2.11.2020 - 8.11.2020 #1 and #2 ✓
9.11.2020 - 15.11.2020 #3 and #4 ✓
16.11.2020 - 22.11.2020 #5 ✓
23.11.2020 - 29.11.2020 #6 and #7 ✓

30.11.2020 - 6.12.2020 #8 and #9
7.12.2020 - 13.12.2020 #10 and #11
14.12.2020 - 20.12.2020 #12 and #13
21.12.2020 - 27.12.2020 #14
4.1.2021 - 10.1.2021 #15
11.1.2021 - 17.1.2021 #16 and #17

18.1.2021 - 24.1.2021 #18 and #19
25.1.2021 - 31.1.2021 #20 and #21
1.2.2021 - 7.2.2021 #22 and #23

8.2.2021 - 14.2.2021 #24 and #25
12.2.2021 QA session for the course and final exam

1 Grundlegende Konzepte und mathematisches Wissen

1.1 Wichtige Rechenoperationen

XOR-Operation

Für die XOR-Operation, auch $\oplus$-Operation genannt, werden zwei n lange Binärzahlen $b1 = \|_{i=0}^{n} b_i$ und $b2 = \|_{i=0}^{n} b_i$ verrechnet, indem die einzelnen Bits nach folgender Wahrheitstafel verrechnet werden:

$b1_i$	$b2_i$	$b1_i \oplus b2_i$
0	0	0
0	1	1
1	0	1
1	1	0

Beispiel 1.1. (XOR-Operator)

$b1$	=	10001000
$b2$	=	11110000
result $b_1 \oplus b_2$	=	01111000

1.2 Grundlegende Wahrscheinlichkeitstheorie

Wahrscheinlichkeitsraum

Ein Wahrscheinlichkeitsraum besteht aus drei Komponenten:
- Ω ist der Ergebnisraum (sample space)
- Σ ist die Menge an (Zwischen-)Ereignissen (event set)
- $P : \Sigma \to [0, 1]$ ist die Wahrscheinlichkeitsfunktion

Bemerkung 1.2. (Begrifflichkeiten)

- Die Wahrscheinlichkeitsfunktion kann auch Pr[...] genannt werden kann (Grund: Im Kontext von P und NP ist die Doppelbelegung von P doof). Es gibt keinen Unterschied zwischen beiden Notationen
- **Zufallsvariable:** Abstrakt: Eine ZV filtert die Elemente aus Ω, welchen schließlich ein Wert $\neq 0$ zugewiesen werden kann. Verständlich: Eine ZV ist eine Teilmenge von Ω. Σ bildet die Menge aller möglichen Zufallsvariablen

- **Ereignis:** Ein Element aus Ω

Beispiel 1.3. (Experiment mit einem d4-DnD-Würfel)

$\Omega = \{1, 2, 3, 4\}$

$\Sigma = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \Omega\}$

$P(\{1\}) = P(\{2\}) = P(\{3\}) = P(\{4\}) = 0.25$

$P(\{1, 2\}) = P(\{1, 3\}) = P(\{1, 4\}) = ... = P(\{3, 4\}) = 0.5$

$P(\{1, 2\} \wedge \{1, 4\}) = P(\{1, 2\} \wedge \{1, 3\}) = 0.25$

$P(\{1, 3\} \mid \{1, 2, 4\}) = P(\{2, 3\} \mid \{1, 3, 4\}) = 0.25/0.75 = 0.33...$

Anschaulich stellt die Zufallesvariable $\mathcal{X} = \{2, 4\}$ die Frage nach allen gerade Ergebnissen. Die Wahrscheinlichkeit für ein Event aus $x \in \mathcal{X}$ wird als $P(x = 2) = 0.25$ geschrieben.

Formelsammlung Stocharschtik

Für zwei **unabhängige** Events gilt:

$$P(a \wedge b) = P(a \wedge b) = P(a) \cdot P(b)$$

$$P(a \mid b) = P(a)$$

Für zwei **abhängige** Events gilt:

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}$$

Generell gilt:

$$P(a \vee b) = P(a) + P(b) + P(a \wedge b)$$

Bayesian Rules für **abhängige** Events:

- **Product Rule:**

$$P(a \wedge b) = P(a \mid b) \cdot P(b)$$

- **Chain Rule:**

$$P(x_1, ..., x_n) = P(x_n \mid x_{n-1}, ..., x_1) \cdot P(x_{n-1} \mid x_{n-2}, ..., x_1) \cdot ...$$

- **Marginalization:**

$$P(x) = \sum_{y \in Y} P(x, y)$$

$$P(a) = P(a \mid b) \cdot P(b) + P(a \mid \neg b) \cdot P(\neg b)$$

Bayes Theorem:

$$P(a \mid b) = \frac{P(a) \cdot P(b \mid a)}{P(b)}$$

Union Bound:

$$P(E_1 \vee ... \vee E_n) \leq P(E_1) + ... + P(E_n)$$

2 Symmetrische Verschlüsselung

2.1 Grundlegende Begrifflichkeiten und Definitionen

Definition 2.1 (Grundschemata einer Private Key Encryption)

Ein **Private Key Encryption** Schemata $\Pi = (Gen, Enc, Dec)$ wird über einen Klartextraum $\mathcal{M} = \{0, 1\}^*$ anhand von drei Algorithmen definiert:

- Gen: **Schlüsselgenerierung** (Key-Generation) erzeugt als Output einen Schlüssel k anhand einer gewissen Wahrscheinlichkeitsverteilung
- $Enc_k(m)$: **Verschlüsselung** (Encryption) nimmt als Input den Schlüssel k und die Klartextnachricht $m \in \mathcal{M}$ und erzeugt daraus als Output den Chiffretext $c \in C$
- $Dec_k(c)$: **Entschlüsselung** (Decryption) nimmt als Input den Schlüssel k und den Chiffre-

text c und erzeugt daraus als Output den ursprünglichen Klartext m

Bemerkung 2.2.

- Für diese Schemata gilt immer die **perfekte Korrektheit** $Dec_k(Enc_k(m)) = m$
- $P(k)$ gibt die Wahrscheinlichkeit an, dass Gen den Schlüssel k erzeugt, also
- $P(m)$ gibt die Wahrscheinlichkeit an, dass die Nachricht m verwendet wird
- $P(c)$ gibt die Wahrscheinlichkeit an, dass der erzeugte Ciphertext c ist

Definition 2.3 (Angriffsszenarien)

- **Ciphertext-Only Attack:** Der Angreifer sieht nur das Chiphrat
- **Known-Plaintext Attack:** Dem Angreifer sind gewisse Klartext Ciphertext Paare bekannt
- **Chosen-Plaintext Attack:** Der Angreifer kann jeden Klartext entschlüsseln lassen
- **Chosen-Ciphertext Attack:** Der Angreifer kann gewisse Chiphretexte entschlüsseln lassen

Definition 2.4 (Principles of Modern Cryptography)

Principle 1: Formal Definition
- Definition des Systems, dessen Sicherheit, Ziele, der möglichen Angriffsszenarien und der Möglichkeiten aller Parteien, die das System nutzen

Principle 2: Precise Assumptions
- Die Annahmen zur Sicherheit eines Systems müssen präzise und nachvollziehbar sein. Beispielsweise eignen sich mathematische Probleme wie der diskrete Logarithmus oder das Faktorisierungsproblem

Principle 3: Proof of security
- Der Sicherheitsbeweis zeigt, dass das definierte System anhand der getätigten Annahmen auch wirklich sicher ist

2.2 Historische Verschlüsselungen

Definition 2.5 (Caesar Cipher)

- **GEN:** Der Schlüssel kann eine beliebige ganze Zahl $k \in \mathbb{Z}$ sein
- **ENC:** $c_i = m_i + k \mod 26$
- **DEC:** $m_i = c_i - k \mod 26$

Bemerkung 2.6.

- Sehr einfach mit Brute Force zu knacken
- Daraus kann man ableiten, dass der Schlüsselraum so groß sein muss, dass man diesen nicht einfach durch ausprobieren durchsuchen kann

Mono-Alphabetic Substitution Cipher

- **GEN:** Der Schlüssel ist eine zufällige Substitution $f(x) = y$ für jeden Buchstaben x
- **ENC:** $c_i = f(m_i)$
- **DEC:** $m_i = f^{-1}(c_i)$

Beispiel 2.7. (Schlüssel einer Substitutions Cipher)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
X	E	U	A	D	N	B	K	V	M	R	O	C	Q	F	S	Y	H	W	G	L	Z	I	J	P	T

Bemerkung 2.8.

- Ca. $26! = 2^{88}$ verschiedene Schlüssel
- Mittels Häufigkeitsanalyse entschlüsselbar

Exkurs: Häufigkeitsanalyse

In jeder Sprache kommen gewisse Buchstaben häufiger vor als andere. Kennt man diese Verteilung, kann man bei einer Monoalphabetischen Verschlüsselung die Häufigkeit aller Buch-

staben ermitteln und diese mit der Buchstabenverteilung der Ausgangssprache vergleichen. Damit können Informationen über den Schlüssel gewonnen werden.

Vigenere Cipher

- Verschlüsselungsverfahren ist ähnlich zur Monoalphabetischen Substitution
- Der Schlüssel ist ein Wort, welches solange wiederholt wird, bis die Länge des Klartextes erreicht ist
- War hunderte Jahre sicher, heutzutage aber nicht mehr (Kasiski Angriff)

Beispiel 2.9. (Vigenere Verschlüsselung)

Plaintext:	tellhimaboutme
Key:	cafecafecafeca
Ciphertext:	WFRQKJSFEPAYPF

2.3 Perfekte Sicherheit

Perfekte Sicherheit

Ein Verschlüsselungssystem mit Klartexten $m \in \mathcal{M}$ und Chiffretexten $c \in C$ ist genau dann sicher, wenn Klartext und Chiffretext voneinander unabhängig sind (mit $P(c) > 0$):

$$P(m \mid c) = P(m)$$

Daraus kann man folgern:

$$P(c \mid m) = P(c)$$

Bemerkung 2.10.

