

# Einwegfunktionen

**Ziel:** CPA-sichere Verschlüsselung aus Trapdoor-Einwegpermutation

**Später:** CCA-sichere Verschlüsselung aus Trapdoor-Einwegperm.

## Spiel Invertieren $Invert_{\mathcal{A},f}(n)$

Sei  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  effizient berechenbar,  $\mathcal{A}$  ein Invertierer für  $f$ .

1 Wähle  $x \in_R \{0, 1\}^n$ . Berechne  $y \leftarrow f(x)$ .

2  $x' \leftarrow \mathcal{A}(1^n, y)$

3  $Invert_{\mathcal{A},f}(n) = \begin{cases} 1 & \text{falls } f(x') = y \\ 0 & \text{sonst} \end{cases}$ .

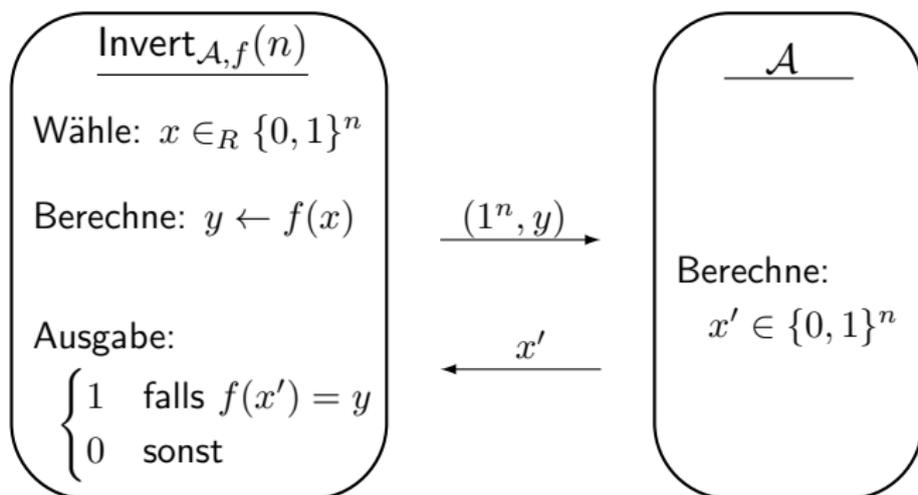
## Definition Einwegfunktion

Eine Funktion  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  heißt *Einwegfunktion*, falls

1 Es existiert ein deterministischer pt Alg  $\mathcal{B}$  mit  $f(x) \leftarrow \mathcal{B}(x)$ .

2 Für alle ppt Algorithmen  $\mathcal{A}$  gilt  $\text{Ws}[Invert_{\mathcal{A},f}(n) = 1] \leq \text{negl}(n)$ .

# Spiel Invertieren



# Existenz von Einwegfunktionen

- **Problem:** Existenz von Einwegfunktionen ist ein offenes Problem.

## Satz (Levin)

Wenn Einwegfunktionen existieren, dann kann man auch explizit eine angeben.

Beweis (Idee). Konstruktion von  $f_{Levin}$  mit Eingabe  $x$ .

- 1 Interpretiere  $x = M||z$ , mit  $|M| = \log |z|$ .
- 2 Führe  $M$  auf Eingabe  $z$  aus.
- 3 Definiere  $f_{Levin(x)} = \begin{cases} M(z): & M \text{ terminiert nach } |z|^3 \text{ Schritten} \\ \perp & : \text{ sonst} \end{cases}$

Mann kann zeigen:

- Falls Einwegfunktionen existieren, dann gilt für alle ppt  $\mathcal{A}$ :  
 $Ws[Invert_{\mathcal{A}, f_{Levin}}(n) = 1] \leq 1 - 1/q(n)$ , für ein Polynom  $q$ .
- Durch "Hardness amplification" kann aus  $f_{Levin}$  eine Einwegfunktion  $f_{Levin'}$  konstruiert werden.

# Die Faktorisierungsannahme

- Konstruktion einer Einwegfunktion unter Komplexitätsannahme (z.B. Faktorisierung)
- Verwenden dazu  $(N, p, q) \leftarrow \text{GenModulus}(1^n)$  von RSA.

