# CPA-Sicherheit unserer Konstruktion

## **Satz** CPA-Sicherheit von $\Pi_{cpa}$

Sei $\Pi_f$ eine Trapdoor-Einwegpermutation mit Hardcore-Prädikat $hc$. Dann ist $\Pi_{cpa}$ CPA-sicher.

**Beweis:**

- Sei $\mathcal{A}$ ein Angreifer mit Erfolgsws $\epsilon(n) = \mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi_f}(n) = 1]$.
- OBdA $(m_0, m_1) \leftarrow \mathcal{A}(pk)$ mit $\{m_0, m_1\} = \{0,1\}$. (Warum?)
- Verwenden $\mathcal{A}$, um $\mathcal{A}'$ im Spiel $CompHC_{\mathcal{A}',\Pi_f}(n)$ zu konstruieren.

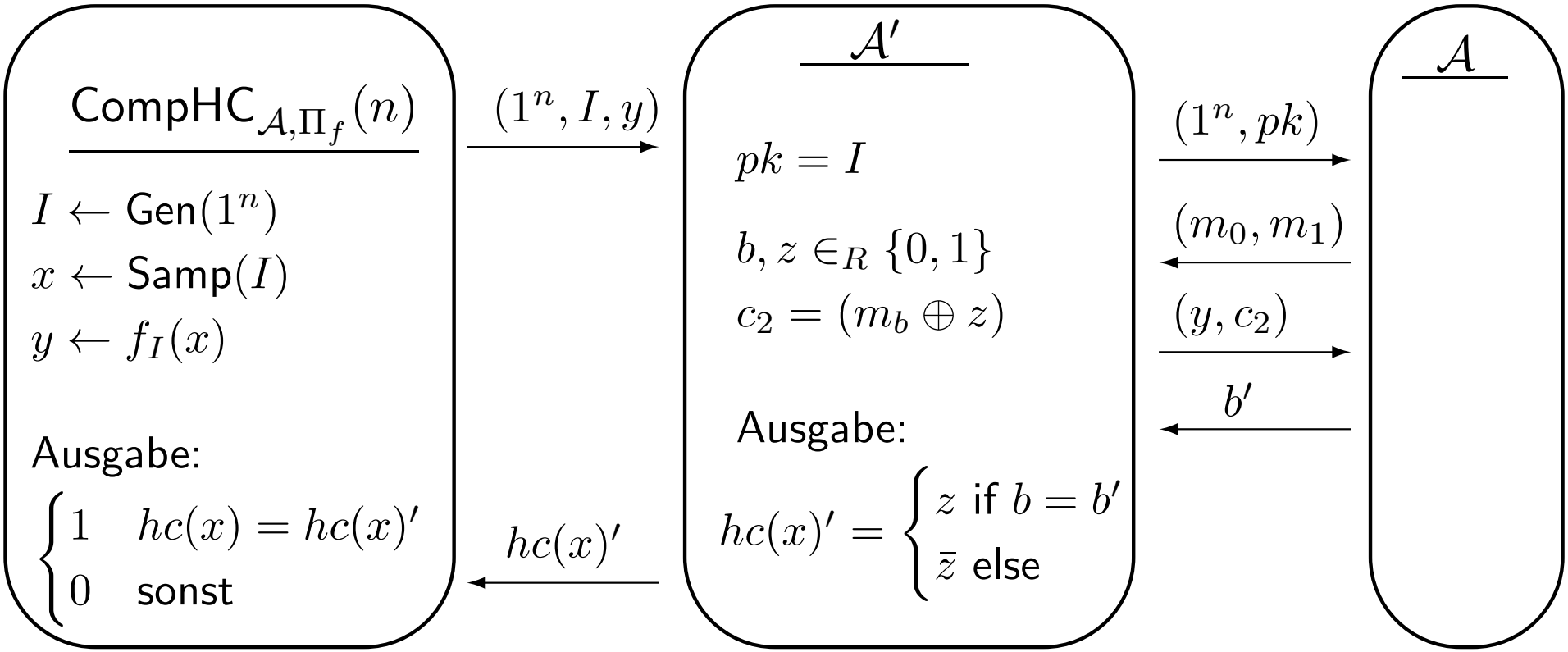
## **Algorithmus** Angreifer $\mathcal{A}'$

**Eingabe:** $1^n, I, y = f(x) \in D$

1. Setze $pk \leftarrow I$ und berechne $(m_0, m_1) \leftarrow \mathcal{A}(pk)$.
2. Wähle $b, z \in_R \{0,1\}$. Setze $c_2 \leftarrow m_b \oplus z$.
3. $b' \leftarrow \mathcal{A}(y, c_2)$