- Nach Claude Shannon: Das Verschlüsselungsverfahren ist perfekt sicher, genau dann, wenn die Wahrscheinlichkeitsverteilung auf dem Schlüsselraum die Gleichverteilung ist und es zu jedem Klartext genau einen Schlüssel und Chiffretext gibt.
- Daraus kann $P(m \wedge c) = P(m) \cdot P(c)$ gefolgert werden
- Zudem kann $P(c \mid m_0) = P(c \mid m_1)$
- Perfekte Sicherheit ist unpraktisch, da die Größe des Schlüssels ineffizient ist

One Time Pad

Sei der Schlüsseltext $c \in C$ und die Klartextnachricht $m \in \mathcal{M}$ und beides von der Länge n:

- **Gen:** Zufälliger Schlüssel: $k \in \mathcal{K} = \{0, 1\}^n$
- **Enc:** $c = m \oplus k$
- **Dec:** $m = c \oplus k$

Beweis 2.11.
Man weiß, dass $P(k) = 2^{-n}$ und damit $P(k = m\ XOR\ c) = 2^{-n}$ gilt. Damit kann man folgern:

$$P(c) = P(c = m\ XOR\ k) = P(k = m\ XOR\ c) = 2^{-n}$$

$$P(c \mid m) = P(c = m\ XOR\ k) = P(k = m\ XOR\ c) = 2^{-n}$$

(Man kann nach dem gleichen Prinzip folgern, dass $P(m) = 2^{-n}$ gilt. Dies würde den Beweis aber seiner Eleganz berauben). Anhand dieser Erkentnisse kann schließlich, unter Verwendung von Bayes'Theorem, die Gleichheit des Ausgangstheorems abgeleitet werden:

$$P(m \mid c) = \frac{P(c \mid m) \cdot P(m)}{P(c)} = \frac{2^{-n} \cdot P(m)}{2^{-n}} = P(m)$$

□

Bemerkung 2.12.

- One Time Pad ist perfekt korrekt: $Dec_k(Enc_k(m)) = Dec_k(k \oplus m) = k \oplus k \oplus m = m$
- One Time Pad erfüllt die Bedingungen für Perfekte Sicherheit

Two Time Pad

Seien die Schlüsseltexte $c_1, c_2 \in C$ und die Klartextnachrichten $m_1, m_2 \in \mathcal{M}$ von Länge n:
- **Gen:** Zufälliger Schlüssel: $k \in \mathcal{K} = \{0,1\}^n$
- **Enc:** $(c_1, c_2) = (m_1, m_2) \oplus k$
- **Dec:** $(m_1, m_2) = (c_1, c_2) \oplus k$

Beweis 2.13 (Unsicherheit von Two Time Pad).
Die perfekte Sicherheit zweier Nachrichten kann folgendermaßen definiert werden:

$$P[m_1 \wedge m_2 \mid c_1 \wedge c_2] = P[m_1 \wedge m_2] = P[m_1 \wedge m_2]$$

Man solle also zeigen, dass beide Klartexte unabhängig vom Chiffretext sind. Dafür kann man nun die erste Formel umschreiben, da offensichtlich beide Klartexte voneinander unabhängig sind:

$$P(m_1 \wedge m_2 \mid c_1 \wedge c_2) = P(m_1 \mid m_2 \wedge c_1 \wedge c_2) \cdot P(m_2 \mid c_1 \wedge c_2)$$

Damit kann man nun eine Fallunterscheidung beginnen. Betrachte man sich zuerst m_2. In diesem Fall kann die dazugehörige Formel noch vereinfacht werden, da beim One Time Pad Klartexte von Chiffretexten anderer Klartexte unabhängig sind:

$$P(m_2 \mid c_1 \wedge c_2) = P(m_2 \mid c_2) = P(m_2)$$

Der letzte Umformungsschritt ist möglich, da dies aus dem One Time Pad für eine Nachricht hervorgeht. Damit wäre dann auch für den ersten Fall die Gültigkeit bewiesen.
Der zweite Fall ist etwas komplizierter. Zuerst muss man feststellen, dass $c_1 = m_1 \oplus k$ und $c_2 = m_2 \oplus k$ anhand der arithmetischen Operationen des One Time Pads gilt. Damit kann folgender Zusammenhang abgeleitet werden $m_1 = c_1 \oplus c_2 \oplus m_1$. Setzt man dies ein so erhält man:

$$P(m_1 \mid m_2 \wedge c_1 \wedge c_2) = P(m_1 = c_1 \oplus c_2 \ \oplus \ m_2 \mid m_2 \wedge c_1 \wedge c_2)$$

Diese Wahrscheinlichkeit wertet folglich immer zu 1 aus. Es sollte aber, damit die Ursprungsaussage bewiesen wird, $P(m_1)$ herauskommen. Dies führt nämlich zu dem Problem, dass bei Einsetzen in die Ausgangsformeln folgendes herauskommt:

$$P(m_1 \wedge m_2 \mid c_1 \wedge c_2) = P(m_1 \mid m_2 \wedge c_1 \wedge c_2) \cdot P(m_2 \mid c_1 \wedge c_2) = 1 \cdot P(m_2) \neq P[m_1 \wedge m_2]$$

Damit ist also per Widerspruch bewiesen, dass Two Time Pad unsicher ist. □

Perfekte Ununterscheidbarkeit (Indistinguishability)

Ein Kryptosystem Π = (Gen, Enc, Dec) ist perfekt Ununterscheidbar, wenn für jeden Angreifer $\mathcal{A}$ gilt:

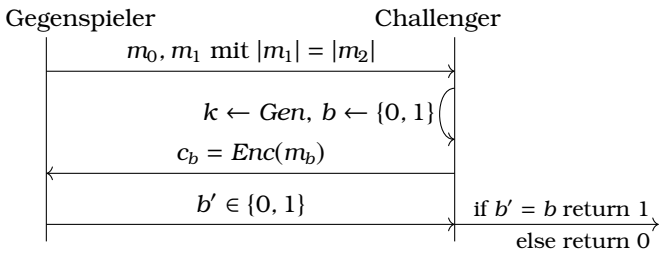
$$P[PrivK^{eav}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$$

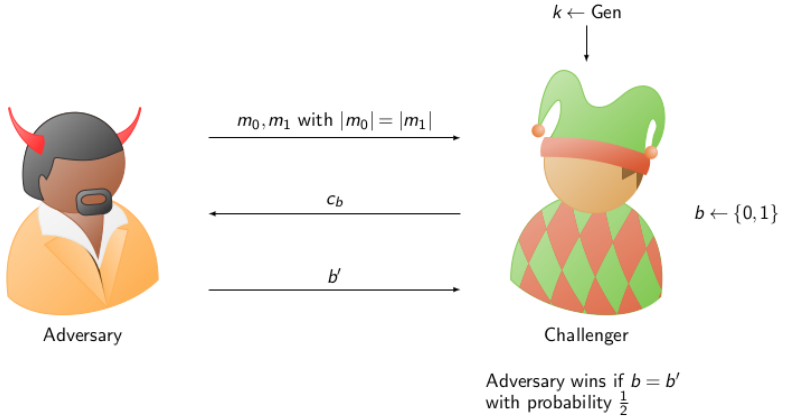
Experiment $PrivK^{eav}_{\mathcal{A},\Pi}$

Sei Π = (Gen, Enc, Dec) ein Kryptosystem mit Klartextraum $\mathcal{M}$. Sei $\mathcal{A}$ der Angreifer/Gegenspieler, welcher formell als Algorithmus angesehen werden kann. Das Experiment findet zwischen dem Angreifer und einem imaginären Herausforderer C statt:
- $\mathcal{A}$ erzeugt zwei Nachrichten $m_0, m_1 \in \mathcal{M}$ und gibt diese C
- C erzeugt einen zufälligen Schlüssel und ein zufälliges Bit $b \in \{0,1\}$. Dann erzeugt C den Chiffretext $Enc(m_b) = c$ und schickt diesen $\mathcal{A}$
- $\mathcal{A}$ verarbeitet c und schickt $b' \in \{0,1\}$ an C
- Der Ausgang des Experimentes ist 1, wenn $b' = b$ gilt. In diesem Fall hat $\mathcal{A}$ gewonnen und man schreibt $PrivK^{eav}_{\mathcal{A},\Pi} = 1$

Alternative Darstellung:

$PrivK^{eav}_{\mathcal{A},\Pi}$
$(m_0, m_1) \leftarrow \mathcal{A}, m_0, m_1 \in \mathcal{M}$
$k \leftarrow Gen$
$b \leftarrow \{0,1\}$
$c \leftarrow Enc(m_b)$
$b' \leftarrow \mathcal{A}(c)$
if $b' = b$ return 1 else return 0





Bemerkung 2.14.

- Theorem und Experiment gehören zusammen und bauen aufeinander auf
- Es gilt: Jeglicher Angreifer in diesem Experiment nicht besser sein kann, als ein Angreifer der rät
- Es gilt: Perfekte Sicherheit ⇔ Perfekte Ununterscheidbarkeit
- Zur Vereinfachung kann angenommen werden, dass die beiden Nachrichten gleich lang sind, da man ansonsten einen Spezialfall für Padding benötigt

Beispiel 2.15. (Ununterscheidbarkeit und Vigenere-Chiffre)

Anhand des aufgezeigten Experimentes kann nun Beispielsweise bewiesen werden, dass Vigenere unsicher ist. Dafür verwendet der Angreifer die Texte $m_0 = aaaaa...$ und $m_1 = adfiwofn...$, also ein Text mit beliebig vielen gleichen Buchstaben und ein Text mit beliebig vielen zufälligen Buchstaben. Der Chiffretext von m_0 sollte also, ähnlich zum Schlüssel bei Vigenere, Zyklenweise identisch sein, wohingegen m_1 weiterhin komplett zufällig ist. Dadurch kann der Angreifer die Chiffretexte auseinanderhalten und die Ununterscheidbarkeit ist nicht gegeben.

