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 $\mathcal{A}^H$ stellt oBdA verschiedene Anfragen $x_1, \ldots, x_q$. (Warum sollte jeder Angreifer so verfahren?)
- $\mathcal{A}^H$ gewinnt offenbar falls $x_i = x$ für ein $i \in [q]$. Es gilt
$$\mathrm{Ws}[x_i = x \text{ für ein } i] = \sum_{i=1}^{q} \mathrm{Ws}[x_i = x] = \frac{q}{2^n}.$$
- $\mathcal{A}^H$ gewinnt ebenfalls für $H(x_i) = y$ für mindestens ein $i \in [q]$, d.h.
$$\mathrm{Ws}[H(x_i) = y \text{ für mindestens ein } i] \leq \sum_{i=1}^{q} \mathrm{Ws}[H(x_i) = y] = \frac{q}{2^{\ell(n)}}.$$
- Damit gilt $Ws[Invert_{\mathcal{A}^H,H}(n) = 1] \leq \frac{q}{2^n} + \frac{q}{2^{\ell(n)}}$.
- 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 $\mathcal{A}^H$ stellt oBdA verschiedene Anfragen $x_1, \ldots, x_q$.
- $\mathcal{A}^H$ gewinnt mit

$$\begin{aligned} \mathrm{Ws}[H(x_i) = H(x_j) \text{ für ein } i \neq j] &\leq \sum_{i \neq j} \mathrm{Ws}[H(x_i) = H(x_j)] \\ &= \frac{\binom{q}{2}}{2^{\ell(n)}} \leq \frac{q^2}{2^{\ell(n)+1}}. \end{aligned}$$

- 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 im Beweis an. Die *Random Oracle Methode* besteht aus 2 Schritten:

1. Konstruiere ein Verfahren Π mit einer Hash Funktion H und beweise die Sicherheit von Π^H im ROM, d.h. wenn H als Random Oracle modelliert wird.
2. Instantiiere Π 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.

CPA-sicheres effizientes RSA im ROM

Verschlüsselung Π =ROM-RSA

1. **Gen:** $(N, e, d) \leftarrow \mathit{GenRSA}(1^n)$ mit $pk = (N, e, H)$, $sk = (N, d)$ mit Hash Funktion $H : \mathbb{Z}_N^* \to \{0,1\}^{\ell(n)}$.
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 Π^H CPA-sicher.

Beweis:

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

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

- Zeigen
  $\mathrm{Ws}[PubK^{cpa}_{\mathcal{A}^H,\Pi^H}(n) = 1 \mid \overline{Query}] = \frac{1}{2}$ 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: $\mathrm{Ws}[PubK^{cpa}_{\mathcal{A}^H,\Pi^H}(n) = 1 \mid \overline{Query}] = \frac{1}{2}$

- 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 $\mathrm{Ws}[PubK^{cpa}_{\mathcal{A}^H,\Pi^H}(n) = 1 \mid \overline{Query}] = \frac{1}{2}$.

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

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

- Idee: Verwende Anfragen von $\mathcal{A}^H$, 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. $(m_0, m_1) \leftarrow \mathcal{A}^H(N, e)$, beantworte Orakelanfragen r_i an $H(\cdot)$

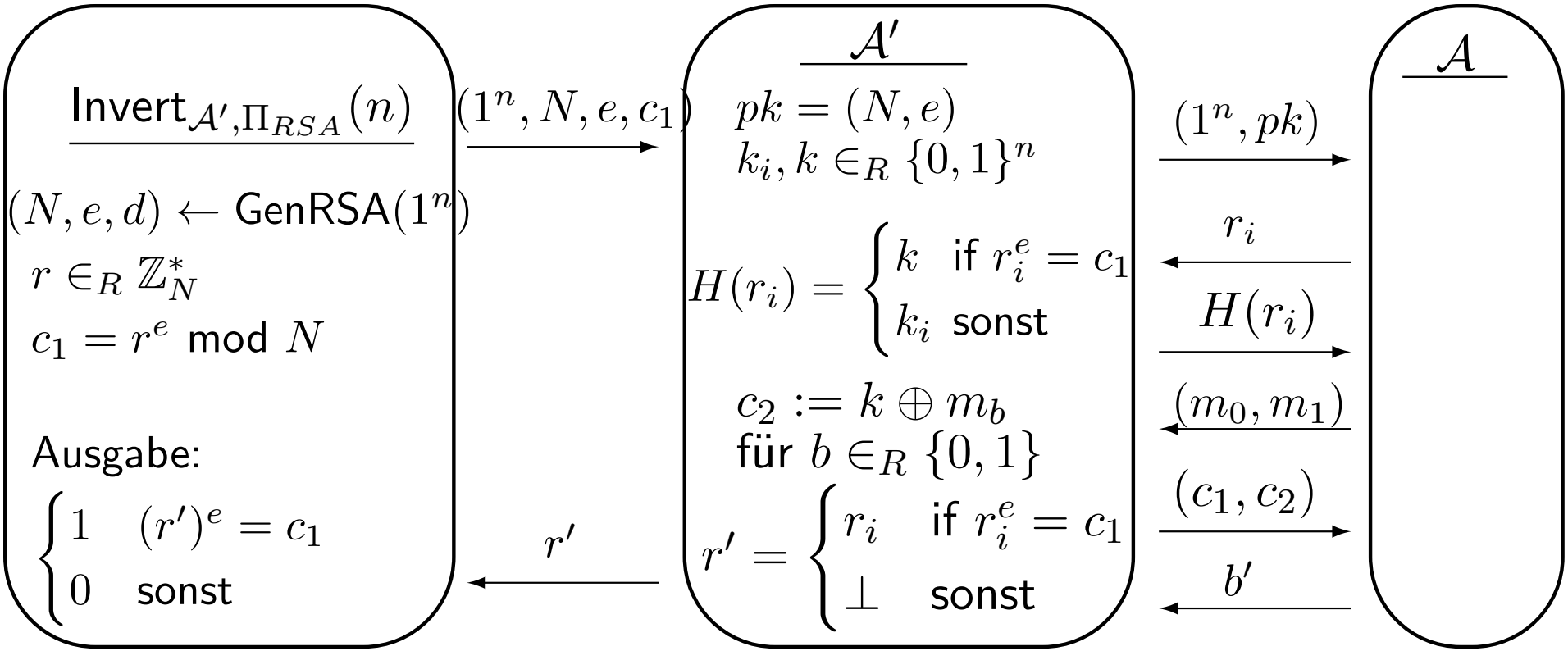
$$\text{konsistent mit } \begin{cases} k_i = k & \text{für } r_i^e = c_1 \bmod N \\ k_i \in_R \{0,1\}^{\ell(n)} & \text{sonst} \end{cases}.$$

