung

Sei $\Pi = (Gen, Enc, Dec)$ mit deterministischer Verschlüsselung Enc. Dann ist Π unsicher gegenüber mult-KPA Angriffen.

- Voriger Angreifer $\mathcal{A}$ nutzt lediglich, dass für zwei identische Nachrichten $m_0 = m_1$ auch die Chiffretexte identisch sind.

Notwendig: Wir benötigen randomisiertes Enc, dass identische Nachrichten auf unterschiedliche Chiffretexte abbildet.

Synchronisierte sichere mehrfache Verschlüsselung

Synchronisierter Modus für Stromchiffren:

- Nutze für Nachrichten $m_1, m_2, \ldots, m_n$ sukzessive Teil des Bitstroms $G(s) = s_1 s_2 \ldots s_n$ mit $|s_i| = |m_i|$.
- D.h. es werden nie Teile des Bitstroms wiederverwendet.
- Ermöglicht einfaches Protokoll zur Kommunikation von A und B:
  - A verschlüsselt mit s_1, B entschlüsselt mit s_1.
  - Danach verschlüsselt B mit s_2, mit dem auch A entschlüsselt, usw.
- Erfordert, dass A und B die Position im Bitstrom *synchronisieren*.
- Verfahren ist sicher, da die Gesamtheit der Nachrichten als einzelne Nachricht $m = m_1 \ldots m_n$ aufgefasst werden kann.

Nicht-synchronisierte mehrfache Verschlüsselung

Nicht-synchronisierter Modus für Stromchiffren:

- Erweitern Funktionalität von Pseudozufallsgeneratoren G:
  1. G erhält zwei Eingaben: Schlüssel k und Initialisierungsvektor IV.
  2. $G(k, IV)$ ist pseudozufällig selbst für bekanntes IV.
- Verschlüsselung von m mit erweiterten Pseudozufallsgeneratoren:

$$Enc_k(m) := (IV, G(k, IV) \oplus m) \text{ für } IV \in_R \{0,1\}^n.$$

- Entschlüsselung möglich, da IV im Klartext mitgesendet wird.
- D.h. eine Nachricht m besitzt 2^n mögliche Verschlüsselungen.
- **Warnung:** Konstruktion solch erweiterter G ist nicht-trivial.

CPA Spiel

Szenario: Wir betrachten aktive Angriffe.

- D.h. $\mathcal{A}$ darf sich Nachrichten nach Wahl verschlüsseln lassen.
- $\mathcal{A}$ erhält dazu Zugriff auf ein Verschlüsselungsorakel $Enc_k(\cdot)$.
- Notation für die Fähigkeit des Orakelzugriffs: $\mathcal{A}^{Enc_k(\cdot)}$.

Spiel CPA Ununterscheidbarkeit von Chiffretexten $PrivK^{cpa}_{\mathcal{A},\Pi}(n)$

Sei Π ein Verschlüsselungsverfahren und $\mathcal{A}$ ein Angreifer.

1. $k \leftarrow Gen(1^n)$.
2. $(m_0, m_1) \leftarrow \mathcal{A}^{Enc_k(\cdot)}(1^n)$, d.h. $\mathcal{A}$ darf $Enc_k(m)$ für beliebige m anfragen.
3. Wähle $b \in_R \{0,1\}$ und verschlüssele $c \leftarrow Enc_k(m_b)$.
4. $b' \leftarrow \mathcal{A}^{Enc_k(\cdot)}(c)$, d.h. $\mathcal{A}$ darf $Enc_k(m)$ für beliebige m anfragen.
5. $$PrivK^{cpa}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{für } b = b' \\ 0 & \text{sonst} \end{cases}.$$

CPA Sicherheit

Definition CPA Sicherheit

Ein Verschlüsselungsschema $\Pi = (Gen, Enc, Dec)$ besitzt *ununterscheidbare Chiffretexte gegenüber CPA* falls für alle ppt $\mathcal{A}$:

$$\mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

Der Wsraum ist definiert über die Münzwürfe von $\mathcal{A}$ und $PrivK^{cpa}_{\mathcal{A},\Pi}$.

Notation: Wir bezeichnen Π als *CPA sicher*.

CPA-Unsicherheit deterministischer Verschlüsselung

Satz Unsicherheit deterministischer Verschlüsselung

Sei $\Pi = (Gen, Enc, Dec)$ ein Verschlüsselungsschema mit deterministischem Enc. Dann ist Π **nicht** CPA-sicher.

Beweis: Konstruieren folgenden CPA Angreifer $\mathcal{A}$.

Algorithmus CPA Angreifer $\mathcal{A}$

EINGABE: 1^n

1. Sende (m_0, m_1) für beliebige verschiedene $m_0, m_1 \in \mathcal{M}$.
2. Erhalte $c := Enc_k(m_b)$ für $b \in_R \{0,1\}$.
3. Stelle Orakelanfrage $c_0 := Enc_k(m_0)$.

AUSGABE: $b' = \begin{cases} 0 & \text{falls } c = c' \\ 1 & \text{sonst} \end{cases}$.

- Es gilt $\mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] = 1$.

Mult-CPA Spiel

Wie CPA-Spiel, nur dass mehrfache Verschlüsselungen erlaubt sind.

Spiel Mehrfache Verschlüsselung $PrivK^{mult-cpa}_{\mathcal{A},\Pi}(n)$

Sei Π ein Verschlüsselungsverfahren und $\mathcal{A}$ ein Angreifer.

1. $(M_0, M_1) \leftarrow \mathcal{A}^{Enc_k(\cdot)}(1^n)$ mit $M_0 = (m_0^1, \ldots, m_0^t)$, $M_1 = (m_1^1, \ldots, m_1^t)$ und $|m_0^i| = |m_1^i|$ für alle $i \in [t]$.
2. $k \leftarrow Gen(1^n)$.
3. Wähle $b \in_R \{0,1\}$. $b' \leftarrow \mathcal{A}^{Enc_k(\cdot)}((Enc_k(m_b^1), \ldots, Enc_k(m_b^t))$.
4. $PrivK^{mult-cpa}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{für } b = b' \\ 0 & \text{sonst} \end{cases}$.

Definition Mult-CPA Sicherheit

Π heißt *mult-CPA* sicher, falls für alle ppt $\mathcal{A}$ gilt

$$\mathrm{Ws}[PrivK^{mult-cpa}_{\mathcal{A},\Pi}(n) = 1] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

CPA-Sicherheit mehrfacher Verschlüsselung

Satz CPA-Sicherheit mehrfacher Verschlüsselung

Sei Π ein Verschlüsselungsschema. Dann ist Π CPA-sicher gdw Π mult-CPA sicher ist.

Beweis "$\Rightarrow$": Für $t = 2$. Rückrichtung ist trivial.

- Ein Angreifer $\mathcal{A}$ gewinnt das Spiel $PrivK^{mult-cpa}_{\mathcal{A},\Pi}(n)$ mit Ws

$$\frac{1}{2}\mathrm{Ws}[\mathcal{A}(Enc_k(m_0^1), Enc_k(m_0^2)) = 0] + \frac{1}{2}\mathrm{Ws}[\mathcal{A}(Enc_k(m_1^1), Enc_k(m_1^2)) = 1].$$

- Daraus folgt $\mathrm{Ws}[PrivK^{mult-cpa}_{\mathcal{A},\Pi}(n) = 1] + \frac{1}{2} =$

$$\frac{1}{2}\mathrm{Ws}[\mathcal{A}(Enc_k(m_0^1), Enc_k(m_0^2)) = 0] + \frac{1}{2}\mathrm{Ws}[\mathcal{A}(Enc_k(m_1^1), Enc_k(m_1^2)) = 1]$$
$$+ \quad \frac{1}{2}\left(\mathrm{Ws}[\mathcal{A}(Enc_k(m_0^1), Enc_k(m_1^2)) = 0] + \mathrm{Ws}[\mathcal{A}(Enc_k(m_0^1), Enc_k(m_1^2)) = 1]\right)$$

- **Ziel:** Zeigen, dass $\mathrm{Ws}[PrivK^{mult-cpa}_{\mathcal{A},\Pi}(n) = 1] + \frac{1}{2} \leq 1 + \mathrm{negl}(n)$.

Betrachten der Hybride

Lemma

$\frac{1}{2}\mathrm{Ws}[\mathcal{A}(Enc_k(m_0^1), Enc_k(m_0^2)) = 0] + \frac{1}{2}\mathrm{Ws}[\mathcal{A}(Enc_k(m_0^1), Enc_k(m_1^2)) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$.

Beweis: Sei $\mathcal{A}'$ Angreifer für *einfache* Verschlüsselungen.

- $\mathcal{A}'$ versucht mittels $\mathcal{A}$ das Spiel $PrivK^{cpa}_{\mathcal{A}',\Pi}(n)$ zu gewinnnen.

Strategie von CPA Angreifer $\mathcal{A}'$

EINGABE: 1^n und Orakelzugriff $Enc_k(\cdot)$

1. $\mathcal{A}'$ gibt 1^n und Orakelzugriff $Enc_k(\cdot)$ an $\mathcal{A}$ weiter.
2. $(M_0, M_1) \leftarrow \mathcal{A}^{Enc_k(\cdot)}(1^n)$ mit $M_0 = (m_0^1, m_0^2)$ und $M_1 = (m_1^1, m_1^2)$.
3. $\mathcal{A}'$ gibt (m_0^2, m_1^2) aus. $\mathcal{A}'$ erhält Chiffretext $c := Enc_k(m_b^2)$.
4. $b' \leftarrow \mathcal{A}(Enc_k(m_0^1), c)$.

AUSGABE: b'

- $\mathrm{Ws}[\mathcal{A}'(Enc_k(m_0^2)) = 0] = \mathrm{Ws}[\mathcal{A}((Enc_k(m_0^1), Enc_k(m_0^2)) = 0]$ und
- $\mathrm{Ws}[\mathcal{A}'(Enc_k(m_1^2)) = 1] = \mathrm{Ws}[\mathcal{A}((Enc_k(m_0^1), Enc_k(m_1^2)) = 1]$.

Fortsetzung Hybridtechnik

Beweis(Fortsetzung):

- CPA Sicherheit von Π bei einzelnen Nachrichten impliziert

$$\begin{aligned}
\frac{1}{2} + \mathrm{negl}(n) &\geq \mathrm{Ws}[PrivK^{cpa}_{\mathcal{A}',\Pi}(n) = 1] \\
&= \frac{1}{2}\mathrm{Ws}[\mathcal{A}'(Enc_k(m_0^2)) = 0] + \frac{1}{2}\mathrm{Ws}[\mathcal{A}'(Enc_k(m_1^2)) = 1] \\
&= \frac{1}{2}\mathrm{Ws}[\mathcal{A}((Enc_k(m_0^1), Enc_k(m_0^2)) = 0] + \\
&\quad \frac{1}{2}\mathrm{Ws}[\mathcal{A}((Enc_k(m_0^1), Enc_k(m_1^2)) = 1] \quad \square_{\text{Lemma}}
\end{aligned}$$

- Analog kann gezeigt werden, dass

$$\begin{aligned}
\frac{1}{2} + \mathrm{negl}(n) &\geq \frac{1}{2}\mathrm{Ws}[\mathcal{A}((Enc_k(m_0^1), Enc_k(m_1^2)) = 0] + \\
&\quad \frac{1}{2}\mathrm{Ws}[\mathcal{A}((Enc_k(m_1^1), Enc_k(m_1^2)) = 1]
\end{aligned}$$

- Daraus folgt $\mathrm{Ws}[PrivK^{mult}_{\mathcal{A},\Pi}(n)] + \frac{1}{2} \leq 1 + \mathrm{negl}(n)$. $\square_{\text{Satz für } t=2}$

Von fester zu beliebiger Nachrichtenlänge

- Beweistechnik für allgemeines t: Definiere für $0 \leq i \leq t$ Hybride
$$C^{(i)} = (Enc_k(m_0^1), \ldots, Enc_k(m_0^i), Enc_k(m_1^{i+1}), \ldots, Enc_k(m_1^t)).$$
- $\mathrm{Ws}[PrivK_{\mathcal{A},\Pi}^{mult-cpa}(n) = 1] = \frac{1}{2} \cdot \mathrm{Ws}[\mathcal{A}(C^{(t)}) = 0] + \frac{1}{2} \cdot \mathrm{Ws}[\mathcal{A}(C^{(0)} = 1]$.
- $\mathcal{A}'$ unterscheidet $Enc_k(m_0^i)$ und $Enc_k(m_1^i)$ für zufälliges $0 \leq i \leq t$.
- Entspricht dem Unterscheiden von $C^{(i)}$ und $C^{(i-1)}$.
- Liefert $\Pr[PrivK_{\mathcal{A},\Pi}^{mult-cpa}(n)] \leq \frac{1}{2} + t \cdot \mathrm{negl}(n)$ $\square_{\text{Satz}}$.

Von fester zu beliebiger Nachrichtenlänge

- Sei Π ein Verschlüsselungsverfahren mit Klartexten aus $\{0,1\}^n$.
- Splitte $m \in \{0,1\}^*$ in $m_1, \ldots m_t$ mit $m_i \in \{0,1\}^n$.
- Definiere Π' vermöge $Enc'_k(m) = Enc_k(m_1) \ldots Enc_k(m_t)$.
- Aus vorigem Satz folgt: Π' ist CPA-sicher, falls Π CPA-sicher ist.

Zufallsfunktionen

Definition Echte Zufallsfunktionen:

Sei $Func_n = \{f \mid f : \{0,1\}^n \to \{0,1\}^n\}$. Wir bezeichnen $f \in_R Func_n$ als *echte Zufallsfunktion* auf n Bits.