2.4 Grundideen zur berechenbaren kryptographischen Sicherheit

2.4.1 Präzisere Formulierung symmetrischer Verfahren

Definition 2.16 (Grundlegende Begrifflichkeiten und Abkürzungen)

- Ein **polynomialzeit Algorithmus** kann in polynomieller Zeit berechnet werden, d.h. die Funktion der Laufzeit abhängig von der Inputgröße wächst nicht wesentlicher schneller als ein Polynom (Landau Symbol O-Notation)
- Ein **probabilistischer polynomialzeit Algorithmus** berechnet für jede zufällige Eingabe in polynomieller Zeit das Ergebnis

Definition 2.17 (Grundschemata Berechenbarer Symmetrische Verschlüsselung)

Ein **Private Key Encryption** Schemata $\Pi = (Gen, Enc, Dec)$ wird über einen Klartextraum $\mathcal{M} = \{0,1\}^*$ anhand von drei probablistischen polynomialzeit Algorithmen definiert:

- $k \leftarrow Gen(1^n)$: **Schlüsselgenerierung** (Key-Generation) erzeugt als Output einen Schlüssel k anhand einer gewissen Wahrscheinlichkeitsverteilung aus dem Schlüsselraum $\mathcal{K}$ und

kann dafür einen Inputparameter 1^n verwenden
- $c \leftarrow Enc_k(m)$: **Verschlüsselung** (Encryption) nimmt als Input den Schlüssel k und die Klartextnachricht $m \in \mathcal{M}$ und erzeugt daraus als Output den Chiffretext $c \in C$
- $m \leftarrow Dec_k(c)$: **Entschlüsselung** (Decryption) nimmt als Input den Schlüssel k und den Chiffretext c und erzeugt daraus als Output den ursprünglichen Klartext m

Bemerkung 2.18.

- Für diese Schemata gilt immer die **perfekte Korrektheit** $Dec_k(Enc_k(m)) = m$
- Der große Unterschied zur vorhergehenden Definition ist, dass hierbei nun auf $P =? \neq NP$ miteinbezogen wird

2.4.2 Vernachlässigbarkeit

Vernachlässigbare Funktionen

Eine Funktion f ist **vernachlässigbar** wenn es zu jedem Polynom p ein N gibt, sodass für alle Zahlen n mit $n > N$ gilt:

$$f(n) < \frac{1}{p(n)}$$

Beispiel 2.19. (Vernachlässigbare Funktionen)

(a) $\frac{1}{2^x}$: Vernachlässigbar, da 2^x exponentiell ist

(b) $\frac{1}{x^{100}}$: Nicht Vernachlässigbar, da x^{100} nicht exponentiell ist. In anderen Worten: Sei $p(x) = x^{101}$. Setzt man dies ins Theorem ein so erhält man:

$$f(x) = \frac{1}{x^{100}} \not< \frac{1}{x^{101}}$$

(c) $h(x)$ mit $h(x) < f(x) \forall x$. Offensichtlich ist das Vernachlässigbar:

$$h(x) < f(x) < \frac{1}{p(x)} \Rightarrow h(x) < \frac{1}{p(x)}$$

(d) $\frac{1}{2^x} + \frac{1}{7^x}$: Vernachlässigbar, da Addition vernachlässigbarer Funktionen

(e) $\frac{1}{2^x} \cdot \frac{1}{x^7}$: Vernachlässigbar, da Verknüpfung einer vernachlässigbaren Funktion mit positivem Polynom.

(f) $\frac{f(x)}{g(x)}$ Nicht Vernachlässigbar. Gilt nämlich $f(x) = g(x)$ so ist das Ergebnis immer 1, also eine Konstante.

(g) 2^{-100}: Nicht Vernachlässigbar, da es ein Konstanter Wert ist.

Bemerkung 2.20.

- Vernachlässigbare Funktionen werden auch als $negl(n)$ geschrieben
- Es wurde sich hier für den analytischen Ansatz, statt den stochastischen Ansatz, entschieden, da man damit besser Unterschiede bei verschieden Leistungsstarken Rechenmaschinen abdecken kann
- Jede Funktion die exponentiell gegen null geht ist vernachlässigbar: Gilt $f(n) = 1/f'(n)$ und ist $f'(n)$ exponentiell, so ist $f(x)$ vernachlässigbar: Polynome wachsen nämlich nie schneller als exponentielle Funktionen
- Konstante Werte sind nicht vernachlässigbar: $p(n) = n$ ist ein Gegenbeispiel anhand des Theorems
- Seien $f_1(x)$ und $f_2(x)$ vernachlässigbare Funktionen, dann ist $f_3(x) = f_1(x) + f_2(x)$ vernachlässigbar: Es gibt ein $x > N_{2p}$ sodass $f(x) < \frac{1}{2p(x)}$ und ein $x > N'_{2p}$, sodass $g(x) < \frac{1}{2p(x)}$ gilt. Daraus folgt $f(x) + g(x) < \frac{1}{p(x)}$ für alle $x > max\{N_{2p}, N'_{2p}\}$
- Sei $f_1(x)$ eine vernachlässigbare Funktion und $p(x)$ ein positives Polynom, dann ist

$f_2(x) = p(x) \cdot f_1(x)$ vernachlässigbar: Sei $q(x)$ ein Polynom. Dann existiert ein N_{pq}, sodass $f(x) < \frac{1}{p(x)q(x)}$ für alle $x > N_{pq}$ gilt. Dies kann man zu $f(x)q(x) < \frac{1}{p(x)}$ umformen.

2.4.3 Computell berechenbare Ununterscheidbarkeit

Berechenbare Ununterscheidbarkeit (Indistinguishability)

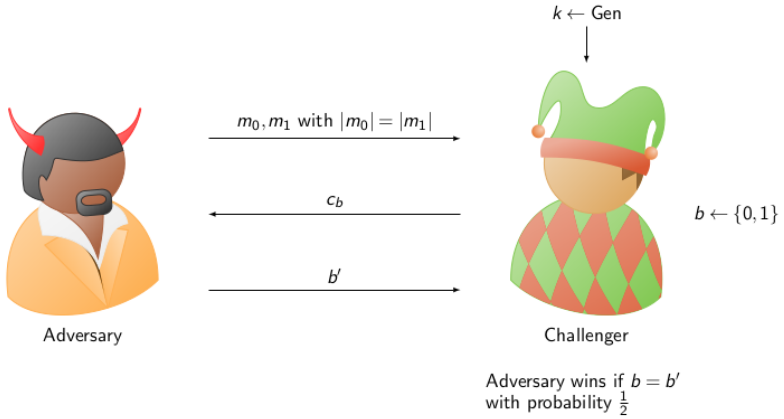
Ein Kryptosystem Π = (Gen, Enc, Dec) hat eine ununterscheidbare Verschlüsselung, wenn für jeden PPT (propabilistischen polynomialzeit) Angreifer $\mathcal{A}$ eine vernachlässigbare Funktion existiert, sodass gilt:

$$P[PrivK^{eav}_{\mathcal{A},\Pi} = 1] \leq \frac{1}{2} + negl(n) \forall n \in \mathbb{N}$$

Anmerkung: Die Wahrscheinlichkeit wird über die Zufälligkeit von $\mathcal{A}$ und die des Experimentes berechnet.

Experiment $PrivK^{eav}_{\mathcal{A},\Pi}(n)$

Dieses Experimant ist wieder mit einem Gegenspieler $\mathcal{A}$ und einem Challenger:



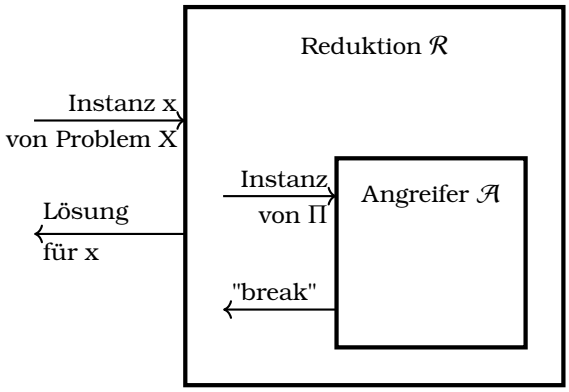
$PrivK^{eav}_{\mathcal{A},\Pi}(n)$
$(m_0, m_1) \leftarrow \mathcal{A}(1^n), m_0, m_1 \in \mathcal{M}, \lvert m_0\rvert = \lvert m_1\rvert$
$k \leftarrow Gen(1^n)$
$b \leftarrow \{0,1\}$
$c \leftarrow Enc(m_b)$
$b' \leftarrow \mathcal{A}(c)$
if $b' = b$ return 1 else return 0

Man schreibt $PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1$ wenn der Output des Experimentes 1 ist und $\mathcal{A}$ gewonnen hat.

Bemerkung 2.21.

- **Abkürzung:** ING-EAV-Single-Sicherheit (indistinguishable single encryption eavesdropper security) bzw. EAV-Single-Sicherheit
- Das ist die grundlegenste Definition von berechenbarer Sicherheit
- Damit wird die Sicherheit vor Ciphertext Only Angriffen definiert, wobei der Gegenspieler nur einen einzigen Ciphertext sieht, bzw. wenn für jede Nachricht ein neuer Schlüssel erzeugt wird
- Der Unterschied von $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ zu $PrivK^{eav}_{\mathcal{A},\Pi}$ ist, dass nur polynomialzeit Algorithmen, Nachrichten gleicher Länge und ein zusätzlicher Sicherheitsparameter betrachetet werden
- Gilt perfekte Ununterscheidbarkeit, so folgt daraus auch berechenbare Ununterscheidbarkeit

2.4.4 Grundidee des Reduktionsbeweises



Bemerkung 2.22. (Theoretische Vorgehensweise eines Reduktionsbeweises)

1. Ein zugrundeliegendes Problem, hier X, wird als schwierig angesehen und kann daher von einem PPT Algorithmus nur mit vernachlässigbarer Wahrscheinlichkeit effizient gelöst werden
2. Wir nehmen an, dass ein Kryptosystem Π nicht sicher ist und daher ein effizienter Angreifer $\mathcal{A}$ gegen dieses Kryptosystem existiert. Wie dieser funktioniert ist egal. Nehme man zudem an, dass $\mathcal{A}$ Π mit nicht vernachlässigbarer Wahrscheinlichkeit $\epsilon(n)$ brechen kann
3. Man konstruiert einen Angreifer $\mathcal{A}'$, auch Reduktion $\mathcal{R}$ genannt, welcher Problem X anhand von $\mathcal{A}$ (Black Box Prinzip) lösen kann. Dafür übergibt $\mathcal{A}'$ geschickt die Daten von x (indem man z.b. als Challenger bei $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ agiert), bzw. Abwandlungen davon, an $\mathcal{A}$ und versucht aus dessen Output x zu lösen. $\mathcal{A}'$ sollte anhand des Output x mit mind. einer invertierten polynomiellen Wahrscheinlichkeit $1/p(n)$ brechen
4. Daraus kann man folgern, dass $\mathcal{A}'$ das Problem X mit Wahrscheinlichkeit $\epsilon(n)/p(n)$ löst, welche folglich auch nicht vernachlässigbar ist. Dadurch hat $\mathcal{A}'$ etwas geschafft, was wir als unmöglich angenommen haben
5. Abschließend stellt man also fest, dass es sich um einen Widerspruch zu der definierten Annahme handelt, und damit das Schemata Π sicher ist

2.5 Pseudorandomgeneratoren

Theorem 2.23 (Pseudorandomness)

Eine Verteilung $\mathcal{D}$ über eine Menge von Zeichenketten der Länge l ist **pseudorandom**, wenn $\mathcal{D}$ ununterscheidbar zu einer gleichverteilten (echt zufälligen) Menge von Zeichenketten der Länge l ist.