3. Berechne $c \leftarrow (c_1, k \oplus m_b)$ für ein $b \in_R \{0,1\}$.
4. $b' \leftarrow \mathcal{A}^H(c)$, beantworte Anfragen von $\mathcal{A}^H$ 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 $\mathit{Ws}[\mathit{Query}] = \mathit{Ws}[\mathcal{A}'(N, e, r^e) = r] \leq \mathrm{negl}(n)$.

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



Sicherheit gegenüber CCA

Idee:

- Ersetze One-Time Pad durch CCA-sicheres Secret Key Verfahren.
- *Erinnerung Krypto I:* Konstruktion von CPA-sicherem Secret Key Verfahren mittels Pseudozufallsfunktion (und MAC) möglich.

Verschlüsselung Π = ROM-RSA-2

Sei $\Pi' = (Gen', Enc', Dec')$ ein CPA-sicheres Secret Key Verschlüsselungsverfahren.

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

Sicherheit von ROM-RSA-2

Satz Sicherheit von ROM-RSA-2

Unter der RSA-Annahme, für ein Random Oracle H und ein CPA-sicheres Π' liefert ROM-RSA-2 CCA-sichere Verschlüsselung.

Beweis:

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

$$\epsilon(n) \quad = \quad \mathrm{Ws}[PubK^{cca}_{\mathcal{A}^H,\Pi^H}(n) = 1 \mid \overline{Query}] + \mathrm{Ws}[Query].$$

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

Beweis der CCA-Sicherheit von ROM-RSA

Beweis:

- Falls r nicht an H angefragt wird, lernt Angreifer $\mathcal{A}^H$ nur durch Anfragen $Dec_{sk}(c_1, \cdot)$ Informationen über $k = H(r)$
- Sonst ist $k = H(r)$ uniform.
- Ohne Beweis: es existiert ein Angreifer $\mathcal{B}$ mit: $\mathrm{Ws}[PubK^{cca}_{\mathcal{A}^H,\Pi^H}(n) = 1 \mid \overline{Query}] \leq \mathrm{Ws}[PrivK^{cca}_{\mathcal{B},\Pi'} = 1] = \frac{1}{2} + \mathrm{negl}(n)$.

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

- Problem: Müssen Orakel $Dec_{sk}(\cdot)$ simulieren, ohne sk zu kennen.
- Verwende dazu geschicktes Simulieren des Random Oracles $H(\cdot)$ und des Dec Oracles $Dec_{sk}(\cdot)$.

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. $(m_0, m_1) \leftarrow \mathcal{A}^H(N, e)$ beantworte Hash Anfragen r_i an $H(\cdot)$

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

3. Beantworte Anfragen $\hat{c}_j = (\hat{c}_{j,1}, \hat{c}_{j,2})$ an $Dec_{sk}(\cdot) =$

$$\text{konsistent mit } \begin{cases} Dec'_k(\hat{c}_{j,2}) & \text{für } \hat{c}_{j,1} = c_1 \\ Dec'_{k_i}(\hat{c}_{j,2}) & \exists i : \hat{c}_{j,1}^e = r_i \\ Dec'_{\hat{k}_j}(\hat{c}_{j,2}) \text{ für } \hat{k}_j \in_R \{0,1\}^{\ell(n)} & \text{sonst} \end{cases} .$$

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

AUSGABE: r

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

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