Anmerkungen:

- Können $f \in Func_n$ mittels vollständiger Wertetabelle beschreiben.
- Damit kann f als Bitstring der Länge $n \cdot 2^n$ dargestellt werden: n Bits pro $f(x)$ für alle $x \in \{0,1\}^n$.
- Es gibt $2^{n \cdot 2^n}$ Strings dieser Länge $n \cdot 2^n$, d.h. $|Func_n| = 2^{n \cdot 2^n}$.

Definition längenerhaltende, schlüsselabhängige Funktion

Sei F ein pt Algorithmus. F heißt *längenerhaltende, schlüsselabhängige Funktion* falls F eine Fkt. $\{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$ berechnet. Notation: $F_k(x) := F(k,x)$, wobei k der Schlüssel ist.

Anmerkung:

- Zur Übersichtlichkeit der Notation verwenden wir stets $m = n$.

Pseudozufallsfunktion

Definition Pseudozufallsfunktion

Sei F ein längenerhaltende, schlüsselabhängige Funktion. Wir bezeichnen F als *Pseudozufallsfunktion*, falls für alle ppt D gilt

$$\left|\mathrm{Ws}[D^{F_k(\cdot)}(1^n) = 1] - \mathrm{Ws}[D^{f(\cdot)}(1^n) = 1]\right| \leq \mathrm{negl}(n),$$

wobei $k \in_R \{0,1\}^n$ und $f \in_R Func_n$.

Anmerkungen:

- Die Beschreibungslänge von f ist $n2^n$ Bits, d.h. exponentiell in n.
- Daher erhält ein ppt D nicht f, sondern Orakelzugriff auf f und F_k.
- D kann nur polynomiell viele Anfragen an sein Orakel stellen.
- Danach muss D entscheiden, ob sein Orakel einer echten Zufallsfunktion oder einer Pseudozufallsfunktion entspricht.

Existenz und Verschlüsselung mit Pseudozufallsfkt

Fakt Existenz von Pseudozufallsfunktionen

Pseudozufallsfunktionen existieren gdw Pseudozufallsgeneratoren existieren.

$\Rightarrow$: siehe Übung

Algorithmus Verschlüsselung Π_B

Sei F eine längenerhaltende, schlüsselabhängige Funktion auf n Bits. Wir definieren $\Pi_B = (Gen, Enc, Dec)$ für Nachrichten der Länge n.

1. **Gen:** Wähle $k \in_R \{0,1\}^n$.
2. **Enc:** Für $m \in \{0,1\}^n$ wähle $r \in_R \{0,1\}^n$ und berechne
$$c := (r, F_k(r) \oplus m).$$
3. **Dec:** Für $c = (c_1, c_2) \in \{0,1\}^n \times \{0,1\}^n$ berechne
$$m := F_k(c_1) \oplus c_2.$$

Sicherheit von Π_B

Satz Sicherheit von Π_B

Sei F eine Pseudozufallsfunktion. Dann ist Π_B CPA-sicher.

Intuition:

- $F_k(r)$ ist nicht unterscheidbar von n-Bit Zufallsstring.
- D.h. in der zweiten Komponente ist die Verteilung ununterscheidbar von einem One-Time Pad.
- Vorsicht: Benötigen, dass r nicht wiederverwendet wird.

Beweis:

- Sei $\mathcal{A}$ ein CPA-Angreifer mit Vorteil $\epsilon(n)$.
- Konstruieren mittels $\mathcal{A}$ einen Unterscheider D für $F_k(\cdot)$ und $f(\cdot)$.

Unterscheider D

Algorithmus Unterscheider *D*

EINGABE: 1^n, $\mathcal{O} : \{0,1\}^n \leftarrow \{0,1\}^n$ (mit $\mathcal{O} = F_k(\cdot)$ oder $\mathcal{O} = f(\cdot)$)

1. Beantworte Verschlüsselungsanfragen $Enc_k(m)$ von $\mathcal{A}$ wie folgt: Wähle $r \in_R \{0,1\}^n$ und sende $(r, \mathcal{O}(r) \oplus m)$ an $\mathcal{A}$.
2. Beantworte Challenge (m_0, m_1) von $\mathcal{A}$ wie folgt: Wähle $r_c \in_R \{0,1\}^n$, $b \in_R \{0,1\}$ und sende $(r_c, \mathcal{O}(r_c) \oplus m_b)$ an $\mathcal{A}$.
3. Erhalte nach weiteren Verschlüsselsanfragen von $\mathcal{A}$ Bit b'.

$$\text{AUSGABE:} = \begin{cases} 1 & \text{falls } b' = b, \text{ Interpretation: } \mathcal{O} = F_k(\cdot) \\ 0 & \text{sonst,} \qquad\quad \text{Interpretation: } \mathcal{O} = f(\cdot) \end{cases}.$$

Fall 1: $\mathcal{O} = F_k(\cdot)$, d.h. wir verwenden eine Pseudozufallsfunktion.

- Dann ist die Verteilung von $\mathcal{A}$ identisch zu Π_B. Damit gilt

$$\text{Ws}[D^{F_k(\cdot)}(1^n) = 1] = \text{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi_B}(n) = 1] = \tfrac{1}{2} + \epsilon(n).$$

Verwenden einer echten Zufallsfunktion

Fall 2: $\mathcal{O} = f(\cdot)$, d.h. wir verwenden eine echte Zufallsfunktion.

- Sei Π' das Protokoll Π_B unter Verwendung von $f(\cdot)$ statt $F_k(\cdot)$.
- Sei *Repeat* das Ereignis, dass r_c in einer der Verschlüsselungsanfragen verwendet wurde.
- Für alle Angreifer $\mathcal{A}$ gilt $\mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi'}(n) = 1]$

$$\begin{aligned} &= \mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi'}(n) = 1 \cap Repeat] + \mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi'}(n) = 1 \cap \overline{Repeat}] \\ &\leq \mathrm{Ws}[Repeat] + \mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi'}(n) = 1 \mid \overline{Repeat}] \end{aligned}$$

- Ein ppt Angreifer $\mathcal{A}$ stelle insgesamt polynomiell viele Anfragen.
- Sei $q(n)$ die Anzahl der Anfragen. Dann gilt

$$\mathrm{Ws}[Repeat] = 1 - \underbrace{\left(1 - \frac{1}{2^n}\right)^{q(n)}}_{\geq 1 - \frac{q(n)}{2^n}} \leq \frac{q(n)}{2^n} = \mathrm{negl}(n).$$

Fall 2: (Fortsetzung)

- Aufgrund der perfekten Sicherheit des One-Time Pads gilt

$$\mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi'}(n) = 1 \mid \overline{Repeat}] = \tfrac{1}{2}.$$

- Es folgt $\mathrm{Ws}[D^{f(\cdot)}(1^n) = 1] = \mathrm{Ws}[PrivK^{cpa}_{\mathcal{A},\Pi'}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$.

- Aus der Pseudozufälligkeit von F folgt insgesamt

$$\mathrm{negl}(n) \geq \Big| \underbrace{\mathrm{Ws}[D^{F_k(\cdot)}(1^n) = 1]}_{\frac{1}{2}+\epsilon(n)} - \underbrace{\mathrm{Ws}[D^{f(\cdot)}(1^n) = 1]}_{\leq \frac{1}{2}+\mathrm{negl}(n)} \Big|.$$

- Es folgt $\epsilon \leq \mathrm{negl}(n)$ für alle polynomiellen Angreifer $\mathcal{A}$.

Nachrichten beliebiger Länge

Algorithmus Verschlüsselung Π'_B

Sei F eine längenerhaltende, schlüsselabhängige Funktion auf n Bits. Wir definieren $\Pi'_B = (Gen, Enc, Dec)$ für Nachrichten $m \in \{0,1\}^*$.

1. **Gen:** Wähle $k \in_R \{0,1\}^n$.
2. **Enc:** Für $m = m_1 \dots m_\ell$ mit $m_i \in \{0,1\}^n$ wähle $r_1, \dots r_\ell$ mit $r_i \in_R \{0,1\}^n$ und berechne
$$c := (r_1, \dots, r_\ell, F_k(r_1) \oplus m_1, \dots, F_k(r_\ell) \oplus m_\ell).$$
3. **Dec:** Für $c = (c_1, ..., c_{2\ell}) \in (\{0,1\}^n)^{2\ell}$ berechne
$$m := F_k(c_1) \oplus c_{\ell+1} \dots F_k(c_\ell) \oplus F_k(c_{2\ell}).$$

CPA-Sicherheit von Π'_B

Satz CPA-Sicherheit von Π'_B

Sei F eine Pseudozufallsfunktion. Dann ist Π'_B CPA-sicher.

Beweis:

- Aus der CPA-Sicherheit von Π_B folgt die mult-CPA Sicherheit von Π_B und damit die CPA-Sicherheit von Π'_B.

Nachteil: Chiffetexte sind doppelt so lang wie Klartexte (Nachrichtenexpansion 2).

Pseudozufallspermutationen

Definition schlüsselabhängige Permutation

Seien F, F^{-1} pt Algorithmen. F heißt *schlüsselabhängige Permutation* auf n Bits falls

1. F berechnet eine Funktion $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, so dass für alle $k \in \{0,1\}^n$ die Funktion $F_k(\cdot)$ eine Bijektion ist.
2. $F_k^{-1}(\cdot)$ berechnet die Umkehrfunktion von $F_k(\cdot)$.

Definition Pseudozufallspermutation

Sei F eine schlüsselabhängige Permutation auf n Bits. Wir bezeichnen F als *Pseudozufallspermutation*, falls für alle ppt D gilt

$$\left|\mathrm{Ws}[D^{F_k(\cdot)}(1^n) = 1] - \mathrm{Ws}[D^{f(\cdot)}(1^n) = 1]\right| \leq \mathrm{negl}(n),$$

mit $k \in_R \{0,1\}^n$ und $f \in_R Perm_n$, wobei $Perm_n$ die Menge aller Permutationen auf n Bits ist.

Starke Pseudozufallspermutationen

Satz Pseudozufallspermutationen und Pseudozufallsfunktionen

Jede Pseudozufallspermutation ist eine Pseudozufallsfunktion.

Beweis: Übung.

Definition Starke Pseudozufallspermutation (Blockchiffre)

Sei F eine schlüsselabhängige Permutation auf n Bits. Wir bezeichnen F als *starke Pseudozufallspermutation (Blockchiffre)*, falls für alle ppt D gilt

$$\left|\mathrm{Ws}[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n)=1]-\mathrm{Ws}[D^{f(\cdot),f^{-1}(\cdot)}(1^n)=1]\right|\leq \mathrm{negl}(n),$$

mit $k \in_R \{0,1\}^n$ und $f \in_R Perm_n$.

Blockchiffren als kryptographische Primitive

Anmerkungen: Blockchiffren

- Praktische Realisierungen von starken Pseudozufalls-permutationen bezeichnet man als *Blockchiffren*.
- Wir haben gesehen, dass Blockchiffren $F_k(\cdot)$ zur Konstruktion CPA-sicherer Verschlüsselung verwendet werden können.
- Vorsicht: Blockchiffren selbst sind keine sicheren Verschlüs-selungsverfahren.
- $c := F_k(m)$ ist eine deterministische, unsichere Verschlüsselung.
- D.h. wir benötigen einen Randomisierungsprozess bei Enc.

Bsp: DES (Data Encryption Standard, 1976)

- $F : \{0,1\}^{56} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$
- Problem des zu kleinen Schlüsselraums
- bester bekannter KPA Angriff mit 2^{43} Klartexten

AES (Advanced Encryption Standard, 2002)

- $F\{0,1\}^k \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ mit $k \in \{128, 192, 256\}$
- Derzeit kein erfolgreicher Angriff bekannt.

Modes of Operation – Electronic Code Book (ECB)

Ziel: Verschlüsseln von Nachrichten $m = m_1 \dots m_\ell \in (\{0,1\}^n)^\ell$ mittels Blockchiffre unter Verwendung kleiner Nachrichtenexpansion.

Algorithmus Electronic Code Book (ECB) Modus

1. **Enc:** $c := (F_k(m_1), \dots, F_k(m_\ell))$
2. **Dec:** $m := F_k^{-1}(c_1), \dots, F_k^{-1}(m_\ell)$

Nachteil:

- Enc ist deterministisch, d.h. ECB ist nicht mult-KPA sicher.
- Daher sollte der ECB Modus nie verwendet werden.

Modes of Operation – Cipher Block Chaining (CBC)

Algorithmus Cipher Block Chaining (CBC) Modus

1. **Enc:** Wähle Initialisierungsvektor $c_0 := IV \in_R \{0,1\}^n$. Berechne
$$c_i := F_k(c_{i-1} \oplus m_i) \quad \text{für } i = 1, \ldots, \ell.$$
2. **Dec:** Für $c = (c_0, c_1, \ldots, c_\ell)$ berechne
$$m_i := F_k^{-1}(c_i) \oplus c_{i-1} \quad \text{für } i = 1, \ldots, \ell.$$

Vorteile:

- CPA-Sicherheit von CBC kann gezeigt werden.
- Nachrichtenexpansion ist $\frac{\ell+1}{\ell}$.

Nachteil:

- Verschlüsselung muss sequentiell durchgeführt werden.