Bemerkung 2.24.

- Ein einzelnes Element kann zufällig sein, sondern nur eine Menge an Elementen
- Ich übersetze in dieser Arbeit *uniform* mit *echtzufaellig*
- In der echten Welt gibt es (*fast*) keinen perfekten Zufall, welcher zu einer Gleichverteilung führen würde. Es gibt immer Faktoren, egal wie minimal, die das Ergebnis verfälschen und damit "nur" pseudorandom machen (z.b. hat jeder Würfel, aber einer gewissen Genauigkeit Herstellungsmängel und ist damit nur Pseudorandom. Das Argument brauch ich beim nächsten Pen and Paper Abend...)

Definition 2.25 (Pseudorandom Generator)

Sei p ein beliebiges Polynom und G ein deterministischer polynomialzeit Algorithmus, welcher für die Eingabe $s \in \{0,1\}^n$ mit $n \in \mathbb{N}\backslash\{0\}$ eine Zeichenkette der Länge $l(n)$ ausgibt. G ist ein **Pseudorandom Generator PRG**, falls folgende Eigenschaften erfüllt sind:

- **Expansion:** Für jedes n gilt $l(n) > n$
- **Pseudorandomness:** Für einen stochastischen polynomialzeit berechenbaren Unterschei-

der D gilt:

$$| Pr[D(r) = 1] - Pr[D(G(s)) = 1] | \leq negl(n)$$

Dabei wird $Pr[D(r) = 1]$ anhand der gleichverteilen (echten zufälligen) Wahl von $r \in \{0,1\}^n$ und anhand der Zufallsprinzip von D berechnet. Die zweite Wahrscheinlichkeit wird anhand der gleichverteilen (echten zufälligen) Wahl von $s \in \{0,1\}^n$ und dem Zufallsprinzip von D berechnet.

Beispiel 2.26.

Sei $G : \{0,1\}^n \to \{0,1\}^{n+1}$ ein pseudorandom Generator. Sind die folgenden darauf aufbauenden Generatoren auch pseudorandom?

- $G_1(x) := G(x)||1$: Nein. Bekommt der Unterscheider D eine pseudozufällige Zeichenkette, so hat diese immer eine 1 am Ende und D kann eine eins ausgeben, wodurch er mit 100% richtig liegt. Bekommt D nun eine echt zufällige gibt er bei einer 1 am Ende wieder 1 aus, aber bei einer 0 am Ende schließlich 0. Damit hat er eine 50% Erfolgswahrscheinlichkeit:

$$| Pr[D(r) = 1] - Pr[D(G(s)) = 1] | = \left| \frac{1}{2} - 1 \right| = \frac{1}{2} \not\leq negl(n)$$

- $G_2(x) = G(x)||G(x)$: Ähnliche Argumentation wie G_1
- $G_3(x||b) = G(s)||b$ mit $|b| = 1$: G_3 ist auch weiterhin ein PRG. Zuersteinmal sind die nötigen Bedingungen (determinismus, polynomialzeit, Expansion) trivialerweise erfüllt. Für die Pseudorandomness nutzt man einen Reduktionsbeweis:
  1. Grundlegende Annahme: G ist ein PRG.
  2. Annahme: G_b ist kein PRG und es gibt einen Unterscheider D, der den Output von G_b besser von einer echt zufälligen Zeichenkette unterscheiden kann, als mit vernachlässigbarer Wahrscheinlichkeit
  3. Nun will man einen Unterscheider D' bauen, der anhand von D den PRG G unterscheiden kann: D' bekommt den Input s, hängt da ein echt zufälliges $b \leftarrow \{0,1\}$ dran (das ist nötig, da D einen string benötigt, der ein Bit länger ist). D' gibt nun das aus, was D ausgeben würde
  4. Dadurch entstehen zwei Fälle: 1. D' bekommt eine echt zufällige Zeichenkette, wandelt diese in eine weitere echt zufällige Zeichenkette um, D rät und damit auch D'. 2. Bekommt D' eine Zeichenkette von G_b, so ist die durch anhängen weiterhin pseudorandom, D erkennt dies und damit auch D', wodurch in dem Fall gilt (wenn in diesem Fall 1 zurückgegeben wird):

$$| Pr[D(r) = 1] - Pr[D(G(s)) = 1] | = \left| \frac{1}{2} - 1 \right| = \frac{1}{2} \not\leq negl(n)$$

  5. Somit könnte man einen Unterscheider D' entwickeln, der G mit nicht vernachlässigbarer Wahrscheinlichkeit unterscheidet, was einen Widerspruch darstellt
- $G_4(x) = G(x||0)$: Ein Gegenbeispiel hierfür wäre der PRG G_3. Wird dieser verwendet, ist am Ende immer eine 0 und die gleiche Argumentation wie für G_1 kann verwendet werden

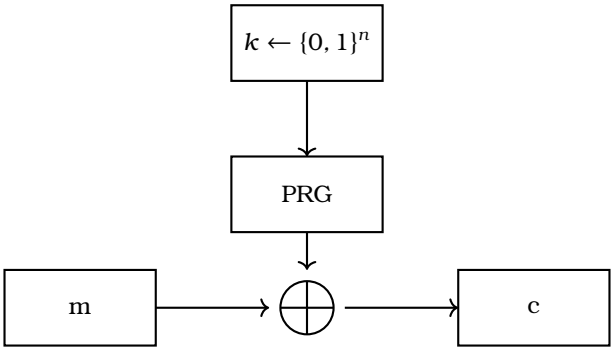
---

Bemerkung 2.27.

- In anderen Worten: Ein Pseudorandom Generator nutzt eine kleine zufällige Zeichenkette um daraus eine größere pseudozufällige Zeichenkette zu erzeugen
- PRG sind praktisch, da die Erzeugung von echten Zufallszahlen aufwändig ist, und man damit aus wenigen Zufallszahlen viele machen kann
- **Erklärung des Unterscheiders:** Der Unterscheider D bekommt einen String s (dieser ist echt zufällig oder pseudo zufällig). D verarbeitet die Eingabe und gibt 1 zurück, wenn er denkt s ist echt zufällig und 0, wenn er denkt s ist pseudorandom. (Die Ausgabe 1 und 0 kann auch vertauscht werden). Mit $Pr[D(G(s)) = 1]$ wird die Wahrscheinlichkeit gegeben, dass D 1 ausgibt, wenn er ein pseudozufälliges s bekommt. Mit $Pr[D(s) = 1]$ wird die Wahrscheinlichkeit gegeben, dass D 1 ausgibt, wenn er ein echt zufälliges s bekommt.
- Um zu beweisen, dass der PRG nicht funktioniert versucht man einen Unterscheider

zu konstruieren, welcher besser als mit vernachlässigbarer Wahrscheinlichkeit richtig entscheiden kann
- Um die Sicherheit zu beweisen nutzt man einen Reduktionsbeweis, insofern der neue PRG auf einem richtigen PRG (oder etwas vergleichbarem) aufbaut
- In der Praxis verwendet man teilweise Lineare Kongruenzgleichungen oder logistische Funktionen als PRG, da diese effizient zu berechnen sind. Diese eignen sich aber nicht für kryptographische Anwendungen

2.6 Symmetrische Verschlüsselung mit fester Länge anhand von PRG



Definition 2.28 (Symmetrische Verschlüsselung anhand von PRG)

Sei G ein pseudorandom Generator PRG mit Expansionsfaktor $l(n)$. Damit kann ein symmetrisches Verfahren Π_{PRG} für Nachrichten der Länge $l(n)$ definiert werden:
- $Gen(1^n)$: Schlüssel $k \leftarrow \{0,1\}^n$ wird mit gleichverteilter Wahrscheinlichkeit erzeugt
- $Enc_k(m)$: Nachrichten $m \in \{0,1\}^{l(n)}$ werden mit $c = m \oplus G(k)$ verschlüsselt
- $Dec_k(c)$: Erzeugt den Klartext $m = c \oplus G(k)$

Satz 2.29 (Ununterscheidbarkeit des PRG Kryptosystems)

Sei G ein Pseudorandom Generator, dann ist Π_{PRG} ein symmetrisches Verschlüsselungsverfahren fester Länge, welches berechenbar Ununterscheidbar gegenüber einem abhörendem Angreifer ($PrivK^{eav}_{\mathcal{A},\Pi}(n)$) ist.