**Ausgabe:** $hc(x)' = \begin{cases} z & \text{falls } b = b' \\ \bar{z} & \text{sonst} \end{cases}$.

# Angreifer $\mathcal{A}'$ für das Hardcore-Prädikat



# CPA-Sicherheit von $\Pi_{cpa}$

**Beweis:** Fortsetzung

- Sei $x = f^{-1}(y)$. $\mathcal{A}_{hc}$ rät $z = hc(x)$.
- Es gilt $\mathrm{Ws}[\mathcal{A}_{hc}(f(x)) = hc(x)] =$
$$\frac{1}{2} \cdot \mathrm{Ws}[b = b' \mid z = hc(x)] + \frac{1}{2} \cdot \mathrm{Ws}[b \neq b' \mid z \neq hc(x)].$$
- **1. Fall** $z = hc(x)$: $(y, c_2)$ ist korrekte Verschlüsselung von $m_b$, d.h.
$$\mathrm{Ws}[b = b' \mid z = hc(x)] = \epsilon(n).$$
- **2. Fall** $z \neq hc(x)$: Es gilt
$$\begin{aligned}(y, c_2) &= (f(x), z \oplus m_b) \\ &= (f(x), z \oplus 1 \oplus m_b \oplus 1) = (f(x), hc(x) \oplus m_b \oplus 1).\end{aligned}$$
D.h. $(y, c_2)$ ist korrekte Verschlüsselung von $m_b \oplus 1 = m_{1-b}$.
$$\mathrm{Ws}[b \neq b' \mid z \neq hc(x)] = \mathrm{Ws}[1 - b = b' \mid z \oplus 1 = hc(x)] = \epsilon(n).$$

Da $hc$ ein Hardcore-Prädikat ist, folgt

$$\frac{1}{2} + \mathrm{negl}(n) \geq \mathrm{Ws}[A_{hc}(f(x)) = hc(x)] = \epsilon(n).$$

# Random Oracle

## **Definition** Random Oracle

Sei $\mathcal{F}^{n,\ell}$ die Menge aller Funktionen $\{0,1\}^n \to \{0,1\}^{\ell(n)}$. Ein *Random Oracle* ist eine zufällige Funktion $H \in_R \mathcal{F}^{n,\ell}$. Wir besitzen keine Beschreibung von $H$. Bei Anfrage $x$ liefert das Random Oracle $H(x)$.

**Anmerkung:** Bildliche Darstellung

- Ein Random Oracle $H$ ist eine Funktion in einer schwarzen Box.
- $H$ ist beobachtbar über das Eingabe/Ausgabe-Verhalten der Box.

**Alternative Beschreibung** eines Random Oracles

- Oracle erhält Anfragen $x_1, \ldots, x_q$.
- Falls $x_i \neq x_j$ für alle $j < i$, gib $y_i \in_R \{0,1\}^{\ell(n)}$ aus.
- Falls $x_i = x_j$ für ein $j < i$, gib $y_j$ aus.
- D.h. wir können uns vorstellen, dass das Orakel die Antworten auf Anfragen bei Bedarf erzeugt und konsistent beantwortet.
- Damit können Random Oracle durch ppt Alg. simuliert werden.

# Random Oracles liefern Einwegfunktionen

**Satz**

Für polynomielles $\ell(n)$ sind Random Oracles $H$ Einwegfunktionen.

**Beweis:**

- Sei $x \in_R \{0,1\}^n$ und $y = H(x)$. Wollen ein Urbild von $y$ ermitteln.
- Jeder Angreifer $\mathcal{A}$ stellt oBdA verschiedene Anfragen $x_1, \ldots, x_q$. (Warum sollte jeder Angreifer so verfahren?)
- $\mathcal{A}$ gewinnt offenbar falls $x_i = x$ für ein $i \in [q]$. Es gilt
$$\text{Ws}[x_i = x \text{ für ein } i] = \sum_{i=1}^{q} \text{Ws}[x_i = x] = \frac{q}{2^n}.$$
- $\mathcal{A}$ gewinnt ebenfalls für $H(x_i) = y$ für mindestens ein $i \in [q]$, d.h.
$$\text{Ws}[H(x_i) = y \text{ für mindestens ein } i] \leq \sum_{i=1}^{q} \text{Ws}[H(x_i) = y] = \frac{q}{2^{\ell(n)}}.$$
- Damit gilt $Ws[Invert_{\mathcal{A},H}(n) = 1] \leq \frac{q}{2^n} + \frac{q}{2^{\ell(n)}}$.
- Für polynomielles $q, \ell(n)$ ist dies vernachlässigbar in $n$.

# Random Oracle und kollisionsresistentes Hashen

**Satz**

Für polynomielles $\ell(n)$ sind Random Oracles kollisionsresistent.

**Beweis:**

- Jeder Angreifer $\mathcal{A}$ stellt oBdA verschiedene Anfragen $x_1, \ldots, x_q$.
- $\mathcal{A}$ gewinnt mit

$$\begin{aligned} \mathrm{Ws}[H(x_i) = H(x_j) \text{ für ein } i \neq j] &\leq \sum_{i \neq j} \mathrm{Ws}[H(x_i) = H(x_j)] \\ &= \frac{\binom{q}{2}}{2^{\ell(n)}} \leq \frac{q^2}{2^{\ell(n)}}. \end{aligned}$$

- Dies ist vernachlässigbar für polynomielles $q, \ell(n)$.