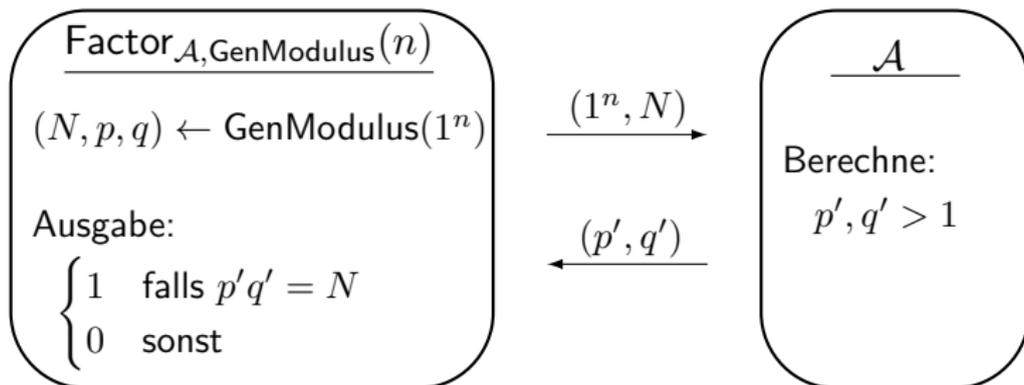
## Spiel Faktorisierungsspiel $\text{Factor}_{\mathcal{A}, \text{GenModulus}}(n)$

1  $(N, p, q) \leftarrow \text{GenModulus}(1^n)$

2  $(p', q') \leftarrow \mathcal{A}(N)$  mit  $p', q' > 1$ .

3 
$$\text{Factor}_{\mathcal{A}, \text{GenModulus}}(n) = \begin{cases} 1 & \text{falls } p'q' = N \\ 0 & \text{sonst} \end{cases} .$$

# Spiel Faktorisieren



## Definition Faktorisierungsannahme

Faktorisieren ist hart bezüglich *GenModulus* falls für alle ppt Algorithmen  $\mathcal{A}$  gilt  $\text{Ws}[\text{Factor}_{\mathcal{A}, \text{GenModulus}}(n) = 1] \leq \text{negl}(n)$ .

*Faktorisierungsannahme:* Faktorisieren ist hart bezüglich *GenModulus*.

# Konstruktion aus Faktorisierungsannahme

- Sei  $p(n)$  ein Polynom, so dass  $GenModulus(1^n)$  höchstens  $p(n)$  Zufallsbits verwendet.
- OBdA sei  $p(n) : \mathbb{N} \rightarrow \mathbb{N}$  monoton wachsend.

## Algorithmus FACTOR-ONEWAY $f_{FO}$

**Eingabe:**  $x \in \{0, 1\}^*$

- 1 Berechne  $n$  mit  $p(n) \leq |x| < p(n+1)$ .
- 2  $(N, p, q) := GenModulus(1^n, x)$ , wobei  $GenModulus$  die Eingabe  $x$  als internen Zufallsstring verwendet.

**Ausgabe:**  $N$

**Bemerkung:**

- $GenModulus(1^n, x)$  ist deterministisch. (Derandomisierung)

# Existenz von Einwegfunktionen

## Satz Einweg-Eigenschaft von $f_{FO}$

Unter der Faktorisierungsannahme ist  $f_{FO}$  eine Einwegfunktion.

### Beweis:

- Sei  $\mathcal{A}$  ein Invertierer für  $f_{FO}$  mit Erfolgsws  $\epsilon(n)$ .
- Konstruieren mit  $\mathcal{A}$  Faktorisierer  $\mathcal{A}'$  im Spiel  $Factor_{\mathcal{A}', GenModulus}(n)$ .