Beweis 2.30.
Grundidee des Beweises ist eine Reduktion:
- Grundlegendes schweres Problem: G ist ein PRG und es gibt daher keinen Unterscheider D, welcher effizient und gut ist
- Annahme: Π_{PRG} ist unsicher, d.h. es existiert ein Angreifer $\mathcal{A}$ welcher im Experiment $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ besser als mit vernachlässigbarer Wslk. unterscheiden kann, um welche Nachricht es sich handelt
- Damit kann man einen Unterscheider D konstruieren, welcher G effizient unterscheidet:
  1. D bekommt als Input einen String $w \in \{0,1\}^{l(n)}$
  2. D starte das $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ mit A und erhält $m_0, m_1 \in \{0,1\}^{l(n)}$
  3. D wählt zufällig ein $b \in \{0,1\}$ und erzeugt $c = w \oplus m_b$
  4. D gibt c an A und erhält b'. Der Output von D ist 1 falls $b' = b$, ansonsten 0
- Es entstehen zwei Fälle:
  1. $w \in \{0,1\}^{l(n)}$ wird echtzufällig gewählt und A muss versuchen ein One-Time-Pad zu unterscheiden, was durch informationstheoretische Sicherheit unmöglich ist, und damit A und folglich auch D eine Erfolgswahrscheinlichkeit von $1/2$ haben
  2. $w \in \{0,1\}^{l(n)}$ wird vom PRG erzeugt. A kann dies besser als mit vernachlässigbarer Wslk. unterscheiden, wodurch auch die Ergebnisse von D besser als mit vernachlässigbarer Wslk. richtig sind
- **Fazit:** Da das grundlegende Problem (G ist ein PRG) dadurch gebrochen wurde entsteht ein Widerspruch. Dieser Widerspruch beweist, dass die Sicherheit von G auch die Sicherheit von Π_{PRG} impliziert

□

2.7 Sicherheit bei mehrfacher Verschlüsselung mit gleichem Schlüssel

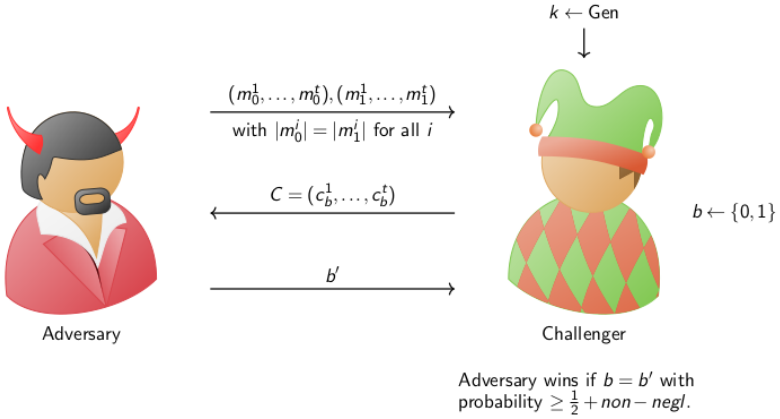
Definition 2.31 (Sicherheit für mehrfache Verschlüsselung mit gleichem Schlüssel)

Ein Kryptosystem Π hat mehrfach ununterscheidbare Verschlüsselung gegenüber einem Abhörer, wenn für jeder probabilistische polynomialzeit Angreifer $\mathcal{A}$ eine vernachlässigbare Funktion $negl(n)$ existiert, sodass gilt:

$$P[PrivK^{mult}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

Experiment $PrivK^{mult}_{\mathcal{A},\Pi}(n)$

Dieses Experimant ist wieder mit einem Gegenspieler $\mathcal{A}$ und einem Challenger:



$PrivK^{mult}_{\mathcal{A},\Pi}(n)$
$(m_0^1, ..., m_0^t, m_1^1, ..., m_1^t) \leftarrow \mathcal{A}(1^n), \lvert m_0^i\rvert = \lvert m_1^i\rvert \forall i \in [1,t]$
$k \leftarrow Gen(1^n)$
$b \leftarrow \{0,1\}$
$C = (c_b^1, ..., c_b^t) \leftarrow (Enc_k(m_b^1), ..., Enc_k(m_b^t))$
$b' \leftarrow \mathcal{A}(C)$
if $b' = b$ return 1 else return 0

Man schreibt $PrivK^{mult}_{\mathcal{A},\Pi}(n) = 1$ wenn der Output des Experimentes 1 ist und $\mathcal{A}$ gewonnen hat.

Bemerkung 2.32.

- **Abkürzung:** EAV-Mult-Sicherheit (indistinguishable multiple encryption eavesdropper security) bzw. EAV-Mult Sicherheit
- **Vergleich zu de vorherigen Sicherheitsdefinitionen:** Diese Sicherheitsdefinition ist stärker als die vorrangegangene Definition, bei der nur eine Nachricht verschlüsselt und unterschieden wurde, da EAV-Single ein Spezialfall von EAV-Mult ist
- **Konstruktion:** Ein deterministische Kryptosystem kann die Sicherheit für mehrfache Verschlüsselung niemals erfüllen: Würde man hierbei als Gegenspieler $M_0 = (0^n, 0^n)$ und $M_1 = (0^n, 1^n)$ für $PrivK^{mult}_{\mathcal{A},\Pi}(n)$ wählen, so könnte man die Verschlüsselungen immer unterscheiden
- **Folgerung:** Kryptosysteme für mehrfache Verschlüsselungen dürfen nicht deterministisch sein und müssen Zufallsvariablen enthalten

2.8 Sicherheit gegen Chosen-Plaintext Attacks

Definition 2.33 (Sicherheit gegen Chosen Plaintext Attacks)

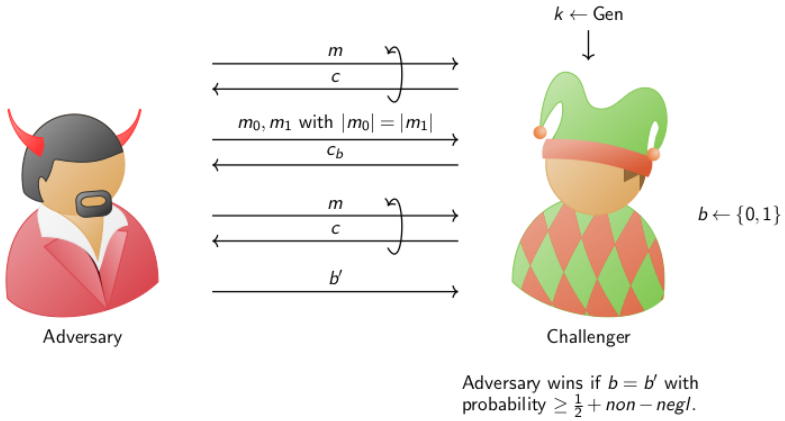
Ein Kryptosystem Π hat ununterscheidbare Verschlüsselung gegenüber einem Chosen Plaintext Attack, auch CPA-Secure bzw. CPA-sicher genannt, wenn für alle probabilistischen polynomial-

zeit Angreifer $\mathcal{A}$ eine vernachlässigbare Funktion $negl(n)$ existiert, sodass gilt:

$$P[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

Experiment $PrivK^{cpa}_{\mathcal{A},\Pi}(n)$

Dieses Experimant ist wieder mit einem Gegenspieler $\mathcal{A}$ und einem Challenger:



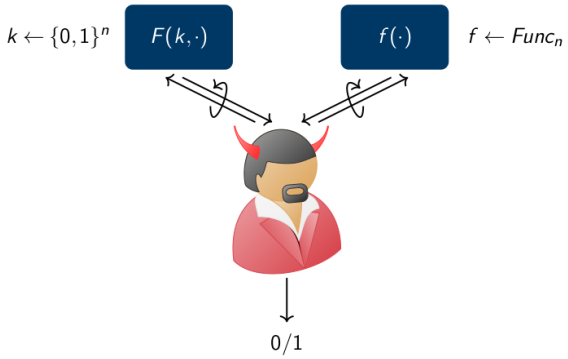
$PrivK^{cpa}_{\mathcal{A},\Pi}(n)$
$k \leftarrow Gen(1^n)$
$(m_0, m_1) \leftarrow \mathcal{A}^{Enc_k(\cdot)}, \mid m_0 \mid = \mid m_1 \mid$
$b \leftarrow \{0,1\}$
$c_b \leftarrow Enc_k(m_b)$
$b' \leftarrow \mathcal{A}^{Enc_k(\cdot)}(c_b)$
if $b' = b$ return 1 else return 0

Mit $\mathcal{A}^{Enc_k(\cdot)}$ ist gemeint, dass der Gegenspieler $\mathcal{A}$ Zugang zu einem Verschlüsselungsorakel hat, welches beliebige Nachrichten von $\mathcal{A}$ mit dem Schlüssel des Challengers k verschlüsselt. $\mathcal{A}$ kann das Orakel so oft nutzen, wie er will. Man schreibt $PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1$ wenn der Output des Experimentes 1 ist und $\mathcal{A}$ gewonnen hat.

Bemerkung 2.34.

- **Abkürzung:** Diese Sicherheit wird auch als ING-CPA-Sicherheit (Indistinguishable chosen plaintext attack secure) bzw. CPA-Sicherheit bezeichnet
- **Erinnerung:** Chosen Plaintext Angriffe ermöglichem einen Angreifer die Verschlüsselung beliebiger Nachrichten mit dem Verfahren, welches die Angegriffenen Parteien nutzen
- **Anmerkung Einfache und Mehrfache Verschlüsselung:** Ein Kryptosystem das CPA-Sicher für einfache Verschlüsselung mit gleichem Schlüssel ist, ist auch CPA-Sicher für mehrfache Verschlüsselung mit gleichem Schlüssel und andersrum, d.h. (ING-CPA-Single-Secure ⇔ ING-CPA-Mult-Secure) = ING-CPA-Secure!
- **Vergleich zu den vorherigen Sicherheitsdefinitionen:** Da das ING-EAV-Mult und ING-EAV-Single Experiment im CPA-Experiment enthalten sind, kann man per Reduktion zeigen, dass CPA-Sicherheit auch ING-EAV-Mult-Sicherheit und ING-EAV-Single-Sicherheit impliziert (Andersum gilt dies nicht). D.h. auch, dass CPA-Sicherheit stärker als die vorherigen Sicherheitsdefinitionen ist
- **Anmerkung:** Diese Definition stellt das Minimum an Sicherheit dar, welches heute für Kryptosysteme gelten sollte

2.9 Pseudorandom Funktionen



Definition 2.35 (Pseudorandom Funktionen PRF)

Sei $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ eine effiziente, längenerhaltende, schlüsselabhängige Funktion. F ist eine pseudorandom Funktion PRF, wenn für alle probabilistischen polynomialzeit Unterscheider D eine vernachlässigbare Funktion $negl(n)$ existiert, sodass gilt:

$$| P[D^{F(k,\cdot)}(1^n) = 1] - P[D^{f(\cdot)}(1^n) = 1] | \leq negl(n)$$

Dabei wird $P[D^{F(k,\cdot)}(1^n) = 1]$ anhand der gleichverteilen (echten zufälligen) Wahl von $k \in \{0,1\}^n$ und anhand der Zufallsprinzip von D berechnet. Die zweite Wahrscheinlichkeit wird anhand der gleichverteilen (echten zufälligen) Wahl von $f \in Func_n$ und dem Zufallsprinzip von D berechnet.

Beispiel 2.36.