# Random Oracle Methode

## **Definition** Random Oracle Modell / Methode

Das *Random Oracle Modell (ROM)* nimmt die Existenz von Random Oracles an. Die *Random Oracle Methode* besteht aus 2 Schritten:

1. Konstruiere ein Verfahren $\Pi$ mit Hilfe eines Random Oracles $H$ und beweise die Sicherheit von $\Pi$ im ROM.
2. Instantiiere $\Pi$ mit einer kryptographischen Hashfunktion $H'$ anstelle von $H$, z.B. mit SHA-1.

**Negativ:**

- Beschreibung von $H'$ spezifiziert $H'(x)$ für alle $x$.
- Es existieren künstliche Kryptosysteme, die sicher im Random Oracle Modell aber unsicher für *jede* Instantiierung von $H'$ sind.

**Positiv:**

- Ein Beweis im ROM ist besser als kein Beweis.
- Erfolgreicher Angriff muss die Instantiierung von $H'$ attackieren.
- $H'$ kann leicht durch eine andere Hashfunktion ersetzt werden.

# CPA-sicheres effizientes RSA im ROM

## **Verschlüsselung** ROM-RSA

Sei $H : \mathbb{Z}_N^* \to \{0,1\}^{\ell(n)}$ ein Random Oracle.

1. **Gen:** $(N, e, d) \leftarrow GenRSA(1^n)$ mit $pk = (N, e), sk = (N, d)$.
2. **Enc:** Für $m \in \{0,1\}^{\ell(n)}$, wähle $r \in_R \mathbb{Z}_N^*$. Berechne
$$c \leftarrow (r^e \bmod N, H(r) \oplus m).$$
3. **Dec:** Für $c = (c_1, c_2)$ berechne
$$r := c_1^d \bmod N \text{ und } m := H(r) \oplus c_2.$$

# Sicherheit von RSA im Random Oracle Modell

**Satz** CPA-Sicherheit von ROM-RSA

Unter der RSA-Annahme und für ein Random Oracle $H$ ist ROM-RSA $\Pi$ CPA-sicher.

**Beweis:**

- Sei $\mathcal{A}$ ein Angreifer mit Erfolgws $\epsilon(n) = \mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1]$.
- $\mathcal{A}$ darf Orakelanfragen an $H$ stellen, sowohl vor Ausgabe von $(m_0, m_1)$ als auch nach Erhalt von $Enc(m_b)$.
- Definiere $Query$ : Ereignis $\mathcal{A}$ stellt Anfrage $r = c_1^d \bmod N$ an $H$.

$$\begin{aligned}
\epsilon(n) &= \mathrm{Ws}[\overline{Query}] \cdot \mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1 \mid \overline{Query}] + \\
&\quad \mathrm{Ws}[Query] \cdot \mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1 \mid Query] \\
&\leq \mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1 \mid \overline{Query}] + \mathrm{Ws}[Query].
\end{aligned}$$

- Zeigen
  $\mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1 \mid \overline{Query}] = \frac{1}{2}$ und $\mathrm{Ws}[Query] \leq \mathrm{negl}(n)$.
- Daraus folgt $\epsilon(n) \leq \frac{1}{2} + \mathrm{negl}(n)$.

# Beweis der CPA-Sicherheit von ROM-RSA (1/3)

**Beweis:**

- Falls $r$ nicht an $H$ angefragt wird, ist $H(r) \oplus m$ nach Eigenschaft des Random Oracles ein perfektes One-Time Pad für $m$.
- Daraus folgt $\mathrm{Ws}[PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1 \mid \overline{Query}] = \frac{1}{2}$.

# Beweis der CPA-Sicherheit von ROM-RSA (2/3)

**Beweis:** $\mathrm{Ws}[Query] \leq \mathrm{negl}(n)$

- Idee: Verwende Anfragen von $\mathcal{A}$, um $e$-te Wurzeln zu berechnen.

## Algorithmus RSA-Invertierer $\mathcal{A}'$

EINGABE: $N, e, c_1 = r^e \bmod N$

1. Wähle $k \in_R \{0,1\}^{\ell(n)}$. (Wir setzen $H(r) = k$, ohne $r$ zu kennen.)
2. $(m_0, m_1) \leftarrow \mathcal{A}(N, e)$, beantworte Orakelanfragen $r_i$ an $H(\cdot)$

$$\text{konsistent mit } \begin{cases} k_i = k & \text{für } r_i^e = c_1 \bmod N \\ k_i \in_R \{0,1\}^{\ell(n)} & \text{sonst} \end{cases}.$$

3. Berechne $c \leftarrow (c_1, k \oplus m_b)$ für ein $b \in_R \{0,1\}$.
4. $b' \leftarrow \mathcal{A}(c)$, beantworte Anfragen von $\mathcal{A}$ an $H(\cdot)$ wie zuvor.
5. Falls $r_i^e = c_1 \bmod N$ für eine der Orakelanfragen, setze $r \leftarrow r_i$.

AUSGABE: $r$

- Es gilt $Ws[Query] = Ws[\mathcal{A}'(N, e, r^e) = r] \leq \mathrm{negl}(n)$.

# Beweis der CPA-Sicherheit von ROM-RSA (3/3)