Modes of Operation – Output Feedback (OFB)

Algorithmus Output Feedback (OFB) Modus

1. **Enc:** Wähle $r_0 := IV \in_R \{0,1\}^n$. Berechne $r_i := F_k(r_{i-1})$ für $i \in [\ell]$,
$$c_i := r_i \oplus m_i \quad \text{für } i = 1, \ldots, \ell.$$
2. **Dec:** Für $c = (IV, c_1, \ldots, c_\ell)$ berechne $r_i := F_k(r_{i-1})$ für $i \in [\ell]$,
$$m_i := c_i \oplus r_i \quad \text{für } i = 1, \ldots, \ell.$$

Vorteile:

- CPA-Sicherheit von OFB kann gezeigt werden.
- Nachrichtenexpansion ist $\frac{\ell+1}{\ell}$.
- Pseudozufallsfunktion F_k genügt, Invertierbarkeit nicht nötig.
- Die Berechnung der Zufallspads r_i kann unabhängig von der Nachricht nur mittels IV durchgeführt werden.

Nachteil:

- Berechnung der Zufallspads r_i ist sequentiell.

Modes of Operation – Counter (CTR)

Algorithmus Counter (CTR) Modus

1. **Enc:** Wähle $ctr := IV \in_R \{0,1\}^n$.
   Berechne $r_i := F_k(ctr + i \bmod 2^n)$ für $i \in [\ell]$ und
   $$c_i := r_i \oplus m_i \quad \text{für } i = 1, \ldots, \ell.$$
2. **Dec:** Für $c = (IV, c_1, \ldots, c_\ell)$ berechne $r_i := F_k(r_{i-1})$ für $i \in [\ell]$,
   $$m_i := c_i \oplus r_i \quad \text{für } i = 1, \ldots, \ell.$$

Vorteile:

- CPA-Sicherheit von CTR kann gezeigt werden.
- Nachrichtenexpansion ist $\frac{\ell+1}{\ell}$.
- Pseudozufallsfunktion F_k genügt, Invertierbarkeit nicht nötig.
- Berechnung der Zufallspads r_i unabhängig von der Nachricht.
- Ver-/Entschlüsselung sind vollständig parallelisierbar.

Sicherheit des Counter Modes

Satz Sicherheit des CTR Modes

Sei F eine Pseuozufallsfunktion. Dann ist der CTR Modus CPA-sicher.

Beweisskizze:

- Können Unterscheider D mittels CPA-Angreifers $\mathcal{A}$ konstruieren.
- Beweis verläuft analog zum Beweis der CPA-Sicherheit von Π_B.
- Pseudozufälliges Pad $F_k(r)$ darf nicht wiederverwendet werden.
- Hier verbraucht aber jede $Enc(\cdot)$-Anfrage einen Block von Pads.

Sicherheit des Counter Modes

Beweisskizze: Fortsetzung

- Sei $q(n)$ eine polynomielle obere Schranke sowohl für
  - die Anzahl der Anfragen von $\mathcal{A}$ an das Orakel $Enc(\cdot)$ als auch für
  - die Anzahl der zu verschlüsselnden Blöcke.
- D.h. $Enc(m)$-Anfragen erfolgen für $m \in (\{0,1\}^n)^\ell$ mit $\ell \leq q(n)$.
- Jede solche Anfrage verbraucht ein Intervall $ctr \ldots ctr + \ell - 1$ von Pads $F_k(ctr), \ldots, F_k(ctr + \ell - 1)$ der Länge höchstens $q(n)$.
- Sei I das Intervall zum Verschlüsseln der Challenge m_b.
- Sei I_j das Intervall aus der j-ten $Enc(\cdot)$-Anfrage für $j = 1, \ldots, q(n)$.
- $Ws[PrivK^{cpa}_{\mathcal{A},\Pi_{ctr}}(n) = 1] = 1$, falls sich I mit einem I_j überschneidet.
- $\mathrm{Ws}[I \text{ überschneidet sich mit } I_j] \leq \frac{2q(n)}{2^n}$ für jedes feste j.
- Damit $\mathrm{Ws}[I \text{ überschneidet sich mit einem der } I_j] \leq \frac{2q^2(n)}{2^n} = \mathrm{negl}(n)$.

Blocklänge und Sicherheit

Anmerkung: Wahl der Blocklänge

- Vorige Beweisskizze zeigt Angriff mittels Intervallüberschneidung.
- Erreichen Erfolgsws der Größenordnung $\frac{q(n)^2}{2^n}$ für Blocklänge n.
- D.h. wir erhalten einen generischen Angriff für Blockchiffren mit Hilfe von $q(n) = 2^{\frac{n}{2}}$ Anfragen von Nachrichten $m \in (\{0,1\}^n)^{q(n)}$.
- Damit muss nicht nur die Schlüssellänge einer Blockchiffre hinreichend groß gewählt werden, sondern auch die Blocklänge n.

CCA-Spiel

Szenario: CCA-Sicherheit

- $\mathcal{A}$ erhält im *PrivK*-Spiel Zugriff auf Orakel $Enc_k(\cdot)$ und $Dec_k(\cdot)$.

Spiel CCA Ununterscheidbarkeit von Chiffretexten $PrivK^{cca}_{\mathcal{A},\Pi}(n)$

Sei Π ein Verschlüsselungsverfahren und $\mathcal{A}$ ein Angreifer.

1. $k \leftarrow Gen(1^n)$.
2. $(m_0, m_1) \leftarrow \mathcal{A}^{Enc_k(\cdot),Dec_k(\cdot)}(1^n)$, d.h. $\mathcal{A}$ darf $Enc_k(m)$ und $Dec_k(c')$ für beliebige m und c' anfragen.
3. Wähle $b \in_R \{0,1\}$ und verschlüssele $c \leftarrow Enc_k(m_b)$.
4. $b' \leftarrow \mathcal{A}^{Enc_k(\cdot),Dec_k(\cdot)}(c')$, d.h. $\mathcal{A}$ darf $Enc_k(m)$ und $Dec_k(c')$ für beliebige m und $c' \neq c$ anfragen.
5. $PrivK^{cca}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{für } b = b' \\ 0 & \text{sonst} \end{cases}$.

Anmerkung:

- Ohne die Einschränkung $c' \neq c$ kann $\mathcal{A}$ das Spiel stets gewinnen.

CCA Sicherheit

Definition CCA Sicherheit

Ein Verschlüsselungsschema $\Pi = (Gen, Enc, Dec)$ besitzt *ununterscheidbare Chiffretexte gegenüber CCA* falls für alle ppt $\mathcal{A}$:

$$\mathrm{Ws}[PrivK^{cca}_{\mathcal{A},\Pi}(n) = 1] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

Der Wsraum ist definiert über die Münzwürfe von $\mathcal{A}$ und $PrivK^{cca}_{\mathcal{A},\Pi}$.

Notation: Wir bezeichnen Π als *CCA-sicher*.

CCA-Unsicherheit von Π_B

Beobachtung: Π_B ist nicht CCA-sicher.

Algorithmus CCA-Angreifer $\mathcal{A}$ für Π_B

EINGABE: 1^n, Zugriff auf Orakel $Enc(\cdot)$ und $Dec(\cdot)$

1. Wähle $(m_0, m_1) = (0^n, 1^n)$
2. Erhalte Chiffretext $c = (c_1, c_2) = (r, F_k(r) \oplus m_b)$.
3. Berechne $c' = (c_1, c_2 \oplus 1^n)$. Sei m' die Antwort auf die Entschlüsselungsanfrage $Dec_k(c')$.

AUSGABE: $b' = \begin{cases} 0 & \text{für } m' = 1^n \\ 1 & \text{sonst} \end{cases}$.

Es gilt $\mathrm{Ws}[PrivK^{cca}_{\mathcal{A},\Pi_B}(n) = 1] = 1$.

Ziel: Werden CCA-sichere Verschlüsselung mittels sogenannter MACs (Message Authentication Codes) konstruieren.

Integrität und Authentizität von Nachrichten

1. **Integrität:** Überprüfen, dass eine Nachricht nicht verändert wurde.
2. **Authentizität:** Überprüfen, dass eine Nachricht wirklich vom Absender kommt.

Ziel: Können Angriffe nicht verhindern, müssen sie aber erkennen.

Anm.: Verschlüsselung liefert weder Integrität noch Authentizität.

- Bsp: Verwenden unsere CPA-sichere Verschlüsselung Π_B.
- Chiffretexte besitzen Form $c = (c_1, c_2) = (r, F_k(r) \oplus m) \in \{0,1\}^{2n}$.
- Sei e_i ein Einheitsvektor der Länge n. Dann ist $c' = (c_1, c_2 \oplus e_i)$ eine gültige Verschlüsselung von $m' = m + e_i$.
- D.h. $\mathcal{A}$ kann beliebige Bits von m verändern, ohne m zu kennen.

- Jedes $c = (c_1, c_2)$ ist eine Verschlüsselung von $m = F_k(c_1) \oplus c_2$.
- D.h. $\mathcal{A}$ kann ein gültiges c erzeugen, ohne k zu kennen.

Message Authentication Code (MAC)

Szenario: Integrität und Authentizität mittels MACs.

- Alice und Bob besitzen gemeinsamen Schlüssel k.
- Alice berechnet für m einen MAC-Tag t als Funktion von m und k.
- Alice sendet das Tupel (m, t) an Bob.
- Bob verifiziert, dass t ein gültiger Tag für m ist.

Definition Message Authentication Code (MAC)

Ein *Message Authentication Code (MAC)* besteht aus den ppt Alg.

1. **Gen:** $k \leftarrow Gen(1^n)$
2. **Mac:** Bei Eingabe von k und $m \in \{0,1\}^*$ berechne $t \leftarrow Mac_k(m)$.
3. **Vrfy:** Bei Eingabe (m, t) berechne

$$Vrfy(m,t) = \begin{cases} 1 & \text{falls } t \text{ ein gültiger MAC für } m \text{ ist} \\ 0 & \text{sonst} \end{cases}.$$

Es gilt $Vrfy(m, Mac_k(m)) = 1$ für alle m und k.

Sicherheitsspiel Mac-forge

Spiel Sicherheit von MACs Mac-forge

Sei Π ein MAC mit Sicherheitsparameter n und Angreifer $\mathcal{A}$.

1. $k \leftarrow Gen(1^n)$
2. $(m,t) \leftarrow \mathcal{A}^{Mac_k(\cdot)}(1^n)$. Sei Q die Menge aller $Mac_k(\cdot)$-Anfragen von $\mathcal{A}$ an sein Orakel.
3. $$\text{Mac-forge}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{falls } Vrfy(m,t) = 1 \text{ und } m \notin Q \\ 0 & \text{sonst} \end{cases}.$$

Definition Sicherheit eines MACs

Ein MAC Π heißt *existentiell unfälschbar gegenüber adaptiv gewählten Angriffen* bzw. kurz *sicher* falls für alle ppt Angreifer $\mathcal{A}$ gilt

$$\text{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

Replay Angriffe

Szenario: Replay Angriff

- Alice schickt an ihre Bank eine authentisierte Zahlungsanweisung (m, t) über 100 Euro zugunsten Bob's Konto.
- Aufgrund der MAC-Sicherheit kann Bob den Betrag nicht ändern.
- Die MAC-Sicherheit verhindert nicht, dass Bob (m, t) abfängt und dieselbe Nachricht (m, t) weitere Male an die Bank versenden.
- Abhilfe: Verwenden Nummerierung oder Zeitstempel.

Seriennummer:

- Berechnen MAC von $i||m$ für eindeutige i.
- MAC-Sicherheit: $\mathcal{A}$ kann nicht MAC für $i'||m$ berechnen.

Zeitstempel:

- Sender berechnet MAC von *Systemzeit*$||m$.
- Empfänger verifiziert, dass die *Systemzeit* aktuell ist.

Konstruktion eines sicheren MACs fester Länge

Algorithmus MAC Π_{MAC} fester Länge

Sei F eine Pseudozufallsfunktion mit Blocklänge n. Wir konstruieren einen MAC für Nachrichten $m \in \{0,1\}^n$.

1. **Gen:** Wähle $k \in_R \{0,1\}^n$.
2. **Mac:** Für $m, k \in \{0,1\}^n$ berechne $t := F_k(m)$.
3. **Vrfy:** Für $(m,t) \in \{0,1\}^n \times \{0,1\}^n$ und $k \in \{0,1\}^n$

$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } t = F_k(m) \\ 0 & \text{sonst} \end{cases}.$$

Sicherheit von Π_{MAC}

Satz Sicherheit von Π_{MAC}

Sei F eine Pseudozufallsfunktion. Dann ist Π_{MAC} sicher.

Beweis:

- Sei $\mathcal{A}$ ein Angreifer für Π_{MAC} mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren Unterscheider U für Pseudozufallsfunktionen.

Algorithmus Unterscheider U

EINGABE: 1^n, $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$ mit $\mathcal{O} = F_k(\cdot)$ oder $\mathcal{O} = f(\cdot)$.

1. $(m,t) \leftarrow \mathcal{A}^{Mac(\cdot)}$, beantworte $Mac(m')$-Anfragen mit $t' := \mathcal{O}(m')$.
2. Sei Q die Menge aller von $\mathcal{A}$ gestellten $Mac(\cdot)$-Anfragen.

$$\text{AUSGABE} = \begin{cases} 1 & \text{falls } t' = \mathcal{O}(m'), m' \notin Q, \text{ Interpretation: } \mathcal{O} = F_k(\cdot) \\ 0 & \text{sonst,} \qquad\qquad\qquad\quad \text{Interpretation: } \mathcal{O} = f(\cdot) \end{cases}.$$

Sicherheit von Π_{MAC}

Fall 1: $\mathcal{O} = F_k(\cdot)$, d.h. das Orakel ist eine Pseudozufallsfunktion.