Sei $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ eine PRF. Sind die folgenden Konstruktionen auch Pseudorandom Funktionen? (Längenerhaltung wird Ignoriert!)

- $F'_k(x) = F_k(x)||0$: Keine PRF, ein Unterscheider D gibt ein beliebiges x an das Orkale und erhält y und gibt einfach das letzte Bit von y aus.
- $F'_k(x) = F_k(x \oplus 1^n)$: Ja, $F'_k(x)$ ist weiterhin eine PRF. Beweis durch Reduktion:
  1. Grundlegende Annahme: F_k ist eine PRF
  2. Annahme: F'_k ist keine PRF und es gibt einen Unterscheider D, der den Output von F'_k besser von einer echt zufälligen Zeichenkette unterscheiden kann, als mit vernachlässigbarer Wahrscheinlichkeit
  3. Konstruktion von D' um F_k anhand von D zu unterscheiden: D' nimmt eine beliebige Zeichenkette (Länge n) s berechnet damit $s' = s \oplus 1^n$. s' gibt D' nun an sein Orakel, welches $x = F_k(s') = F_k(s \oplus 1^n)$ berechnet. Dann gibt D' x an D und folglich gibt D' das Ergebnis von D aus
  4. Es entstehen zwei Fälle: 1. D' bekommt eine echt zufällige Funktion als Orakel und kann daher nur mit einer Wslk. von $1/2$ anhand von D den richtigen Output generieren. 2. D' bekommt die Funktion $F_k(x)$ als Orakel, welche D folglich besser als vernachlässigbarer Wslk. unterscheiden kann, wodurch auch D' die Eingabe besser als mit vernachlässigbarer Wslk. unterscheiden kann
  5. Dadurch entsteht ein Widerspruch und F_k ist eine PRF
- $F'_k(x) = F_k(x)||F_k(x \oplus 1^n)$: Keine PRF, ein Unterscheider D gibt zuerst $m_0 = 0^n$ an das Orakel und erhält $y_0^0||y_0^1$, danach gibt er $m_1 = 1^n$ an das Orakel und erhält $y_1^0||y_1^1$. Bei F'_k gilt immer $y_0^0 = y_1^1$ bzw. $y_0^1 = y_1^1$ bei einer echt zufälligen Funktionen nur mit vernachlässigbarer Wslk.
- $F'_k(x_1||x_2) = F_k(x_1)||F_k(x_2)$ mit $x_1, x_2 \in \{0,1\}^n$: Kein PRF, ein Unterscheider D gibt dem Orakel $x||x$ und erhält $y_1||y_2$. Bei F'_k gilt immer $y_1 = y_2$ bei einer echt zufälligen Funktionen nur mit vernachlässigbarer Wslk.
- $F'_k(x) = F_k(0||x)||F_k(1||x)$ mit $x \in \{0,1\}^{n-1}$: Ja, $F'_k(x)$ ist weiterhin eine PRF. Grund: F_k hat immer unterschiedliche Eingaben, wodurch der Output immer pseudozufällig ist. Beweis durch Reduktion:
  1. Grundlegende Annahme: F_k ist eine PRF

2. Annahme: F'_k ist keine PRF und es gibt einen Unterscheider D, der den Output von F'_k besser von einer echt zufälligen Zeichenkette unterscheiden kann, als mit vernachlässigbarer Wahrscheinlichkeit
3. Konstruktion von D' um F_k anhand von D zu unterscheiden: D' gibt seinem Orakel zuerst $0||x$ mit und erhält a. Danach gibt er dem Orakel $1||x$ mit und erhält b. x kann beliebig gewählt von Länge n sein. D' gibt D nun $a||b$ mit und anschließend gibt D' den Output von D aus
4. Es entstehen zwei Fälle: 1. D' bekommt eine echt zufällige Funktion als Orakel und kann daher nur mit einer Wslk. von $1/2$ anhand von D den richtigen Output generieren. 2. D' bekommt die Funktion $F_k(x)$ als Orakel, welche D folglich besser als vernachlässigbarer Wslk. unterscheiden kann, wodurch auch D' die Eingabe besser als mit vernachlässigbarer Wslk. unterscheiden kann
5. Dadurch entsteht ein Widerspruch und F_k ist eine PRF

- $F'_k(x) = F_k(x||0)||F_k(x||1)$ mit $x \in \{0,1\}^{n-1}$: Ja, ähnliche Argumentation wie bei der vorherigen Aufgabenstellung

- $F'_k(x) = F_k(0||x)||F_k(x||1)$ mit $x \in \{0,1\}^{n-1}$: Nein, da man gleiche Inputs erzeugen kann. Ein Unterscheider D erstellt zuerst eine Zeichenkette $m_0 = 0^{n-1}||1$, gibt die dem Orakel und erhält eine Zeichenkette der Form $m_res0 = a||b$. Dann erstellt D die Zeichenkette $m_1 = 0^n$ und gibt die dem Orakel und bekommt immer $m_res1 = c||a$, insofern er mit der Pseudozufälligen Funktion interagiert. Wenn er mit der echt zufälligen Funktion interagiert, erhält D dies nur mit vernachlässigbarer Wslk.

Bemerkung 2.37.

- $Func_n$ ist die Menge aller Funktionen, die ein n-bit Eingabe auf ein n-bit Ausgabe projeziert
- Typischerweiße wird für den Nachweis ein key k echt zufällig gewählt und die zu Prüfende Funktion mit k festgesetzt (Für einen Durchlauf mit dem Experiment). Der Unterscheide bekommt niemals den key
- Anstatt nun also Zufällig aussehende Zeichenkette zu betrachten, wie es bei PRG der Fall war, betrachtet man sich hier zufällig aussehende Funktionen. D.h. Ein Unterscheider bekommt eine Funktion, hier auch Orakel genannt, mit der er beliebig arbeiten kann. Das Orakel ist aber deterministisch, d.h. für die gleiche Eingabe gibt es den gleichen Output. Das Ziel ist zu unterscheiden, ob diese Funktion echt zufällig oder die konstruierte zu testende pseudozufällige Funktion ist. Kann ein Angreifer dies nicht, so unterscheiden, so ist die konstruierte Funktion wirklich pseudozufällig, ansonsten nicht
- **Anmerkung Längenerhaltend:** Dies wird nur zur Vereinfachung angenommen und ist nicht zwingend notwendig

Satz 2.38 (Existenz von Pseudorandom Funktionen)

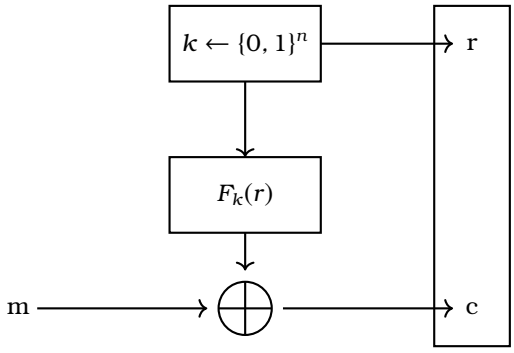
Pseudorandom Funktionen existieren genau dann wenn Pseudoranom Generatoren existieren.

Beweis 2.39.

- Beweisidee: Aus PRF kann man PRG erzeugen und andersrum
- **1. Fall PRG mittel PRF:** Sei F_k eine PRF, dann ist $G(k) = F_k(0^n)||F_k(1^n)$ ein PRG
- **2. Fall PRF mittel PRG:** Möglich mit Goldreich Goldwasser Micali Konstruktion. Diese wird später genauer erläutert

□

2.10 Verschlüsselung mittels PRF mit Sicherheit vor CPA



Definition 2.40 (Symmetrische Verschlüsselung anhand von PRF)

Sei F_k ein pseudorandom Funktion PRF. Damit kann ein symmetrisches Verfahren Π_{PRF} für Nachrichten der Länge n definiert werden:

- $Gen(1^n)$: Schlüssel $k \leftarrow \{0,1\}^n$ wird mit gleichverteilter Wahrscheinlichkeit erzeugt
- $Enc_k(m)$: Wähle $r \leftarrow \{0,1\}^n$ mit gleichverteilter Wahrscheinlichkeit und berechne $c = (r,s) = (r, F_k(r) \oplus m)$
- $Dec_k(c)$: Erhalte $c = (r,s)$ und berechne $m = F_k(r) \oplus c$

Satz 2.41 (CPA-Sicherheit von Π_{PRF})

Ist F eine pseudorandom Funktion, dann ist Π_{PRF} sicher vor Chosen-Plaintext Angriffen.

Beweis 2.42.

- Grundlegende Annahme: F_k ist eine PRF
- Annahme: Π_{PRF} ist unsicher, d.h. es gibt einen Angreifer A der das CPA-Experiment besser als mit vernachlässigbarer Wslk gewinnt:

$$P[P^{cpa}_{\Pi,A}(1^n) = 1] = \frac{1}{2} + \epsilon$$

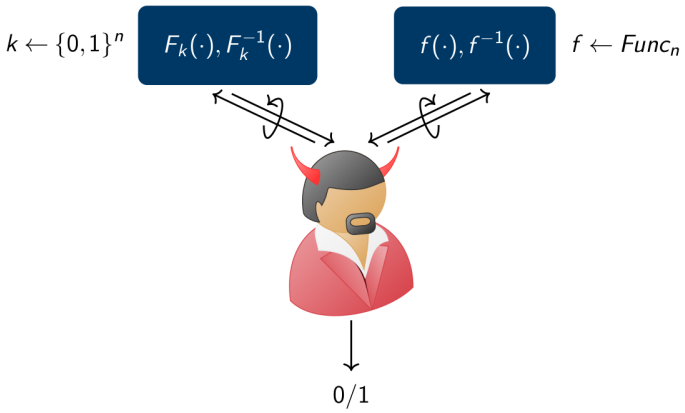
- **Einschub (Kurze genauere Analyse von A):** Kommuniziert A mit einer echt zufälligen Funktion, bzw. einer echten PRF, so kann A im Zuge des Experimentes mehrfach, also $q(n)$-mal, Nachrichten entschlüsseln lassen. Bei jeder Entschlüsselung hat A die Wahrscheinlichkeit $\frac{1}{2^n}$ das CPA-Experiment mit Wslk. 1 zu gewinnen. Daraus folgt:

$$P[P^{cpa}_{\Pi,A}(1^n) = 1] = \frac{1}{2} + \frac{q(n)}{2^n}$$

- Konstruktion von R_A um F_k anhand von A zu unterscheiden: R_A bekommt entweder F_k oder eine echt zufällige Funktion gegeben. Mit dieser kann R_A das Kryptosystem Π_{PRF} simulieren. Damit kann er als Challenger im CPA-Experiment agieren und A als Gegenspieler nutzen. R_A gibt anschließend den Output von A aus. Es entstehen zwei Fälle:
  1. R_A bekommt eine echt zufällige Funktion und simuliert das CPA-Experiment. In diesem Fall ist das Kryptosystem im CPA-Experiment informationstheoretisch sicher. A kann dieses also nur mit Wslk $\frac{1}{2} + \frac{p(n)}{2^n}$ unterscheiden
  2. R_A bekommt eine echt zufällige Funktion und simuliert das CPA-Experiment. A kann dies nun mit Wslk $\frac{1}{2} + \epsilon$ unterscheiden
- Damit kann nun auch R_A die PRF F_k mit mehr als vernachlässigbarer Wahrscheinlichkeit unterscheiden, wodurch ein Widerspruch entsteht und Π_{PRF} CPA-Sicher sein muss