## Algorithmus Faktorisierer $\mathcal{A}'$

EINGABE:  $1^n, N$

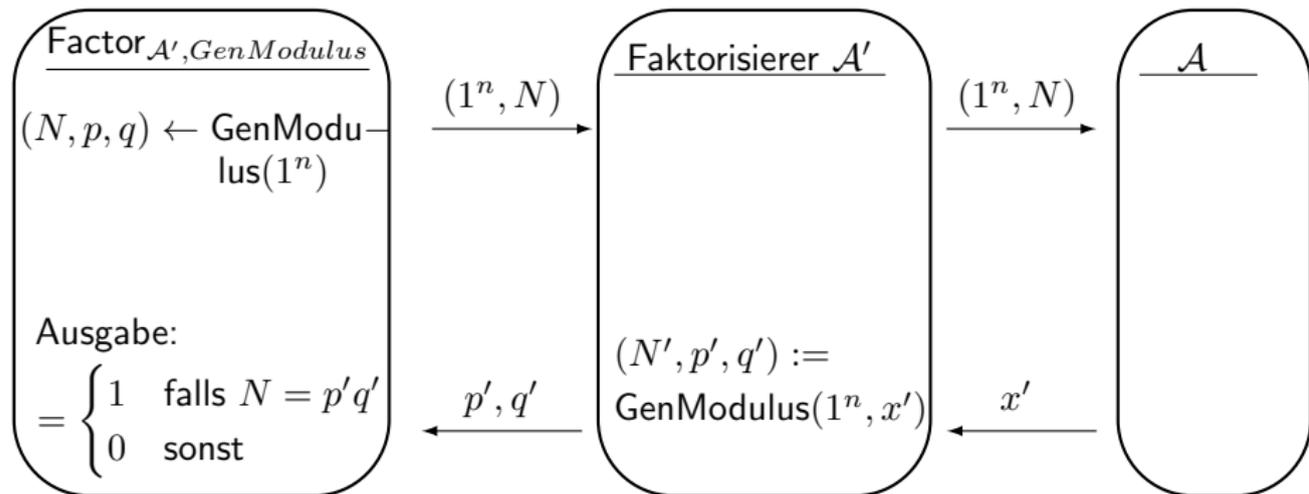
- 1  $x' \leftarrow \mathcal{A}(1^n, N)$ .
- 2  $(N', p', q') \leftarrow GenModulus(1^n, x')$ .

AUSGABE:  $p', q'$

Unter der Faktorisierungsannahme gilt

$$\text{negl}(n) \geq \text{Ws}[Factor_{\mathcal{A}', GenModulus}(n) = 1] = \text{Ws}[Invert_{\mathcal{A}, f_{FO}}(n) = 1] = \epsilon(n).$$

# Faktorisieren mit Invertierer für $f_{FO}$



# Trapdoor-Funktionsfamilie

## Definition Funktionsfamilie

Eine *Funktionsfamilie*  $\Pi_f = (Gen, Samp, f)$  besteht aus 3 ppt Alg:

- 1  $I \leftarrow Gen(1^n)$ , wobei  $I$  eine Urbildmenge  $D$  für  $f$  definiert.
- 2  $x \leftarrow Samp(I)$ , wobei  $x \in_R D$ .
- 3  $y := f_I(x)$  und  $f_I : D \rightarrow D$  ist bijektiv.

## Definition Trapdoor-Funktionsfamilie

*Trapdoor-Funktionsfamilie*  $\Pi_f = (Gen, Samp, f, Inv)$  besteht aus

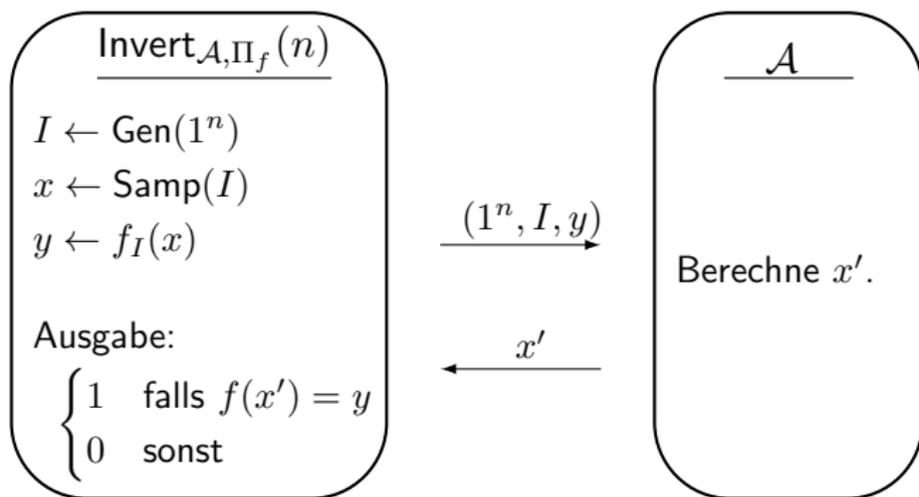
- 1  $(I, td) \leftarrow Gen(1^n)$  mit  $td$  als Trapdoor-Information.
- 2  $x \leftarrow Samp(I)$  wie zuvor.
- 3  $y := f_I(x)$  wie zuvor.
- 4  $x \leftarrow Inv_{td}(y)$  mit  $Inv_{td}(f_I(x)) = x$  für alle  $x \in D$ .