- Dann ist die Verteilung für $\mathcal{A}$ identisch zum Protokoll Π_{MAC}.
- Damit gilt

$$\mathrm{Ws}[U^{F_k(\cdot)}(n) = 1] = \mathrm{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi_{MAC}}(n) = 1] = \epsilon(n).$$

Fall 2: $\mathcal{O} = f(\cdot)$, d.h. das Orakel ist eine echte Zufallsfunktion.

- Sei Π' das Protokoll Π_{MAC} mit $f(\cdot)$ statt $F_k(\cdot)$.
- Für alle $m \notin Q$ ist $t = f(m)$ uniform verteilt in $\{0,1\}^n$.
- Damit gilt

$$\mathrm{Ws}[U^{f(\cdot)}(n) = 1] = \mathrm{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi'}(n) = 1] = \tfrac{1}{2^n}.$$

Aus der Pseudozufälligkeit von F_k folgt für alle ppt U

$$\mathrm{negl}(n) \geq \left|\mathrm{Ws}[U^{F_k(\cdot)}(n) = 1] - \mathrm{Ws}[U^{f(\cdot)}(n) = 1]\right| = |\epsilon - \tfrac{1}{2^n}|.$$

Damit folgt $\epsilon \leq \mathrm{negl}(n) + \frac{1}{2^n} = \mathrm{negl}(n)$ für alle ppt Angreifer $\mathcal{A}$.

Von fester zu variabler Länge

Ziel: Konstruiere MAC für $m = m_1 \dots m_\ell$ für variable Blockzahl ℓ.

Überlegungen zu einer sicheren MAC-Konstruktion:

- **MAC des XOR der Blocks**, d.h. $t := Mac_k(\bigoplus_{i=1}^{\ell} m_i)$.
- Problem: Tag t ist z.B. gültig für $\overline{m_1 m_2} m_3 \dots m_\ell$.

- **MAC jeden Blocks**, d.h. $t = t_1 \dots t_\ell$ für $t_i := Mac_k(m_i)$.
- Problem: $t' = t_2 t_1 t_3 \dots t_\ell$ ist gültig für $m' = m_2 m_1 m_3 \dots m_\ell$.

- **MAC mit Block-Seriennummer**, d.h. $t_i := Mac_k(i||m_i)$.
- Problem: $t' = t_1 \dots t_{\ell-1}$ ist gültig für $m' = m_1 \dots m_{\ell-1}$.

- **MAC mit Nachrichtenlänge**, d.h. $t_i := Mac_k(\ell||i||m_i)$.
- Problem: Seien $t = t_1 \dots t_\ell$, $t' = t'_1 \dots t'_\ell$ gültig für m, m'. Dann ist $t'_1 t_2 \dots t_\ell$ gültig für $m'_1 m_2 \dots m_\ell$, d.h. wir können Tags kombinieren.

- **MAC mit Nachrichten-Seriennummer**: $t_i := Mac_k(r||\ell||i||m_i)$.
- Wir benötigen pro Nachricht m einen eindeutigen Identifikator r.

Sicherer MAC für Nachrichten variabler Länge

Algorithmus MAC Π_{MAC2} variabler Länge

Sei $\Pi' = (Gen', Mac', Vrfy')$ ein MAC für Nachrichten der Länge n.

1. **Gen:** $k \leftarrow Gen(1^n)$
2. **Mac:** Sei $k \in \{0,1\}^n$ und $m = m_1 \dots m_\ell \in (\{0,1\}^{\frac{n}{4}})^\ell$.
   Wähle $r \in_R \{0,1\}^{\frac{n}{4}}$ und berechne
   $$t_i \leftarrow Mac'_k(r||\ell||i||m_i) \text{ für } i = 1, \dots, \ell,$$
   mit Kodierungen $\ell, i \in \{0,1\}^{\frac{n}{4}}$. Ausgabe des Tags $t = (r, t_1 \dots t_\ell)$.
3. **Vrfy:** Für $(m,t) = (m_1 \dots m_\ell, r, t_1, \dots, t_\ell)$
   $$\text{Ausgabe} = \begin{cases} 1 & \text{falls } Vrfy'_k(r||\ell||i||m_i, t_i) = 1 \text{ für } i = 1, \dots, \ell \\ 0 & \text{sonst} \end{cases}.$$

Anmerkung:

- Benötigen $\ell < 2^{\frac{n}{4}}$, sonst kann ℓ nicht mit $\frac{n}{4}$ Bits kodiert werden.

Sicherheit von Π_{MAC2}

Satz Sicherheit von Π_{MAC2}

Sei Π' sicher. Dann ist Π_{MAC2} ebenfalls sicher.

Beweis:

- Sei $\mathcal{A}$ ein Angreifer für Π_{MAC2} mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren einen Angreifer $\mathcal{A}'$ für Π'.

Algorithmus Angreifer $\mathcal{A}'$

EINGABE: 1^n, Orakel $Mac'_k(\cdot)$.

1. Beantworte $Mac_k(m'_1 \dots m'_{\ell'})$-Anfragen von $\mathcal{A}$ wie folgt: Wähle $r' \in_R \{0,1\}^{\frac{n}{4}}$ und berechne $t'_i = Mac'_k(r'||\ell'||i||m'_i)$ für $i = 1, \dots, \ell'$.
2. $(m,t) = (m_1 \dots m_\ell, r, t_1 \dots t_\ell) \leftarrow \mathcal{A}^{Mac_k(\cdot)}(1^n)$.

AUSGABE: Nicht-angefragtes $r||\ell||i||m_i$ mit gültigem Tag t_i, falls ein solches in (m,t) existiert.

Sicherheit von Π_{MAC2}

Wir definieren die folgenden Ereignisse

- *Forge*: Ein Block $r||\ell||i||m_i$ in (m,t) wurde nicht angefragt.
- *Repeat*: Bei 2 MAC-Anfragen wird dasselbe $r_i = r_j$ verwendet.

Aufgrund der Sicherheit von Π' gilt

$$\begin{aligned}
\text{negl}(n) &\geq \text{Ws}[\text{Mac-forge}_{\mathcal{A}',\Pi'}(n) = 1] \\
&= \text{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi_{MAC2}}(n) = 1 \cap \mathit{Forge}] \\
&= \epsilon(n) - \text{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi_{MAC2}}(n) = 1 \cap \overline{\mathit{Forge}}] \\
&= \epsilon(n) - \underbrace{\text{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi_{MAC2}}(n) = 1 \cap \overline{\mathit{Forge}} \cap \overline{\mathit{Repeat}}]}_{\epsilon_1} \\
&\qquad - \underbrace{\text{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi_{MAC2}}(n) = 1 \cap \overline{\mathit{Forge}} \cap \mathit{Repeat}]}_{\leq \text{Ws}[\mathit{Repeat}]}
\end{aligned}$$

Zeigen nun $\epsilon_1 = 0$ und $\text{Ws}[\mathit{Repeat}] \leq \text{negl}(n)$. Damit $\epsilon(n) \leq \text{negl}(n)$.

Sicherheit von Π_{MAC2}

zu zeigen: $\mathrm{Ws}[\mathit{Repeat}] \leq \mathrm{negl}(n)$

- Sei $q(n)$ die Anzahl der MAC-Anfragen von $\mathcal{A}$ an $\mathcal{A}'$.
- Bei der i-ten MAC-Anfrage wähle $\mathcal{A}'$ den Identifikator r_i.
- *Repeat* tritt ein, falls $r_i = r_j$ für ein $i \neq j$. Sei dies Ereignis $E_{i,j}$.
- Nach Geburtstagsparadoxon gilt:

$$\begin{aligned} \mathrm{Ws}[\mathit{Repeat}] = \mathrm{Ws}\Big[\bigcup_{1 \leq i < j \leq q(n)} E_{i,j}\Big] &\leq \sum_{1 \leq i < j \leq q(n)} \mathrm{Ws}[E_{i,j}] \\ &\leq \frac{q(n)^2}{2^{\frac{n}{4}}} = \mathrm{negl}(n). \end{aligned}$$

Sicherheit von Π_{MAC2}

zu zeigen: $\text{Ws}[\text{Mac-forge}_{\mathcal{A},\Pi_{MAC2}}(n) = 1 \cap \overline{Forge} \cap \overline{Repeat}] = 0$

- **Idee:** $\text{Mac-forge}_{\mathcal{A},\Pi_{MAC2}}(n) = 1$ und $\overline{Repeat}$ impliziert $Forge$.
- Sei $(m,t) = (m_1 \ldots m_\ell, r, t_1 \ldots t_\ell)$ die Ausgabe von $\mathcal{A}$.

Fall 1: Identifikator r unterscheidet sich von allen r_i.

- Dann ist $r||\ell||1||m_1$ nicht-angefragt mit gültigem Tag t_1.

Fall 2: $r = r_i$ für genau ein $i \in [q(n)]$.

- Sei $m^{(i)} = m'_1 \ldots m'_{\ell'}$ die von $\mathcal{A}$ angefragte Nachricht.
- Fall $\ell \neq \ell'$: Dann ist $r||\ell||1||m_1$ nicht-angefragt mit gültigem Tag t_1.
- Fall $\ell = \ell'$: Wegen $m \notin Q$ gilt $m \neq m^{(i)}$.
- D.h. es existiert ein j, so dass $m_j \neq m'_j$.
- Damit wurde $r||\ell||j||m_j$ nicht angefragt. Tag t_j ist dafür gültig.

Fall 3: $r = r_i$ für mehrere $i \in [q(n)]$.

- Fall ist ausgeschlossen, da wegen $\overline{Repeat}$ gilt $r_i \neq r_j$ für alle $i \neq j$.

Sicherer MAC für Nachrichten beliebiger Länge

Korollar Sicherer MAC für Nachrichten beliebiger Länge

Sei F eine Pseudozufallsfunktion. Dann ist Π_{MAC2} für $\Pi' = \Pi_{MAC}$ sicher.

Nachteile:

- Für $m \in (\{0,1\}^{\frac{n}{4}})^{\ell}$ sind ℓ Anwendungen von F_k erforderlich.
- Der MAC-Tag $t = (r, t_1 \ldots t_{\ell})$ besitzt Länge $(\ell + 1)n$ Bits.

CBC-MAC für Nachrichten fester Länger

Algorithmus CBC-MAC für Nachrichten fester Länger

Sei F eine Pseudozufallsfunktion und $m \in (\{0,1\}^n)^\ell$ für festes ℓ.

1. **Gen:** Wähle $k \in_R \{0,1\}^n$.
2. **Mac:** Für $k \in \{0,1\}^n$ und $m = m_1 \dots m_\ell \in (\{0,1\}^n)^\ell$ setze $t_0 = 0^n$,
$$t_i := F_k(t_{i-1} \oplus m_i) \text{ für } i = 1, \dots, \ell.$$
Ausgabe des Tags $t := t_\ell$.
3. **Vrfy:** Für $k \in \{0,1\}^n$ und $(m,t) \in (\{0,1\}^n)^\ell \times \{0,1\}^n$,
$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } Mac_k(m) = t \\ 0 & \text{sonst} \end{cases}.$$

Anmerkungen:

- Für den CBC-MAC kann Sicherheit für festes ℓ gezeigt werden.
- Tags besitzen nur Länge n, unabhängig von ℓ.
- Konstruktion ist *unsicher* für variables ℓ (Übung).

Vergleich mit CBC Modus bei Verschlüsselung

Rolle des Initialisierungsvektors

- Benötigen zur Sicherheit beim CBC Modus $IV \in_R \{0,1\}^n$.
- Der CBC-MAC verwendet einen festen Wert $IV = t_0 = 0^n$.
- Verwendet man ein zufälliges $t_0 \in_R \{0,1\}^n$ und gibt (t_0, t_ℓ) als Tag aus, so ist dies eine unsichere MAC-Konstruktion! (Übung)

Ausgabe

- CBC Modus: Ausgabe aller c_i, um entschlüsseln zu können.
- Beim CBC-MAC wird nur t_ℓ ausgegeben. Die Werte $t_1, \ldots, t_{\ell-1}$ kann der Verifizierer selbst bestimmen, die MAC-Länge ist nur n.
- Werden $t_1, \ldots, t_{\ell-1}$ ausgegeben, so ist der MAC unsicher! (Übung)

Man beachte:
Scheinbar harmlose Änderungen können drastische Effekte haben.

Sichere CBC-MACs für Nachrichten variabler Länge

Erzeugen längenabhängiger Schlüssel:

- Berechne $k_\ell := F_k(\ell)$ als Schlüssel für Nachrichten der Länge ℓ.
- Verwende k_ℓ als Schlüssel in CBC, d.h. $t_i := F_{k_\ell}(t_{i-1} \oplus m_i)$.
- Wir erhalten einen eigenen Schlüssel k_ℓ für jede feste Länge ℓ.

Anhängen der Länge:

- Verwende $t_0 := F_k(\ell)$ als Initialisierungsvektor.
- Dies ist äquivalent zum Berechnen des Tags von $\ell||m_1 \ldots m_\ell$.
- Man beachte: Berechnen des Tags $m_1 \ldots m_\ell||\ell$ ist unsicher.

Verwenden zweier Schlüssel:

- Wähle $k_1, k_2 \in_R \{0,1\}^n$. Berechne den Tag $t = F_{k_2}(Mac_{k_1}(m))$.
- Alternativ: Wähle $k_1 := F_k(1)$ und $k_2 := F_k(2)$.
- Vorteil: Mac-Berechnung möglich, ohne ℓ a priori zu kennen.

Hashfunktionen und Kollisionen

Definition Hashfunktion

Eine *Hashfunktion* ist ein Paar (Gen, H) von pt Algorithmen mit

1. **Gen:** $s \leftarrow Gen(1^n)$. *Gen* ist probabilistisch.
2. **H:** H_s berechnet Funktion $\{0,1\}^* \rightarrow \{0,1\}^\ell$. H_s ist deterministisch.