□

2.11 Pseudorandom Permutationen



Definition 2.43 (Pseudorandom Permutierer)

Sei $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ einer effizienter, längenerhaltender, schlüsselabhängiger Permutierern. F ist ein starker pseudoranom Permutierer, wenn für alle probabilistischen polynomialzeit Unterscheider D eine vernachlässigbare Funktion $negl(n)$ existiert, sodass gilt:

$$P[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n) = 1] - P[D^{f_k(\cdot),f_k^{-1}(\cdot)}(1^n) = 1] < negl(n)$$

Dabei wird die erste Wahrscheinlichkeit über die echt zufällige Wahl von $k \in \{0,1\}^n$ und das Zufallsprinzip von D berechnet und die zweite Wahrscheinlichkeit wird über die echt zufällige Wahl von $f \in Perm_n$ und das Zufallsprinzip von D berechnet.

Bemerkung 2.44.

- PRP sind bijektive PRF und damit ein Spezialfall von PRF der die invertierte Berechnung zulässt
- Wenn F ein PRP ist, dann ist F auch eine PRF
- Der Unterscheider hat in diesem Fall ein Orakel welches sowohl F_k als auch F_k^{-1} berechnet
- Ähnlich zu PRP sind Bit-Permutierer. Diese erhalten die 1er und 0er der Eingabe und vertauschen nur die jeweiligen Positionen. Bit-Permutierer sind keine starken PRP, da ein Unterscheider D die Anzahl der 1er und 0er speichert und für die Ein und Ausgabe für sein Orakel abgleicht. Damit kann er einen Bit-Permutierer erkennen

2.12 Operationsmodie für Blockchiffren

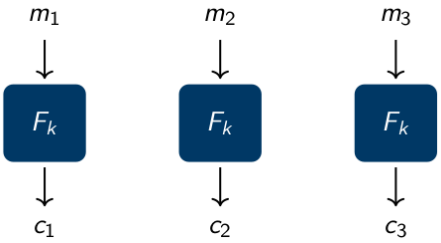
Definition 2.45 (Blockchiffren)

Bisher konnten die Kryptosystem Nachrichten beliebiger Länge bearbeiten. Bei **Blockchiffren** ist dies nicht der Fall. Hierbei werden Nachrichten in festen Blöcke aufgeteilt und diese jeweils verschlüsselt. Der letzte Block wird überlicherweise gepadded, d.h. aufgefüllt damit er die nötige Länge hat.

Bemerkung 2.46.

- Blockchiffren sind (je nach Wahl) sichere Instanzen eines starken pseudorandom Permutierers mit fester Schlüssel und Blocklänge
- Die folgenden Operationsmodie beschreiben unterschiedliche Umsetzungen von Blockchiffren
- Alle Modie die vorgestellt werden sind nicht CCA-Sicher!

2.12.1 Electronic Code Book (ECB) Modus



Satz 2.47 (ECB und CPA-Sicherheit)

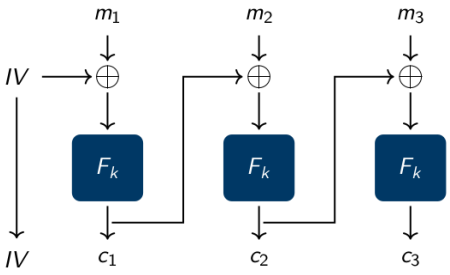
ECB ist nicht CPA-sicher, da ECB deterministisch ist.

Satz 2.48 (ECB und EAV-Sicherheit)

ECB ist nicht EAV-Mult-sicher.

Beweis 2.49.
Als Gegenspieler A im EAV-Mult Experiment kann die Nachricht $m_0 = 0^{2n}$ und die Nachricht $m_0 = 0^n \| 1^n$ dem Challenger geben. Ist der zurückkommende Crpytotext von der Form $c \| c$, dann gibt A 0 aus, ansonsten 1. Damit gewinnt A immer das Experiment und ECB ist nicht EAV-Mult-Sicher. □

2.12.2 Cipher Block Chaining (CBC) Modus



Satz 2.50 (CBC ist CPA-Sicherheit)

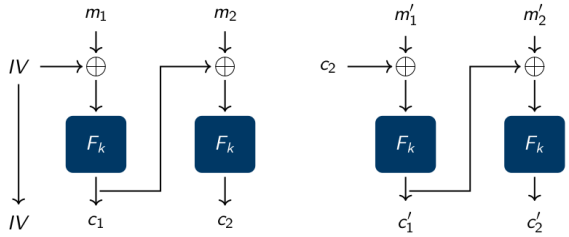
Wenn IV echt zufällig ist und F_k ein PRP ist, dann ist CBC auch CPA-sicher.

Beweis 2.51.
Ähnlich wie der CPA-Beweis für den CTR Modus. □

Bemerkung 2.52.

- IV ein ein echt zufällig ausgwählter Initial Vektor der Länge n
- CBC erfüllt die Eigenschaften der korrekten Ver und Entschlüsselung, daher muss F_k auch ein PRP sein

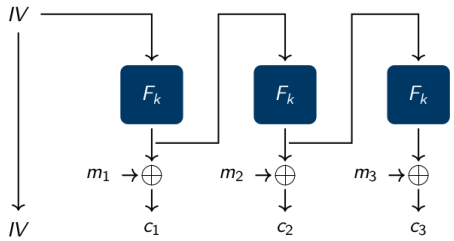
2.12.3 Chained Cipher Block Chaining (Chained CBC) Modus



Bemerkung 2.53.

- Der letzte Block des vorrangegangenen Chiffretextes ist der IV des nächsten Chiffretextes
- Dadurch kennt ein Angreifer einige IV und das Kryptosystem ist nicht mehr CPA-sicher
- Daher gilt: Auch kleine Veränderungen an einem Kryptosystem können es schon unsicher machen

2.12.4 Output Feedback (OFB) Modus



Satz 2.54 (OFB ist CPA-Sicherheit)

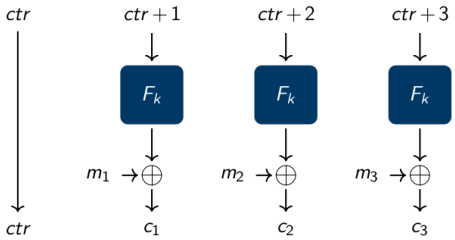
Wenn IV echt zufällig ist und F_k ein PRF ist, dann ist OFB auch CPA-sicher.

Beweis 2.55.
Ähnlich wie der CPA-Beweis für den CTR Modus. □

Bemerkung 2.56.

- Der Vorteil von OFB gegenüber CBC ist, dass hierbei Preprocessing angewendet werden kann (F_k kann zeitlich mit der vorhergehenden Verschlüsselung berechnet werden), wodurch OFB effizienter sein kann

2.12.5 Counter (CTR) Modus



Satz 2.57 (CTR ist CPA-Sicherheit)

Wenn F_k ein PRF ist, dann ist CTR auch CPA-sicher.

Beweis 2.58.

1. Statt CTR mit einer PRF Π definieren wir erstmal ein Kryptosystem Π' nach CTR Prinzip, das statt der PRF eine echt zufällige Funktion verwendet
2. Nun nehmen wir an, dass es einen Angreifer A' gegen Π' gibt, mit welchem das CPA-Experiment für Π' simuliert wird. A' schickt also dem Challenger zwei Nachrichten m_0, m_1 und bekommt die Antwort (IV, c). A' kann nun seinem Orakel $p(n)$-viele Anfragen schicken und bekommt $p(n)$-viele Antworten (IV_i, c_i)
3. Es treten zwei Fälle auf:
   (a) Es entsteht keine Antwort, sodass $IV_i = IV$ gilt. Damit hat A' nur die Möglichkeit zu raten und somit gilt $PrivK^{cpa}_{\mathcal{A}',\Pi}(n) = 1/2$

(b) Es entsteht mind. eine Antwort, sodass $IV_i + x = IV + y$ für beliebige $x, y > 0$ gilt (x, y entstehen durch den Counter bei CTR). Damit könnte A das Experiment mit Wslk. 1 richtig beantworten. Dieser Fall tritt aber nur vernachlässigbar oft auf, da es maximal $p(n)$-Vergleichswerte zu dem IV ausgewählt aus 2^n Möglichkeiten gibt. Daher ist die Wahrscheinlichkeit für diesen Fall vernachlässigbar $negl(n)$

4. Zusammengefasst gilt für A' folglich $PrivK^{cpa}_{\mathcal{A}',\Pi}(n) = 1/2 + negl(n)$, wodurch Π' als CPA-sicher gilt.
5.

TODO Übungsblatt 5 fertig □

Bemerkung 2.59.