$\Pi_f$  ist eine (*Trapdoor-*)*Permutationsfamilie*, falls  $R = D$  und  $f_I : D \rightarrow D$  bijektiv ist.

## Spiel Invertieren einer Funktion $\text{Invert}_{\mathcal{A}, \Pi_f}(n)$

Sei  $\mathcal{A}$  ein Invertierer für die Familie  $\Pi_f$ .

- 1  $I \leftarrow \text{Gen}(1^n)$ ,  $x \leftarrow \text{Samp}(I)$  und  $y := f_I(x)$ .
- 2  $x' \leftarrow \mathcal{A}(I, y)$ .
- 3  $\text{Invert}_{\mathcal{A}, \Pi_f}(n) = \begin{cases} 1 & \text{falls } f_I(x') = y \\ 0 & \text{sonst} \end{cases}$ .



# Konstruktion einer Trapdoor-Einwegfunktion

## Definition Einweg-Funktion

Eine (Trapdoor-)Funktionsfamilie heißt *(Td-)Einwegfunktion* falls für alle ppt Algorithmen  $\mathcal{A}$  gilt  $\mathbb{W}_S[\text{Invert}_{\mathcal{A}, \Pi_f}(n) = 1] \leq \text{negl}(n)$ .

## Bsp: Trapdoor-Einwegpermutation unter RSA-Annahme

- **Gen( $1^n$ ):**  
 $(N, e, d) \leftarrow \text{GenRSA}(1^n)$ , Ausgabe  $I = (N, e)$  und  $td = (N, d)$ .
- **Samp( $I$ ):**  
Wähle  $x \in_R \mathbb{Z}_N$ .
- **$f_I(x)$ :**  
Berechne  $y := x^e \bmod N$ .
- **$\text{Inv}_{td}(y)$ :**  
Berechne  $x := y^d \bmod N$ .

# Hardcore-Prädikat

**Ziel:** Destilliere Komplexität des Invertierens auf ein Bit.

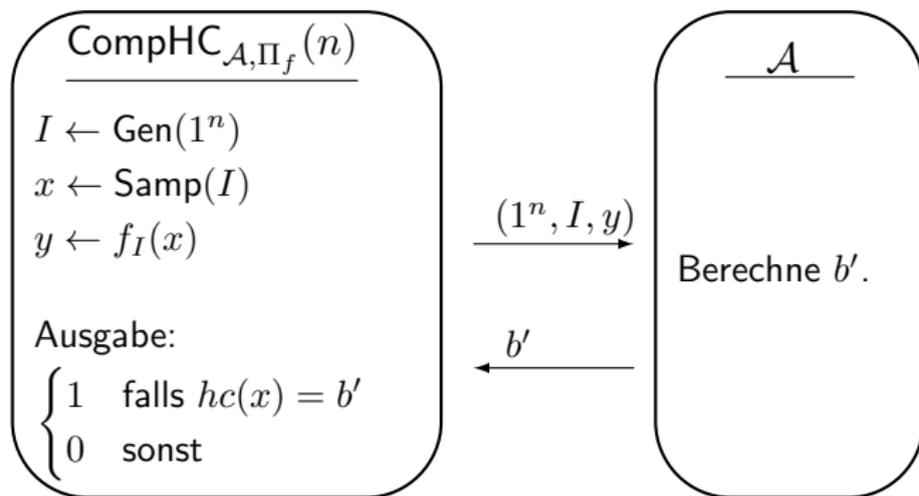
## Definition Hardcore-Prädikat

Sei  $\Pi_f$  eine Einwegfunktion. Sei  $hc$  ein deterministischer pt Alg mit Ausgabe eines Bits  $hc(x)$  bei Eingabe  $x \in D$ .  $hc$  heißt *Hardcore-Prädikat* für  $f$  falls für alle ppt Algorithmen  $\mathcal{A}$  gilt:

$$\text{Ws}[\mathcal{A}(1^n, I, f(x)) = hc(x)] \leq \frac{1}{2} + \text{negl}(n).$$