Spiel $HashColl_{\mathcal{A},\Pi}(n)$

1. $s \leftarrow Gen(1^n)$
2. $(x, x') \leftarrow \mathcal{A}(s)$
3. $$HashColl_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{falls } H_s(x) = H_s(x') \text{ und } x \neq x' \\ 0 & \text{sonst} \end{cases}.$$

Definition Kollisionsresistenz

Eine Hashfunktion Π heißt *kollisionsresistent*, falls für alle ppt $\mathcal{A}$ gilt $\mathrm{Ws}[HashColl_{\mathcal{A},\Pi}(n) = 1] \leq \mathrm{negl}(n)$.

Schwächere Sicherheitskonzepte

2.Urbild Resistenz

Gegeben: s, x
Gesucht: $x' \neq x$ mit $H_s(x') = H_s(x)$

Satz Kollisionresistenz impliziert 2.Urbild Resistenz

Sei Π kollisionsresistent. Dann ist Π 2.Urbild resistent.

Beweis:

- Sei $\mathcal{A}$ ein 2.Urbild Angreifer auf $\Pi = (Gen, H)$ mit Erfolgsws $\epsilon(n)$.

Algorithmus Angreifer $\mathcal{A}'$ auf Kollisionsresistenz

EINGABE: s

1. Wähle $x \in \{0,1\}^*$.
2. $x' \leftarrow \mathcal{A}(s, x)$

AUSGABE: x, x'

- Offenbar gilt $\mathrm{Ws}[HashColl_{\mathcal{A}',\Pi}] = \epsilon(n)$

Schwächere Sicherheitskonzepte

Urbild Resistenz

Gegeben: $s, y = H_s(x)$
Gesucht: x' mit $H_s(x') = y$

Satz 2.Urbild Resistenz impliziert Urbild Resistenz

Sei Π 2.Urbild resistent und komprimierend. Dann ist Π Urbild resistent

Beweisskizze: Sei $\mathcal{A}$ ein Urbild Angreifer auf Π mit Erfolgsws ϵ.

Algorithmus Angreifer $\mathcal{A}'$ auf 2.Urbild

EINGABE: s, x

1. Berechne $y = H_s(x)$.
2. $x' \leftarrow \mathcal{A}(s, y)$

AUSGABE: x, x' falls $x \neq x'$

- Es gilt $x \neq x'$ mit signifikanter Ws, falls H seine Eingabe komprimiert, d.h. der Urbildraum ist größer als der Bildraum.

Damit ist Kollisionsresistenz der *stärkste Sicherheitsbegriff*.

Geburtstagsangriff auf Hashfunktionen

Algorithmus Geburtstagsangriff

EINGABE: s mit $H_s : \{0,1\}^* \to \{0,1\}^\ell$

1. Wähle verschiedene $x_1, \ldots, x_q \in \{0,1\}^*$ für geeignetes q.
2. Berechne $y_i = H_s(x_i)$ für $i = 1, \ldots, q$ und sortiere die y_i.
3. Finde in der sortierten Liste x_i, x_j mit $y_i = y_j$.

AUSGABE: x_i, x_j mit $H_s(x_i) = H_s(x_j)$

Anmerkungen:

- Annahme: y_i sind zufällig gleichverteilt in $\{0,1\}^\ell$.
- Geburtstagsproblem: Für $q = 2^{\frac{\ell}{2}} + 1$ erhalten wir mit Ws mind $1 - e^{-\frac{1}{2}}$ eine Kollision $y_i = y_j$ in Schritt 3. (Übung)
- Die Auswertung von H_s koste konstante Laufzeit $\mathcal{O}(1)$.
- Dann besitzt der Algorithmus Laufzeit $\mathcal{O}(q \log q) = \mathcal{O}(\ell \cdot 2^{\frac{\ell}{2}})$.

Konsequenz für Hashfunktionen:

- Wir benötigen mindestens Ausgabelänge $\ell = 128$ Bit.

Merkle-Damgard Transformation

Ziel: Konstruiere $H : \{0,1\}^* \to \{0,1\}^\ell$ aus $h : \{0,1\}^{2\ell} \to \{0,1\}^\ell$.

Algorithmus Merkle-Damgard Konstruktion

Sei (Gen, h) eine kollisionsresistente Hashfunktion mit $h : \{0,1\}^{2\ell} \to \{0,1\}^\ell$. Wir konstruieren (Gen, H) wie folgt.

1. **Gen:** $s \leftarrow Gen(1^n)$.
2. **H:** Bei Eingabe s und $x \in \{0,1\}^L$:
   - Erweitere x mit Nullen bis die Länge ein Vielfaches von ℓ ist.
   - Schreibe $x = x_1 \ldots x_B$ mit $x_i \in \{0,1\}^\ell$ und $B = \lceil \frac{L}{\ell} \rceil$.
   - Setze $x_{B+1} = L$ (binär kodiert). Initialisiere $z_0 := 0^\ell$, berechne
     $$z_i := h_s(z_{i-1} || x_i) \text{ für } i = 1, \ldots, B+1.$$

   Ausgabe des Hashwerts $H_s(x) := z_{B+1}$.

Sicherheit der Merkle-Damgard Konstruktion

Satz Sicherheit der Merkle-Damgard Konstruktion

Sei (Gen, h) kollisionsresistent. Dann ist auch (Gen, H) kollisionsresistent.

Beweis:

- Sei $\mathcal{A}$ ein Angreifer für H_s mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren einen Angreifer $\mathcal{A}'$ für h_s.

Algorithmus Angreifer $\mathcal{A}'$

EINGABE: s

1. $(x, x') \leftarrow \mathcal{A}(s)$. Sei $x \in \{0,1\}^L$ und $x' \in \{0,1\}^{L'}$. Seien $x_1 \ldots x_B$ und $x'_1 \ldots x'_{B'}$ die mit Nullen erweiterte Darstellung von x, x'.
2. Sei i maximal mit $z_{i-1}||x_i \neq z'_{i-1}||x'_i$ (existiert wegen $x \neq x'$).

   Setze $y := z_{i-1}||x_i$ und $y' := z'_{i-1}||x'_i$.

AUSGABE: (y, y') mit $h_s(y) = h_s(y')$

Sicherheit der Merkle-Damgard Konstruktion

Korrektheit: Zeigen $h_s(y) = h_s(y')$ für $y \neq y'$. Damit folgt

$$\text{negl}(n) \geq \text{Ws}[\mathit{HashColl}_{\mathcal{A}',\Pi_h}(n) = 1] = \text{Ws}[\mathit{HashColl}_{\mathcal{A},\Pi_H}(n) = 1] = \epsilon(n).$$

Fall 1: $L \neq L'$

- Dann gilt $x_{B+1} \neq x'_{B+1}$ und
$$H(x) = z_{B+1} = h_s(\underbrace{z_B||x_{B+1}}_{y}) = h_s(\underbrace{z'_B||x'_{B+1}}_{y'}) = z'_{B+1} = H(x').$$

Fall 2: $L = L'$

- Sei i maximal mit $z_{i-1}||x_i \neq z'_{i-1}||x'_i$.
- Aus der Maximalität von i folgt $z_i = z'_i$.
- Damit gilt $z_i = h_s(\underbrace{z_{i-1}||x_i}_{y}) = h_s(\underbrace{z'_{i-1}||x'_i}_{y'}) = z'_i$.

Hashfunktionen in der Praxis

- Praktische Hashfunktionen verwenden gewöhnlich kein s.
- Damit sind sie im theoretischen Sinne nicht kollisionsresistent, da ein trivialer Angriff existiert, der eine Kollision ausgibt.
- Trotzdem können die besten *bekannten* Angriffe natürlich Komplexität $\Omega(2^{\frac{n}{2}})$ besitzen.
- Fast alle Hashfunktionen verwenden eine Kompressionsfunktion in Kombination mit der Merkle-Damgard Transformation.
- Als kollisionsresistent in der Praxis gelten derzeit z.B. SHA-2, TIGER, Whirlpool, FORK-256.
- Als nicht kollisionsresistent gelten: SHA-0, SHA-1, MD4, MD5, RIPEMD, Snefru, HAVAL, PANAMA, SMASH, etc.
- Kryptanalyse 2004 für SHA-0, SHA-1, MD4, MD5 von Wang et al.
- Derzeit Hash Algorithm Competition der NIST für neuen Standard.

Nested MAC für Nachrichten beliebiger Länge

Idee von NMAC:

- Hashe $m \in \{0,1\}^n$ auf einen Hashwert in $\{0,1\}^n$.
- Verwende MAC Π_h auf dem Hashwert.
- Wir konstruieren Π_h mittels schlüsselabhängiger Hashfunktion.

Algorithmus MAC Π_h fester Länge

Sei (Gen', h) eine kollisionsresistente Hashfkt $h : \{0,1\}^{2n} \to \{0,1\}^n$.

1. **Gen:** $s \leftarrow Gen'(1^n)$. Wähle $k_1 \in_R \{0,1\}^n$.
2. Bei Eingabe (s, k_1) und $m \in \{0,1\}^n$, berechne $t := h_s(k_1||m)$.
3. Bei Eingabe (s, k_1) und $m \in \{0,1\}^n$, verifiziere $t \stackrel{?}{=} h_s(k_1||m)$.

NMAC

Notation: Sei H_s^{IV} eine Merkle-Damgard Hashfunktion, bei der der Initialisierungsvektor auf den Wert IV gesetzt ist.

Algorithmus NMAC (Nested MAC)

Sei (Gen', h), $\Pi_h = (Gen^{\Pi}, Mac^{\Pi}, Vrfy^{\Pi})$ wie zuvor. Sei (Gen', H) die Merkle-Damgard Transformation von (Gen', h).

1. **Gen:** $s \leftarrow Gen'(1^n)$. Wähle Schlüssel $k_1, k_2 \in_R \{0,1\}^n$.
2. **Mac:** Bei Eingabe (s, k_1, k_2) und $m \in \{0,1\}^n$, berechne
$$t := Mac^{\Pi}_{s,k_1}(H_s^{k_2}(m)) = h_s(k_1 || H_s^{k_2}(m)).$$
3. **Vrfy:** Bei Eingabe (s, k_1, k_2) und $(m, t) \in \{0,1\}^n \times \{0,1\}^n$,
$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } t = Mac_{s,k_1,k_2}(m) \\ 0 & \text{sonst} \end{cases}.$$

Praxis-Variante: Fixiere s, d.h. einzelne Hash-Funktion (z.B. SHA-1).

Sicherheit von NMAC

Satz Sicherheit von NMAC

Sei (Gen', h) kollisionsresistent und sei Π sicher. Dann ist NMAC sicher.

Beweisskizze:

- Sei $\mathcal{A}$ ein Angreifer für NMAC.
- $\mathcal{A}$ stelle $Mac(\cdot)$ Orakelanfragen aus $Q = \{m_1, \ldots, m_q\}$.
- Anschließend gebe $\mathcal{A}$ gültiges (m, t) aus mit $m \notin Q$.

Fall 1: Es existiert ein $i \in [q]$ mit $H_s^{k_2}(m) = H_s^{k_2}(m_i)$.

- Wegen $m \neq m_i$ ist (m, m_i) eine Kollision für $H_s^{k_2}$.
- Nach Merkle-Damgard Konstruktion liefert dies Kollision für h_s.

Fall 2: Es gilt $H_s^{k_2}(m) \neq H_s^{k_2}(m_i)$ für alle $i \in [q]$.

- Sei $Q' = \{H_s^{k_2}(m) \mid m \in Q\}$. Es gilt $H_s^{k_2}(m) \notin Q'$.
- Damit ist $(H_s^{k_2}(m), t)$ eine gültige Fälschung für Π.

HMAC – Hash-Based MAC

Nachteil von NMAC: Benötigen das Setzen von IV in H.

Idee von HMAC:

- Erzeuge k_1, k_2 durch Vorschalten einer Anwendung von h_s.
- Definieren Konstanten $opad, ipad$ und berechnen

$$k_1 = h_s(IV||k \oplus opad) \text{ und } k_2 = h_s(IV||k \oplus ipad).$$

Algorithmus HMAC

Sei (Gen', H) wie zuvor. Seien $opad, ipad \in \{0,1\}^n$ konstant.

1. **Gen:** $s \in Gen'(1^n)$. Wähle $k \in_R \{0,1\}^n$.
2. **Mac:** Für (s,k) und $m \in \{0,1\}^n$ berechne
$$H_s(k \oplus opad||H_s(k \oplus ipad||m)).$$
3. **Vrfy:** Für (s,k) und $(m,t) \in \{0,1\}^{2n}$, verifiziere $t \stackrel{?}{=} Mac_{s,k}(m)$.

Anmerkung:

$$Mac_{s,k}(m) = H_s(k \oplus opad||\underbrace{H_s(k \oplus ipad||m)}_{H_s^{k_2}(m)}) = H_s^{k_1}(H_s^{k_2}(m)).$$

HMAC ist eine Variante von NMAC

- Wir berechnen beim HMAC den MAC-Wert $H_s^{k_1}(H_s^{k_2}(m))$.
- D.h. die äußere Hashfunktion $H_s^{k_1}$ wird stets auf einen Nachrichtenblock $H_s^{k_2}(m) \in \{0,1\}^n$ fester Länge angewendet.
- Daher ist das Anhängen der Nachrichtenlänge bei $H_s^{k_1}$ unnötig.
- Entspricht der Berechnung von $h_s^{k_1}(H_s^{k_2}(m))$, analog zu NMAC.
- D.h. HMAC ist ein Spezialfall von NMAC, wobei k_1 und k_2 aus k mittels Anwendung von h_s abgeleitet werden.
- Wir definieren den folgenden Pseudozufallsgenerator

$$G(k) = \underbrace{h_s(IV||k \oplus opad)}_{k_1} || \underbrace{h_s(IV||k \oplus ipad)}_{k_2}.$$

Korollar Sicherheit von HMAC mittels Sicherheit von NMAC

Sei G ein Pseudozufallsgenerator, (Gen', h) kollisionsresistent und Π_h sicher. Dann ist die HMAC-Konstruktion sicher.