- Im Gegensatz zu den anderen Varianten (bis auf ECB) ist dieser Modus gut parallelisierbar und daher effizienter

2.13 Sicherheit gegenüber Chosen Ciphertext Angriffen

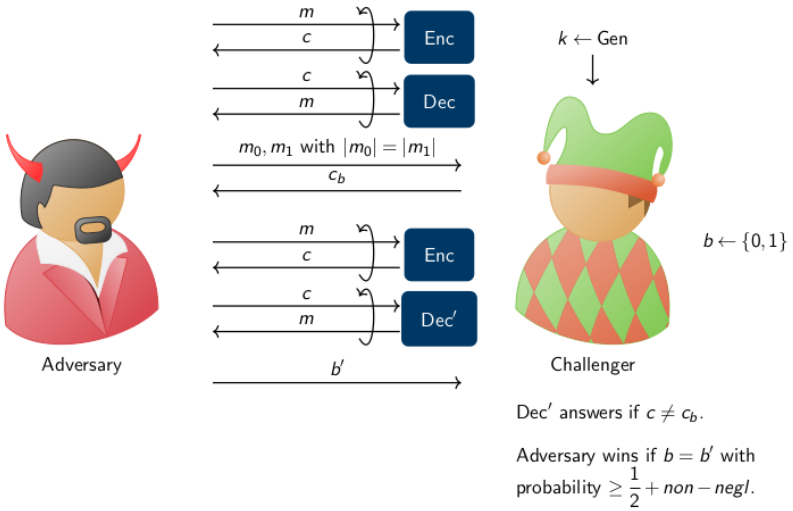
Definition 2.60 (CCA-Sicherheit)

Ein Kryptosystem Π hat ununterscheidare Verschlüsselung gegenüber einem Chosen Ciphertext Angriff, auch CCA-Sicherheit genannt, falls für jeden probabilistischen polynomialzeit Angreifer $\mathcal{A}$ eine vernachlässigbare Funktion $negl(n)$ existiert, sodass gilt:

$$P[PrivK^{cca}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

Experiment $PrivK^{cca}_{\mathcal{A},\Pi}(n)$

Dieses Experimant ist wieder mit einem Gegenspieler $\mathcal{A}$ und einem Challenger:



$PrivK^{cca}_{\mathcal{A},\Pi}(n)$
$k \leftarrow Gen(1^n)$
$(m_0, m_1) \leftarrow \mathcal{A}^{Enc_k(\cdot),Dec_k(\cdot)}(1^n), \lvert m_0\rvert = \lvert m_1\rvert$
$b \leftarrow \{0,1\}$
$c_b \leftarrow Enc_k(m_b)$
$b' \leftarrow \mathcal{A}^{Enc_k(\cdot),Dec_k(\cdot)}(c_b), c \neq c_b$
if $b' = b$ return 1 else return 0

Dabei bedeutet $c \neq c_b$, dass der Gegenspieler $\mathcal{A}$ alle Ciphertexte bis auf c_b seinem Entschlüsselungorakel mitgeben kann. Es wird $PrivK^{cca}_{\mathcal{A},\Pi}(n) = 1$ geschrieben falls de Ausgabe des Experimentes 1 ist, was impliziert, dass $\mathcal{A}$ erfolgreich war.

Bemerkung 2.61.

- **Abkürzung:** Diese Sicherheit wird auch als ING-CCA-Sicherheit (Indistinguishable chosen ciphertext attack secure) bzw. CCA-Sicherheit bezeichnet
- **Erinnerung:** Bei einem Chosen Ciphertext Angriff kann ein Angreifer beliebige Nachrichten (bis auf die abgehörte Nachricht) ver- und entschlüsseln lassen
- **Anmerkung Einfache und Mehrfache Verschlüsselug:** Ein Kryptosystem das CCA-Sicher für einfache Verschlüsselung mit gleichem Schlüssel ist, ist auch CCA-Sicher für mehrfache Verschlüsselung mit gleichem Schlüssel und andersrum, d.h. ING-CCA-Single $\Leftrightarrow$ ING-CCA-Mult!
- **Vergleich zu den vorherigen Sicherheitsdefinitionen:** Da das CPA Experiment im CCA Experiment enthalten ist, kann man per Reduktion zeigen, dass CCA-Sicherheit auch CPA-Sicherheit impliziert (Andersum gilt dies nicht). D.h. auch, dass CCA-Sicherheit stärker als die vorherigen Sicherheitdefinitionen ist
- **Unverkettbarkeit**: CCA-Sicherheit impliziert Unverkettbarkeit (non malleability) von Nachrichten, d.h. verändert der Angreifer im Experiment den Ciffretext auch nur minimal, so verrät die dazugehörige Entschlüsselung nur noch vernachlässigbare viel Informationen über den ürsprünglichen Klartext
- **Konstruktion:** CCA-sichere Verfahren können mit dem bisherigen Wissen nicht implementiert werden. Dafür braucht man z.b. MACs, welche später erläutert werden

Satz 2.62 (CCA-Sicherheit und bisherige Kryptosysteme)

Die bisher vorgestellten Kryptosysteme sind nicht CCA-sicher!

Beweis 2.63.

- Auch CCA-sichere Kryptosysteme müssen nicht deterministisch sein, daher sind alle deterministischen Verfahren nicht CCA-sicher
- Die restlichen bisherigen Verfahren bauen auf PRF und der XOR-Operation auf und funktionieren folgendermaßen:

$$Enc(k, m) = (r, s) = (r, F(k, r) \oplus m)$$

- Ein Gegenspieler A im CCA Experiment sähe folgendermaßen aus:
  1. Setze $m_0 = 0^n$ und $m_1 = 1^n$
  2. A bekommt (r, s) und erzeugt daraus (r, s') indem er das erste Bit von s flippt
  3. A sendet (r, s') an sein Entschlüsselungsorakel und erhält entweder $0\|1^{n-1}$ oder $1\|0^{n-1}$
  4. Er kann die Nachricht unterscheiden, insofern $n > 2$ ist und gewinnt das Experiment

□

Beispiel 2.64.

Beweis, dass CBC nicht CCA-sicher ist:

1. Der Gegenspieler A schickt die beiden Nachrichten $m_0 = 0^n$ und $m_1 = 1^n$
2. A bekommt $IV\|c_1$ und erzeug $(IV \oplus \partial)\|c_1$ mit einem beliebigen ∂
3. A schickt $(IV \oplus \partial)\|c_1$ an sein Entschlüsselungsorakel und erhält als Ausgabe $m' = m_b \oplus \partial$
4. A kann ∂ herausrechnen und die Nachrichten unterscheiden

Beweis, dass CTR nicht CCA-sicher ist:

1. Der Gegenspieler A schickt die beiden Nachrichten $m_0 = 0^n$ und $m_1 = 1^n$
2. A bekommt $c = m_b \oplus PRF(\ldots)$ und erzeug damit $c' = m_b \oplus PRF(\ldots) \oplus \partial$
3. A schickt c' an sein Entschlüsselungsorakel und erhält $m_b \oplus \partial$
4. A kann ∂ herausrechnen und die Nachrichten unterscheiden

2.13.1 Orakel Padding Angriffe*

Zugrundeliegendes System

- Bei den Block Chiffren muss der letzte Block gepadded werden, d.h. es werden Füllbytes angehängt, damit die Größe passt
- Sei L die Blocklänge und b die Anzahl an bytes die angehängt werden soll
- Beim Verschlüsseln (beispielsweise mit CBC-Modus) wird schließlich die Anzahl an bytes die gepadded werden hinzugepadded, sodass beim Entschlüsseln überprüft werden kann, ob das Padding verändert wurde oder nicht
- Wurde das Padding verändert, entsteht ein PaddingIncorrectError
- Somit ist der Wertebereich $b \in \{1, \ldots, L\}$, da $b = 0$ durch die Überprüfung beim Entschlüs-

seln nicht möglich ist

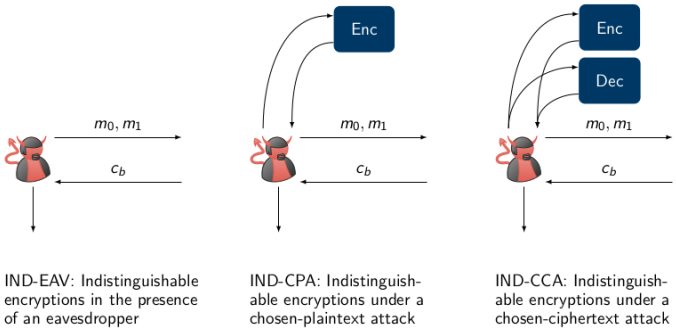
Erläuterung des Angriffes

- Zur Vereinfachung wird ein Beispiel mit dem Ciphertext (IV, c_1, c_2) betrachtet und mit den unbekannten Nachrichten m_1 und m_2
- Es gilt $m_2 = F_k^{-1}(c_2) \oplus c_1$ und $m_2 = m_2' || 0xb, ..., 0xb$
- Der Angriff:
  1. **Die Länge des Paddings herausfinden:** Man modifiziert das letzte Byte von c_1. Falls man einen PaddingIncorrectError bekommt, modifiziert man das vorletzte Byte usw. Das wiederholt man solange, bis man keinen Error bekommt und die Anzahl der benötigten Schritte gibt die Länge des Paddings an (Man nimmt c_1 da Veränderung hierbei auch die gleichen Bytes von c_2 beeinträchtigen und man somit auch den Spezialfall $b = L$ endeckt)
  2. **Entschlüsselung einzelner Bytes:**
     - m_2 endet mit $0xB||0xb...||0xb$, wobei $0xB$ das ite Byte von m_2 ist
     - Dann erstellt man eine Zeichenkette, mit der man das Padding und ite Bit von c_1 verändert: $\delta_i = 0x00||...||0xi||0x(b+1)...||0x(b+1) \oplus 0x00||...||0x00||0xb..||0xb$ (Mit dem hinteren Teil negiert man das vorherige Padding)
     - Man verschickt $(IV, c_1 \oplus \delta_i, c_2)$
     - Man erhält solange einen Error, bis $0x(B \oplus i) = 0x(b+1)$ gilt, woraus man auf das Byte des Klartextes schließen kann
     - Auf Ähnliche Weise können auch die weiteren Bytes entschlüsselt werden

Bemerkung 2.65.

- Dieser Angriff hat realitätsbezug und soll zeigen, dass CCA-Sicherheit nicht nur ein theoretisches Konstrukt darstellt
- Die Rückgabe des Errors stellt ein vereinfachtes Entschlüsselungsorakel dar, was ausreichte um die Nachricht zu entschlüsseln

2.14 Zusammenfassung der einzelnen Sicherheitdefinitionen



EAV-Single $\subset$ EAV-Mult $\subset$ CPA-Single = CPA-Mult = CPA $\subset$ CCA-Single = CCA-Mult = CCA