**Intuition:** Bild  $f(x)$  hilft nicht beim Berechnen von  $hc(x)$ .

# Spiel zum Berechnen des Hardcore-Prädikats



Falls  $hc$  ein Hardcoreprädikat ist, so gilt für alle ppt  $\mathcal{A}$

$$\mathbb{W}_s[\text{CompHC}_{\mathcal{A}, \Pi_f}(n) = 1] = \mathbb{W}_s[\mathcal{A}(1^n, I, f(x)) = hc(x)] \leq \frac{1}{2} + \text{negl}(n).$$

# Hardcore-Prädikate

- $hc(x) := \text{lsb}(x)$  (least significant bit) ist ein Hardcoreprädikat für die RSA Einwegpermutation  $\Pi_{RSA}$  (unter der RSA Annahme).
- Es kann kein festes Hardcoreprädikat  $hc$  für alle Einwegfunktionen geben.
- Warum? Sei  $\Pi_f$  Einwegfunktion mit Hardcoreprädikat  $hc$ . Dann ist  $\Pi_g$  mit  $g(x) := f(x) || hc(x)$  auch eine Einwegfunktion, aber  $hc$  kein Hardcoreprädikat von  $\Pi_g$ .
- Aber: Man kann jede Einwegfunktion  $\Pi_f$  zu einer Einwegfunktion  $\Pi_g$  mit einem festen Hardcoreprädikat verändern.

# Goldreich-Levin Hardcore-Prädikat

## Satz von Goldreich-Levin

Sei  $\Pi_f$  eine Einwegpermutation. Dann existiert eine Einwegpermutation  $\Pi_g$  mit Hardcoreprädikat  $hc$ .

**Konstruktion:** (ohne Beweis)

- Sei  $f$  eine Einwegpermutation mit Definitionsbereich  $\{0, 1\}^n$ .
- Sei  $x = x_1 \dots x_n \in \{0, 1\}^n$ . Konstruiere

$$g(x, r) := (f(x), r) \text{ mit } r \in_R \{0, 1\}^n.$$

- Offenbar ist  $g$  ebenfalls eine Einwegpermutation.
- Wir konstruieren ein Hardcore-Prädikat  $hc$  für  $g$  mittels

$$hc(x, r) = \langle x, r \rangle = \sum_{i=1}^n x_i r_i \bmod 2.$$

- Beweis der Hardcore-Eigenschaft ist nicht-trivial.

# Verschlüsselung aus Trapdoor-Einwegpermutation

## Algorithmus $\Pi_{cpa}$

Sei  $\Pi_f$  eine Td-Einwegpermutation mit Hardcore-Prädikat  $hc$ .

- 1 **Gen:**  $(I, td) \leftarrow Gen(1^n)$ . Ausgabe  $pk = I$  und  $sk = td$ .
- 2 **Enc:** Für  $m \in \{0, 1\}$  setze  $x \leftarrow Sample(I)$  und berechne
$$c \leftarrow (f(x), hc(x) \oplus m).$$
- 3 **Dec:** Für Chiffretext  $c = (c_1, c_2)$  berechne  $x := Inv_{td}(c_1)$  und
$$m := c_2 \oplus hc(x).$$

## Intuition:

- $hc(x)$  ist “pseudozufällig” gegeben  $f(x)$ .
- D.h.  $hc(x) \oplus m$  ist ununterscheidbar von 1-Bit One-Time Pad.

# Bsp: Verschlüsselung mit RSA-Td-Einwegpermutation

## Algorithmus $\Pi_{cpa}^{rsa}$

Sei  $\Pi_{rsa}$  die RSA Td-Einwegpermutation mit Hardcore-Prädikat  $hc$ .

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