Praktische Bedeutung von HMAC

Anwendung von HMAC:

- Vorgestellt 1996 von Bellare, Canetti und Krawczyk.
- HMAC wird in der Praxis oft in Kombination mit SHA-1 verwendet.
- HMAC findet Anwendung z.B. in den Protokollen Internet Protocol Security (IPSec) und Transport Layer Security (TLS).
- Wurde 1998 standardisiert und ist weitverbreitet in der Praxis.
- HMAC ist im Vergleich zum CBC-MAC deutlich schneller.

CCA-sichere Verschlüsselung

Idee:

- Der Verschlüsseler authentisiert c mit Hilfe eines MACs t.
- D.h. nur er ist in der Lage, gültige Paare (c,t) zu erzeugen.
- Damit ist ein CCA-Entschlüsselungsorakel für Angreifer nutzlos.

Algorithmus CCA-sichere Verschlüsselung Π_{cca}

Sei $\Pi_E = (Gen_E, Enc, Dec)$ ein Verschlüsselungsverfahren und $\Pi_M = (Gen_M, Mac, Vrfy)$ ein MAC.

1. **Gen':** $k_1 \leftarrow Gen_E(1^n)$, $k_2 \leftarrow Gen_M(1^n)$.
2. **Enc':** Bei Eingabe von m und (k_1, k_2), berechne $c \leftarrow Enc_{k_1}(m)$ und $t \leftarrow Mac_{k_2}(c)$. Ausgabe des Chiffretextes (c,t).
3. **Dec':** Bei Eingabe von (c,t) und (k_1,k_2),

$$\text{Ausgabe} = \begin{cases} m := Dec_{k_1}(c) & \text{falls } Vrfy_{k_2}(c,t) = 1 \\ \perp & \text{sonst} \end{cases}.$$

Eindeutige Tags

Definition MAC mit eindeutigem Tag

Sei $\Pi_M = (Gen, Mac, Vrfy)$ ein MAC. Π_M besitzt *eindeutige Tags* falls für alle k, m genau ein t existiert mit $Vrfy_k(m,t) = 1$.

Anmerkungen:

- D.h. der *Mac*-Algorithmus ist deterministisch.
- Bsp: CBC-MAC, NMAC und HMAC besitzen alle eindeutige Tags.

Π_{cca} ist CCA-sicher

Satz CCA-Sicherheit von Π_{cca}

Sei Π_E CPA-sicher und Π_M ein sicherer MAC mit eindeutigen Tags. Dann ist Π_{cca} CCA-sicher.

Beweisskizze:

- Offenbar ist Π_{cca} sicher gegenüber CPA-Angreifern $\mathcal{A}$.
- Wir zeigen nun, dass ein $Dec(\cdot)$-Orakel für $\mathcal{A}$ nutzlos ist.
- Sei (c, t) eine Anfrage von $\mathcal{A}$ an $Dec(\cdot)$.

Fall 1: (c, t) kommt aus voriger $Enc(m)$-Anfrage von $\mathcal{A}$.

- Dann weiss $\mathcal{A}$ bereits, dass $Dec(c, t)$ die Antwort m liefert.
- D.h. das Entschlüsselsorakel liefert keine nützliche Information.

Fall 2: (c, t) kommt nicht aus $Enc(m)$-Anfrage.

- Falls $Vrfy_{k_2}(c, t) = 1$, hat $\mathcal{A}$ einen gültigen Tag t für ein neues c konstruiert (folgt aus der Eindeutigkeit der Tags).
- Aufgrund der MAC-Sicherheit geschieht dies mit Ws $\leq \mathrm{negl}(n)$.
- D.h. $Dec(\cdot)$ gibt mit Ws $\geq 1 - \mathrm{negl}(n)$ die nutzlose Ausgabe $\perp$.

Verfahren zur Nachrichtenübermittlung

Ziel: *Vertraulichkeit* und *Integrität* der Nachricht

Definition Nachrichtenübermittlung

Sei $\Pi_E = (Gen_E, Enc, Dec)$ ein Verschlüsselungsverfahren und $\Pi_M = (Gen_M, Mac, Vrfy)$ ein Mac. Ein *Nachrichtenübermittlungs-verfahren* $\Pi = (Gen', EncMac', Dec')$ besteht aus

1. **Gen':** $k_1 \leftarrow Gen_E(1^n)$, $k_2 \leftarrow Gen_M(1^n)$
2. **EncMac':** Bei Eingabe von m und (k_1, k_2), berechne mittels Enc_{k_1} und Mac_{k_2} einen authentisierten Chiffretext γ.
3. **Dec':** Bei Eingabe von γ und (k_1, k_2), berechne mittels Dec_{k_1} und $Vrfy_{k_2}$ einen Klartext m oder eine Fehlerausgabe $\perp$. Es gilt

$$Dec'_{k_1,k_2}(EncMac'_{k_1,k_2}(m)) = m \text{ für alle } (k_1, k_2) \text{ und } m \in \{0,1\}^*.$$

Sicherheitsspiel der Nachrichtenübermittlung

Spiel Nachrichtenübermittlung $Auth_{\mathcal{A},\Pi'}(n)$

Sei $\mathcal{A}$ ein Angreifer für $\Pi' = (Gen', EncMac', Dec')$.

1. $k = (k_1, k_2) \leftarrow Gen'(1^n)$
2. $\gamma \leftarrow \mathcal{A}^{EncMac'_k(\cdot)}(1^n)$, d.h. $\mathcal{A}$ darf $EncMac_k(m)$ für beliebige m anfragen.
3. Sei Q die Menge der $EncMac_k(\cdot)$-Anfragen und $m := Dec'_k(\gamma)$.

$$Auth_{\mathcal{A},\Pi'}(n) = \begin{cases} 1 & \text{falls } m \neq \perp \text{ und } m \notin Q \\ 0 & \text{sonst} \end{cases}.$$

Sicherheit eines Nachrichtenübertragungsverfahrens

Definition Authentisierte Kommunikation

Ein Nachrichtenübertragungsverfahren Π' liefert *authentisierte Kommunikation* falls für alle ppt Angreifer $\mathcal{A}$ gilt

$$\mathrm{Ws}[Auth_{\mathcal{A},\Pi'}(n) = 1] \leq \mathrm{negl}(n).$$

Definition Sicherheit eines Nachrichtenübertragungsverfahrens

Sei Π' ein Nachrichtenübertragungsverfahren. Π' heißt *sicher*, falls es CCA-sicher ist und authentisierte Kommunikation liefert.

Sicherheit von Encrypt-then-authenticate Π_{cca}

Satz Sicherheit von Π_{cca}

Sei Π_E CPA-sicher und Π_M ein sicherer MAC mit eindeutigen Tags. Dann ist Π_{cca} ein sicheres Nachrichtenübermittlungsverfahren.

Beweis:

- Die CCA-Sicherheit von Π_{cca} wurde bereits gezeigt.
- Sei $\mathcal{A}$ ein Angreifer im Spiel $Auth_{\mathcal{A},\Pi_{cca}}(n)$ mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren daraus einen Angreifer $\mathcal{A}'$ für Π_M.

Algorithmus Angreifer $\mathcal{A}'$ für Π_M

EINGABE: 1^n, Orakelzugriff auf $Mac_{k_2}(\cdot)$

1. $k_1 \leftarrow Gen_E(1^n)$
2. $\gamma = (c', t') \leftarrow \mathcal{A}^{EncMac'_{k_1,k_2}(\cdot)}(1^n)$, bei $EncMac'_{k_1,k_2}(m)$-Anfrage berechne $c \leftarrow Enc_{k_1}(m)$, $t = Mac_{k_2}(c)$ und antworte mit (c, t).

AUSGABE: $\gamma = (c', t')$

Sicherheit von Encrypt-then-authenticate Π_{cca}

Beweis: Fortsetzung

- Sei $Q = \{m_1, \ldots, m_q\}$ die Menge der $EncMac_{k_1,k_2}(\cdot)$-Anfragen.
- Seien c_i die Verschlüsselungen von m_i für $i = 1, \ldots, q$ in Schritt 2.
- Falls $\mathcal{A}$ Erfolg hat, so gilt $m' := Dec_{k_1,k_2}(c') \notin Q$.
- Daraus folgt $c' \notin \{c_1, \ldots, c_q\}$. D.h. t' ist ein Tag für eine nicht an das $Mac_{k_2}(\cdot)$-Orakel angefragte Nachricht c'.
- Es folgt

$$\mathrm{Ws}[\text{Mac-forge}_{\mathcal{A}',\Pi_M}(n) = 1] \geq \mathrm{Ws}[Auth_{\mathcal{A},\Pi_{cca}}(n) = 1] = \epsilon(n).$$

- Aus der Sicherheit von Π_M folgt $\epsilon(n) \leq \mathrm{negl}(n)$.

Anmerkung:

- Π_{cca} ist sicher für *jedwede* sichere Instantiierung von Π_E und Π_M.

Die Notwendigkeit zweier Schlüssel k_1, k_2

Faustregel Verwendung verschiedener Schlüssel

Verschiedene Sicherheitsziele sollten durch Wahl verschiedener Schlüssel realisiert werden.

Bsp: Unsicheres Encrypt-then-authenticate durch einen Schlüssel

- Wir verwenden denselben Schlüssel k für Π_E und Π_M.
- Sei F eine starke Pseudozufallspermutation auf n Bits.
- D.h. F^{-1} ist ebenfalls eine starke Pseudozufallspermutation.
- Wir konstruieren ein CPA-sicheres Π_E (Übung) mittels
$$Enc_k(m) = F_k(m||r) \text{ für } m \in \{0,1\}^{\frac{n}{2}}, r \in_R \{0,1\}^{\frac{n}{2}}.$$
- Wir konstruieren einen sicheren MAC Π_M mittels
$$Mac_k(c) = F_k^{-1}(c) \text{ für } c \in \{0,1\}^n.$$
- Encrypt-then-authenticate liefert
$$\gamma = (c,t) = (F_k(m||r), F_k^{-1}(F_k(m||r))) = (c, m||r).$$
- D.h. γ gibt die Nachricht m preis.

Encrypt-and-authenticate kann unsicher sein

Encrypt-and-authenticate:

$$\gamma = EncMac(m) := (c, t) = (Enc_{k_1}(m), Mac_{k_2}(m)).$$

- D.h. der Tag wird für die Nachricht m berechnet, nicht für c.
- Sei Π_E CPA-sicher und $\Pi_M = (Gen_M, Mac, Vrfy)$ ein sicherer Mac.
- Dann ist Π'_M mit $Mac'_{k_2}(m) = (m, Mac_k(m))$ ebenfalls sicher, denn gültige Tags (m, t) für Π'_M liefern gültige Tags t für Π_M.
- Instantiierung von Encrypt-and-authenticate mit Π_E, Π'_M liefert
$$\gamma = (c, (m, Mac_{k_2}(m))).$$
- D.h. γ gibt die Nachricht m preis, obwohl Π_E und Π'_M sicher sind.

Authenticate-then-encrypt kann unsicher sein

Authenticate-then-encrypt: $\gamma = EncMac(m) := Enc_{k_1}(m||Mac_{k_2}(m))$

- Kodieren Nachricht $m \in \{0,1\}^*$ vor Verschlüsselung:
  Jede 0 wird als 00 oder 11 kodiert, jede 1 als 01 oder 10.
- Dekodierung in Zweierblöcken: 00 oder 11 zu 0, 01 oder 10 zu 1.
- Wir verschlüsseln $Enc_k(m) = Enc'_k(\text{Kodierung}(m))$, wobei Enc' die CPA-sichere Verschlüsselung im Counter-Modus ist. Erinnerung:
  $Enc'(m_1 \dots m_\ell) = (ctr, m_1 \oplus r_1, \dots, m_\ell \oplus r_\ell)$ mit $r_i = F_k(ctr + i \bmod 2^n)$.

Algorithmus CCA-Angreifer

EINGABE: $c = Enc'_k(\text{Kodierung}(m||Mac_{k_2}(m)))$, $Dec_k(\cdot)$-Orakel

Für alle Zweierblöcke in c, die eine Kodierung von m enthalten:

1. Berechne c' durch Flippen eines Zweierblocks in c.
2. Falls $Dec(c') \in \{00, 11\}$, entschlüssele den Zweierblock zu 0.
3. Falls $Dec(c') \in \{01, 10\}$, entschlüssele den Zweierblock zu 1.

AUSGABE: m

Authenticate-then-encrypt kann unsicher sein

Fall 1: Zweierblock entspricht Kodierung 00 oder 11 eines Bits $m_j = 0$

- Flippen wechselt zwischen den zwei Kodierungen von $m_j = 0$.
- Damit erhält man bei der Dekodierung noch immer eine 0.
- $Mac_{k_2}(m)$ bleibt gültig, da nur die Kodierung von m geändert wird, nicht m selbst. Mac_{k_2} wird nicht auf $\text{Kodierung}(m)$ angewendet.

Fall 2: Zweierblock entspricht Kodierung 01 oder 10 eines Bits $m_j = 1$

- Argumentation ist analog zum obigen Fall.

Anmerkungen:

- Bsp. zeigt, dass Authenticate-then-encrypt i. allg. nicht sicher ist.
- Das SSL (Secure Sockets Layer) Protokoll im Internet verwendet eine sichere Variante von Authenticate-then-encrypt.

Erinnerung Blockchiffre

Definition schlüsselabhängige Permutation

Seien F, F^{-1} pt Algorithmen. F heißt *schlüsselabhängige Permutation* auf ℓ Bits falls

1. F berechnet eine Funktion $\{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$, so dass für alle $k \in \{0,1\}^n$ die Funktion $F_k(\cdot)$ eine Bijektion ist.
2. $F_k^{-1}(\cdot)$ berechnet die Umkehrfunktion von $F_k(\cdot)$.

Definition Starke Pseudozufallspermutation (Blockchiffre)

Sei F eine schlüsselabhängige Permutation auf ℓ Bits. Wir bezeichnen F als *starke Pseudozufallspermutation (Blockchiffre)*, falls für alle ppt D gilt

$$\left|\mathrm{Ws}[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n)=1] - \mathrm{Ws}[D^{f(\cdot),f^{-1}(\cdot)}(1^n)=1]\right| \leq \mathrm{negl}(n),$$

mit $k \in_R \{0,1\}^n$ und $f \in_R Perm_\ell$.

Angriffe auf Blockchiffren

Angriffe: in aufsteigender Stärke

1. Ciphertext-only: $\mathcal{A}$ erhält $F_k(x_i)$ für unbekannte x_i.
2. Known plaintext: $\mathcal{A}$ erhält Paare $(x_i, F_k(x_i))$
3. Chosen plaintext: $\mathcal{A}$ wählt x_i und erhält $F_k(x_i)$.
4. Chosen ciphertext: $\mathcal{A}$ wählt x_i, y_i und erhält $F_k(x_i), F_k^{-1}(y_i)$.

Sicherheit:

- Jede Pseudozufallspermutation F ist CPA-sicher.
- Jede starke Pseudozufallspermutation F ist CCA-sicher.

Warnung:
Blockchiffren selbst sind **kein sicheres** Verschlüsselungsschema.

Substitutions-Permutations Netzwerk (SPN)

Ziel: Kleine Eingabedifferenzen erzeugen pseudozufällige Ausgaben.

Paradigma Konfusion und Diffusion

Rundeniterierte Vorgehensweise zur Konstruktion einer Blockchiffre

1. **Konfusion:** Permutiere kleine Bitblöcke schlüsselabhängig.
2. **Diffusion:** Permutiere alle Bits.

Bsp: F soll Blocklänge 128 Bits besitzen.

- Konfusion: Definiere schlüsselabhängige Permutation $f_1, \ldots, f_{16}$ auf 8 Bits. Sei $x = x_1 \ldots x_{16} \in (\{0,1\}^8)^{16}$. Definiere
$$F_k(x) = f_1(x_1) \ldots f_{16}(x_{16}).$$
- Diffusion: Permutiere die Bits von $F_k(x)$.
- Iteriere die obigen beiden Schritte hinreichend oft, damit kleine Eingabedifferenzen sich auf alle Ausgabebits auswirken.
- Beschreibungslänge von f_i: $8 \cdot 2^8$ Bits, F : $16 \cdot 8 \cdot 2^8 = 2^{15}$ Bits.
- Länge einer echten Zufallspermutation: $128 \cdot 2^{128} = 2^{135}$ Bits.

Substitutions-Permutations Netzwerk (SPN)

Szenario: Verwende einen Masterschlüssel k.

- Berechne aus dem Masterschlüssel k Rundenschlüssel $k_1, \ldots k_r$ mittels eines sogenannten Keyschedule-Algorithmus.
- Die Permutationsfunktionen $f_1, \ldots, f_m$ werden fest und schlüsselunabhängig gewählt (sogenannte S-Boxen).

Beschreibung Substitutions-Permutations Netzwerk (SPN)

EINGABE: $f_1, \ldots, f_m, k \in \{0,1\}^n, x, \ell$

1. Berechne $k_1, \ldots, k_r \in \{0,1\}^\ell$ aus k. $y \leftarrow x$.
2. For $i \leftarrow 1$ to r
   1. **Schlüsseladdition:** $y \leftarrow y \oplus k_i$. Schreibe $y = y_1 \ldots y_m$.
   2. **Substitution per S-Boxen:** $y \leftarrow f_1(y_1) \ldots, f_m(y_m)$
   3. **Permutation:** $y \leftarrow$ Permutation der Bits von y.

AUSGABE: $F_k(x) := y$

Beobachtung: F ist invertierbar, da jeder Schritt invertierbar ist.

Lawineneffekt

Ziel: Veränderung in Eingabebit wirkt sich auf alle Ausgabebits aus.

Beobachtung Notwendige Eigenschaften für Lawineneffekt

1. **S-Box:** Ändern eines Eingabebits verändert ≥ 2 Ausgabebits.
2. **Permutation:** Ausgabebits einer S-Box werden zu Eingabebits verschiedener S-Boxen.

Beobachtung: Lawineneffekt

- Betrachten ein SPN mit 4 Bit S-Boxen und Blocklänge 128 Bit.
- 1-Bit Eingabedifferenz erzeugt mindestens eine 2-Bit Differenz.
- Eine 2-Bit Differenz resultiert in zwei 1-Bit Differenzen an verschiedenen S-Boxen in der nächsten Runde.
- Diese sorgen für mindestens 4-Bit Differenz, usw.
- D.h. jede Runde verdoppelt potentiell die beeinträchtigen Bits.
- Nach 7 Runden sind alle $2^7 = 128$ Bits von der Veränderung eines Eingabebits beeinträchtigt.

Angriff auf eine Runde eines SPN

Algorithmus Angriff auf eine Runde eines SPN

EINGABE: $x, y = F_k(x)$

1. $y :=$ Invertiere auf y die Permutation und die S-Boxen.
2. Berechne $k := x \oplus y$.

AUSGABE: k

Anmerkungen:

- Die Invertierung in Schritt 1 ist möglich, da sowohl die Permutation als auch die S-Boxen öffentlich sind.
- Nach Invertierung erhält man den Wert $x \oplus k$.

Angriff auf zwei Runden eines SPN

Szenario:

- Wir betrachten ein SPN mit Blocklänge 64 Bit und 2 Runden.
- Der Masterschlüssel $k = k_1 k_2$ ist 128 Bit lang.
- Die Schlüssel $k_1, k_2 \in \{0,1\}^{64}$ sind die zwei Rundenschlüssel.
- Wir schreiben $k_i = k_{i,1} \ldots k_{i,16}$ mit $k_{i,j} \in \{0,1\}^4$.
- Die S-Boxen sind 4 Bit groß. Wir erhalten 6 Paare (x_i, y_i).

Algorithmus Angriff auf zwei Runden eines SPN

EINGABE: (x_i, y_i) für $i = 1, \ldots, 6$

1. $w_1 \ldots w_{16} :=$ Invertiere auf y_i die Permutation und Substitution.
2. For $j = 1$ to 16
   1. w_j wird von Ausgabebits aus höchstens 4 S-Boxen der 1. Runde beeinflusst.
   2. Diese 4 Bits hängen von höchstens vier 4-Bit Werten von k_1 ab. Rate diese 4 Werte. Rate ebenfalls $k_{2,j}$.
   3. Entferne Schlüssel, die nicht mit allen (x_i, y_i) konsistent sind.

AUSGABE: $k = k_1 k_2$

Angriff auf zwei Runden eines SPN

Korrektheit:

- Jeder 4-Bit Block der zweiten Runde hängt von höchstens vier 4-Bit Blöcken der ersten Runde ab.

Laufzeit:

- Schritt 2.2: Raten von 20 Bits, d.h. 2^{20} Möglichkeiten.
- Wir nehmen an, dass ein falscher Schlüssel auf dem j-ten Block eines Paars (x_i, y_i) mit Ws $\frac{1}{2^4}$ korrekt ist.
- D.h. jedes Paar (x_i, y_i) reduziert den Schlüsselraum um Faktor $\frac{1}{2^4}$.
- Nach 6 Paaren (x_i, y_i) verbleibt nur der korrekte Schlüssel.
- Die Gesamtlaufzeit ist beschränkt durch $16 \cdot 2^{20} \cdot 6 < 2^{27}$.
- Man vergleiche mit der Komplexität 2^{128}, um k zu raten.

Distinguisher für 2 Runden

Bsp: Distinguisher für 2 Runden

- Wir betrachten Blocklänge 80 Bit und 4 Bit S-Boxen.
- Wähle x_i, die sich nur im ersten 4-Bit Block unterscheiden.
- Nach 1. Runde: Ausgaben unterscheiden sich in ≤ 4 Blöcken.
- Nach 2. Runde: Ausgaben unterscheiden sich in ≤ 16 Blöcken.
- D.h. nicht alle der 20 Ausgabeblöcke werden verändert.
- Können SPN leicht von Pseudozufallspermutation entscheiden.

Feistelnetzwerk

Szenario:

- Leite aus k Rundenschlüssel $k_1, \ldots, k_r$ ab.
- Teile Nachrichtenblock in linke Seite L_i und rechte Seite R_i.
- Sei n die Blocklänge. Definiere nicht notwendigerweise invertierbare Rundenfunktionen $f_i : \{0,1\}^{\frac{n}{2}} \rightarrow \{0,1\}^{\frac{n}{2}}$.
- Die Funktionen f_i hängen von den Rundenschlüsseln k_i ab.

Algorithmus Feistelnetzwerk

EINGABE: k, x, n, r

1. Leite $k_1, \ldots, k_r$ aus k ab.
2. Setze $(L_0 || R_0) := x$ mit $L_i, R_i \in \{0,1\}^{\frac{n}{2}}$.
3. For $i = 1$ to r
   1. Setze $L_i := R_{i-1}$ und $R_i := L_{i-1} \oplus f_i(R_{i-1})$.

AUSGABE: $F_k(x) := (L_r || R_r)$

Invertierung einer Feisteliteration: $R_{i-1} := L_i$ und $L_{i-1} := R_i \oplus f_i(R_{i-1})$.

DES - Data Encryption Standard

Beschreibung von DES:

- Entwickelt 1973 von IBM, standardisiert 1976.
- DES besitzt Schlüssellänge 56 Bit und Blocklänge 64 Bit.
- Besteht aus Feistelnetzwerk mit 16 Runden.
- Aus den Bits k werden 48-Bit Schlüssel $k_1, \ldots, k_{16}$ ausgewählt.
- Rundenfunktionen f_i sind SPNs mit nicht invertierbaren S-Boxen.

Algorithmus Rundenfunktion f_i

EINGABE: $k_i, R_{i-1} \in \{0,1\}^{32}$

1. $y :=$ Erweitere R_{i-1} auf 48 Bit durch Verdopplung von 16 Bits.
2. $y := y \oplus k_i$
3. $y :=$ Splitte y in 6-Bit Blöcke $y_1 \ldots y_8$ auf. Wende auf jedes y_i eine S-Box $S_i : \{0,1\}^6 \to \{0,1\}^4$ an. Permutiere das Ergebnis.

AUSGABE: $f_i(R_{i-1}) := y$

Die DES S-Boxen

DES S-Boxen:

- Alle 8 S-Boxen realisieren verschiedene Abb. $\{0,1\}^6 \to \{0,1\}^4$.
- Jede S-Box ist eine 4:1-Abbildung.
- D.h. jede S-Box sendet genau 4 Eingaben auf eine Ausgabe.
- Wechsel eines Eingabebits ändert mindestens zwei Ausgabebits.

Lawineneffekt bei DES:

- Wähle (L_0, R_0) und (L_0', R_0) mit 1-Bit Differenz in L_0, L_0'.
- (L_1, R_1) und (L_1', R_1') besitzen 1-Bit Differenz in R_1, R_1'.
- Durch f_2 erhält man mindestens eine 2-Bit Differenz in R_2, R_2'.
- D.h. (L_2, R_2) und (L_2', R_2') besitzen mind. eine 3-Bit Differenz.
- f_3 angewendet auf R_2, R_2' liefert mind. eine 4-Bit Differenz, usw.
- Nach 8 Runden erreicht man volle Diffusion auf alle Ausgabebits.

Angriff auf eine Runde DES

Algorithmus Angriff auf eine Runde DES

EINGABE: (x, y) mit $x = (L_0, R_0)$ und $y = (L_1, R_1)$

1. Wir kennen ein Paar $(R_0, L_0 \oplus R_1)$ mit $f_1(R_0) = L_0 \oplus R_1$.
2. Berechne die Ausgaben für jede der acht S-Boxen.
3. Für jede Ausgabe gibt es 4 mögliche 6-Bit Eingaben, aus denen sich 4 Möglichkeiten für die 6 betreffenden Bits von k_1 ergeben.
4. Teste alle 4^8 Möglichkeiten für k_1 und verifiziere mittels (x, y).

AUSGABE: k_1

Laufzeit:

- Rekonstruieren k_1 mit Komplexität $4^8 = 2^{16}$ und einem Paar (x, y).

Angriff auf zwei Runden DES

Algorithmus Angriff auf zwei Runden DES

EINGABE: (x, y) mit $x = (L_0, R_0)$ und $y = (L_2, R_2)$

1. Setze $L_1 = R_0$ und $R_1 = L_0 \oplus f_1(R_0)$.
2. Verwende $(L_0, R_0), (L_1, R_1)$, um k_1 wie zuvor zu bestimmen.
3. Verwende $(L_1, R_1), (L_2, R_2)$, um k_2 wie zuvor zu bestimmen.

AUSGABE: k_1, k_2

Laufzeit:

- Wir benötigen Laufzeit 2^{17} mit einem Paar (x, y).
- Kann reduziert werden, wenn wir das DES-Keyscheduling berücksichtigen, bei dem Bits von k_1 einige Bits von k_2 festlegen.

Angriff auf drei Runden DES

Eigenschaften des DES-Keyschedule:

- Sei $k = k^{(1)}k^{(2)} \in \{0,1\}^{56}$ der Masterschlüssel mit $k^{(i)} \in \{0,1\}^{28}$.
- Für jeden Rundenschlüssel $k_i \in \{0,1\}^{48}$ gilt: Die ersten 24 Bits von k_i werden aus $k^{(1)}$ gewählt, die zweiten 24 Bits aus $k^{(2)}$.

Idee zum Angriff auf drei Runden DES:

- Sei ein Paar $(x,y) = ((L_0,R_0),(L_3,R_3))$ gegeben.
- Die Eingaben von f_1 bzw. f_3 sind R_0 bzw. R_3.
- Die Ausgaben von f_1 bzw. f_3 sind $(L_0 \oplus L_2)$ bzw. $(L_2 \oplus R_3)$.
- Da L_2 unbekannt ist, sind beide Ausgaben unbekannt.
- Allerdings ist das XOR der Ausgaben von f_1, f_3 bekannt: $L_0 \oplus R_3$.
- Wir raten $k^{(1)}$ und $k^{(2)}$ separat und Testen dieses XOR.

Angriff auf drei Runden DES

Algorithmus Angriff auf drei Runden DES

EINGABE: $(x_1, y_1), (x_2, y_2)$

1. Rate $k^{(1)}$. Berechne die ersten 24 Bits von k_1 und k_3.
2. Berechne die Eingabe der S-Boxen $S_1, \ldots, S_4$ von f_1.
3. Berechne die Eingabe der S-Boxen $S_1, \ldots, S_4$ von f_3.
4. D.h. wir kennen dieselben 16 Ausgabebits von f_1 und f_3.
5. Vergleiche das XOR dieser Bits mit $L_0 \oplus R_3$ für beide Paare (x_1, y_1), (x_2, y_2). Bei Ungleichheit verwerfe $k^{(1)}$.
6. Verfahren analog beim Raten des Teilschlüssels $k^{(2)}$.

AUSGABE: Masterschlüssel $k = k^{(1)}k^{(2)}$

Laufzeit:

- Annnahme: Inkorrekte Teilschlüssel $k^{(1)}$ stimmen auf den 16 Bits in Schritt 5 mit Ws $\frac{1}{2^{16}}$ überein. Daher genügen zwei Paare (x_i, y_i).
- Raten für $k^{(i)}$ jeweils 28 Bits, d.h. die Gesamtkomplexität ist 2^{29}.

Die (Un-)Sicherheit von DES

Sicherheit von DES:

- Bester praktischer Angriff ist noch immer die Brute-Force Suche.
- Die folgende Tabelle gibt eine Übersicht über DES Kryptanalysen.

Jahr	Projekt	Zeit
1997	DESCHALL, Internet	96 Tage
1998	distributed.net, Internet	41 Tage
1998	Deep Crack, 250.000 Dollar Maschine	2 Tage
2008	COPACOBANA, 10.000 Euro FPGAs	1 Tag

- Das Design von DES ist gut, nur die Schlüssellänge ist zu kurz.
- Die Blocklänge von 64 Bits von DES gilt als zu kurz (s. Folie 78).

Strukturelle Angriffe - Differentielle Kryptanalyse

Differentielle Kryptanalyse:

- Eingeführt von Biham und Shamir (1991).
- Idee: Spezifische Eingabedifferenzen Δ_x führen zu spezifischen Ausgabedifferenzen Δ_y.

Definition Differential

Sei F eine Blockchiffre mit Blocklänge n. Ein *Differential* $(\Delta_x, \Delta_y) \in \{0,1\}^n \times \{0,1\}^n$ hält mit Ws p, falls für zufällige k und $x_1, x_2 \in_R \{0,1\}^n$ mit $x_1 \oplus x_2 = \Delta_x$ gilt $\mathrm{Ws}[F_k(x_1) \oplus F_k(x_2) = \Delta_y] = p$.

- Für Pseudozufallsfunktionen sollte kein Differential mit Wahrscheinlichkeit signifikant größer als 2^{-n} halten.
- Für große Differentiale ist F offenbar keine Pseudozufallsfunktion.
- Mehrere große Differentiale führen oft zu einem CPA-Angriff auf k.
- Finden von (Δ_x, Δ_y) geschieht entweder durch Brute-Force Suche oder durch cleveres Ausnutzen der Blockchiffrenstruktur.

Differentielle Kryptanalyse von DES

Differentielle Kryptanalyse von DES:

- CPA-Angriff mit 2^{36} von insgesamt 2^{47} gewählten Klartexten.
- Benötigt Laufzeit 2^{37} und vernachlässigbaren Speicherbedarf.
- Theoretisch deutlich besser als Brute-Force, die Anzahl der benötigten gewählten Klartexte ist allerdings sehr groß.
- DES wurde als resistent gegen differentielle Krypanalyse designt.
- Die DES-Designer veröffentlichten allerdings den Angriff nicht.

Strukturelle Angriffe - Lineare Kryptanalyse

Lineare Kryptanalyse:

- Entwickelt von Matsui (1993). Methode verwendet lineare Beziehungen der Ein- und Ausgabebits.

Definition Bias

Sei F eine Blockchiffre mit Blocklänge n. Seien $I = \{i_1, \ldots, i_\ell\}$, $J = \{j_1, \ldots, j_\ell\} \subseteq \{1, \ldots, n\}$. Die Indexmengen I, J besitzen *Bias* p, falls für zufällige k und $x \in \{0,1\}^n$ mit $y = F_k(x)$ gilt:

$$\mathrm{Ws}[x_{i_1} \oplus \ldots \oplus x_{i_\ell} \oplus y_{j_1} \oplus \ldots \oplus y_{j_\ell} = 0] = p.$$

- Für echte Zufallsfunktionen sollte der Bias nie signifikant $> \frac{1}{2}$ sein.
- Matsui: Ein genügend großer Bias einer Blockchiffre führt dazu, dass der geheime Schlüssel rekonstruiert werden kann.
- Methode ist KPA, d.h. benötigt keine gewählten Klartexte.
- Liefert KPA-Angriff auf DES mit 2^{43} bekannten Paaren (x_i, y_i).
- Laufzeit ist 2^{43}, Speicherplatz ist vernachlässigbar.

Doppelte Verschlüsselung bringt wenig

Szenario: doppelte Verschlüsselung

- Sei F eine Blockchiffre mit Schlüssellänge n wie z.B. DES.
- Dann besitzt $F'_{k_1,k_2}(x) = F_{k_2}(F_{k_1}(x))$ Schlüssellänge $2n$.
- Leider liefert F' kein Sicherheitsniveau von 2^{2n}.

Algorithmus Meet-in-the-Middle Angriff auf doppelte Verschl.

EINGABE: (x_1, y_1), (x_2, y_2)

1. Für alle $k_1 \in \{0,1\}^n$, berechne $z := F_{k_1}(x_1)$. Speichere (z, k_1) in einer nach der ersten Komponente sortierten Liste L_1.
2. Für alle $k_2 \in \{0,1\}^n$, berechne $z := F^{-1}_{k_2}(y_1)$. Speichere (z, k_2) in einer nach der ersten Komponente sortierten Liste L_2.
3. Für alle z mit $(z, k_1) \in L_1$ und $(z, k_2) \in L_2$, speichere (k_1, k_2) in S.
4. Für alle $(k_1, k_2) \in S$: Verifiziere Korrektheit mittels (x_2, y_2).

AUSGABE: $k = (k_1, k_2)$

Doppelte Verschlüsselung bringt wenig

Korrektheit:

- Für korrektes (k_1, k_2) gilt $F_{k_2}(F_{k_1}(x)) = y$, d.h. $F_{k_1}(x) = F_{k_2}^{-1}(y)$.
- Ein falsches (k_1, k_2) erfüllt diese Identität mit Ws etwa 2^{-n}.
- D.h. wir erwarten $2^{2n} \cdot 2^{-n} = 2^n$ Elemente in der Menge S.
- Verifizieren mit (x_2, y_2) liefert den korrekten Schlüssel.

Laufzeit: Wir zählen Operationen auf einzelnen Schlüsseln mit Zeit/Platz $\mathcal{O}(1)$.

- Schritt 1 und 2: jeweils Zeit $\mathcal{O}(n \cdot 2^n)$ und Platz $\mathcal{O}(2^n)$.
- Schritt 3: Laufzeit und Platz $\mathcal{O}(2^n)$.
- Schritt 4 lässt sich in Laufzeit $\mathcal{O}(2^n)$ realisieren.
- D.h. wir erhalten Gesamtlaufzeit $\mathcal{O}(n \cdot 2^n)$ und Platz $\mathcal{O}(2^n)$.
- Damit erhöht sich die Laufzeit gegenüber einem Brute-Force Angriff bei einfacher Verschlüsselung nicht wesentlich.

Dreifache Verschlüsselung

Szenario: dreifache Verschlüsselung

1. Variante 1: $F'_{k_1,k_2,k_3}(x) := F_{k_3}(F^{-1}_{k_2}(F_{k_1}(x)))$
2. Variante 2: $F'_{k_1,k_2}(x) := F_{k_1}(F^{-1}_{k_2}(F_{k_1}(x)))$

Grund des Alternierens von F, F^{-1}, F: Für die Wahl von $k_1 = k_2 = k_3$ erhalten wir eine einfache Anwendung von $F_{k_1}(x)$.

Sicherheit der 1. Variante:

- Meet-in-the-Middle Angriff wie zuvor in Zeit $\mathcal{O}(n \cdot 2^{2n})$.
- Benötigt Speicherplatz $\mathcal{O}(2^{2n})$.

Sicherheit der 2. Variante:

- Bekannter CPA-Angriff mit $\mathcal{O}(2^n)$ gewählten Paaren.
- Zeitkomplexität beträgt ebenfalls $\mathcal{O}(2^n)$.

Triple-DES:

- Beide Varianten von Triple-DES finden in der Praxis Verwendung.
- Löste 1999 DES als Standard ab. Trotz Standardisierung von AES im Jahr 2002 ist Triple-DES auch heute noch weitverbreitet.

AES - Advanced Encryption Standard

NIST Wettbewerb: (National Institute of Standard and Technology)

- Jan 1997: Aufruf zum Konstruktions-Wettbewerb einer Blockchiffre
- Ursprünglich 15 Kandidaten eingereicht.
- Aug 1999: Auswahl von fünf AES-Finalisten

  MARS, RC6, Rijndael, Serpent und Twofish.
- Okt 2000: Auswahl von Rijndael der Autoren Rijmen und Daemen.

Struktur von AES (Rijndael):

- AES ist ein SPN und besitzt Blocklänge 128.
- Schlüssel mit 128, 192 und 256 Bit können verwendet werden.
- Eingaben $x \in \{0,1\}^{128}$ werden rundenweise in einer 4×4-Byte Matrix, der sogenannten Zustandsmatrix, modifiziert.
- Anzahl Runden: 10 für 128-Bit k, 12 für 192-Bit und 14 für 256-Bit.

Die vier Rundenoperationen von AES

Operation 1: AddRoundKey

- Leite aus k einen Rundenschlüssel $k_i \in \{0,1\}^{128}$ ab.
- XOR der Zustandsmatrix mit k_i.

Operation 2: SubByte

- Interpretiere jedes Byte der Zustandsmatrix als Element $x \in \mathbb{F}_8$.
- Ersetze x durch x^{-1} in $\mathbb{F}_8$ und 0^8 durch 0^8.
- Wende eine affine Transformation auf die Zustandsbytes an.
- Man beachte: Dieselbe S-Box wird für alle Bytes verwendet.

Operation 3: ShiftRow

- Verschiebe die 4 Zeilen der 4×4-Zustandsmatrix zyklisch.
- Lasse die 1. Zeile unverändert.
- Verschiebe die 2. Zeile um eine Position nach links, die 3. Zeile um 2 nach links und die 3. Zeile um 3 Positionen nach links.

Die vier Rundenoperationen von AES

Operation 4: MixColumn

- Sei $a_{0,j}, a_{1,j}, a_{2,j}, a_{3,j}$ eine Spalte der Zustandsmatrix.
- Betrachte die Spalte als Element aus $\mathbb{F}_8/(x^4+1)$, d.h.

$$a_{0,j} + a_{1,j}x + a_{2,j}x^2 + a_{3,j}x^3 \text{ mit } a_{i,j} \in \mathbb{F}_8.$$

- Multipliziere mit $c(x) = 2 + x + x^2 + 3x^3 \in \mathbb{F}_8/(x^4+1)$.
- Kodieren $a \in \mathbb{F}_8 = F_2[y]/\pi$, π irreduzibel mit Grad 8, wie folgt: Sei z.B. $a = y^5 + y + 1$. Wir schreiben $a = (00100011) = 36$.
- MixColumn entspricht Multiplikation mit einer Matrix $C \in \mathbb{F}_8^{4\times 4}$.
- D.h. eine Spalte x wird mittels $x \to Cx$ linear abgebildet.
- Menge aller (x, Cx) definiert einen linearen Code mit Distanz 5.
- D.h. unterscheiden sich zwei Spalten x, x' in nur einer Position, so unterscheiden sie sich nach MixColumn in allen 4 Positionen.
- Diese Eigenschaft führt zu einer schnellen Diffusion bei AES.