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 $\mathrm{Ws}[Invert_{\mathcal{A},f}(n) = 1] \leq \mathrm{negl}(n)$.

Spiel Invertieren

$\text{Invert}_{\mathcal{A},f}(n)$		$\mathcal{A}$
Wähle: $x \in_R \{0,1\}^n$		
Berechne: $y \leftarrow f(x)$	$\xrightarrow{(1^n, y)}$	
		Berechne: $x' \in \{0,1\}^n$
Ausgabe: $\begin{cases} 1 & \text{falls } f(x') = y \\ 0 & \text{sonst} \end{cases}$	$\xleftarrow{x'}$	

Die Faktorisierungsannahme

- **Problem:** Existenz von Einwegfunktionen ist ein offenes Problem.
- Konstruktion unter Komplexitätsannahme (z.B. Faktorisierung)
- Verwenden dazu $(N, p, q) \leftarrow GenModulus(1^n)$ von RSA.

Spiel Faktorisierungsspiel $Factor_{\mathcal{A},GenModulus}(n)$

1. $(N, p, q) \leftarrow GenModulus(1^n)$
2. $(p', q') \leftarrow \mathcal{A}(N)$ mit $p', q' > 1$.
3. $Factor_{\mathcal{A},GenModulus}(n) = \begin{cases} 1 & \text{falls } p'q' = N \\ 0 & \text{sonst} \end{cases}.$

Spiel Faktorisieren

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

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

Ausgabe:

$$\begin{cases} 1 & \text{falls } p'q' = N \\ 0 & \text{sonst} \end{cases}$$

$\xrightarrow{(1^n, N)}$

$\xleftarrow{(p', q')}$

$\mathcal{A}$

Berechne:

$p', q' > 1$

Definition Faktorisierungsannahme

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

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

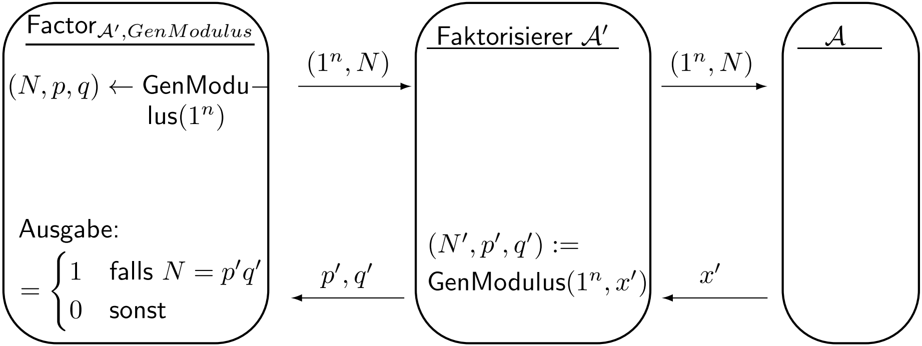
1. $x' \leftarrow \mathcal{A}(1^n, N)$.
2. $(N', p', q') \leftarrow GenModulus(1^n, x')$.

AUSGABE: p', q'

Unter der Faktorisierungsannahme gilt

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

Faktorisieren mit Invertierer für f_{FO}



Trapdoor-Permutationsfamilie

Definition Permutationsfamilie

Eine *Permutationsfamilie* $\Pi_f = (Gen, Samp, f)$ 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 $f : D \rightarrow D$ ist bijektiv.

Definition Trapdoor-Permutationsfamilie

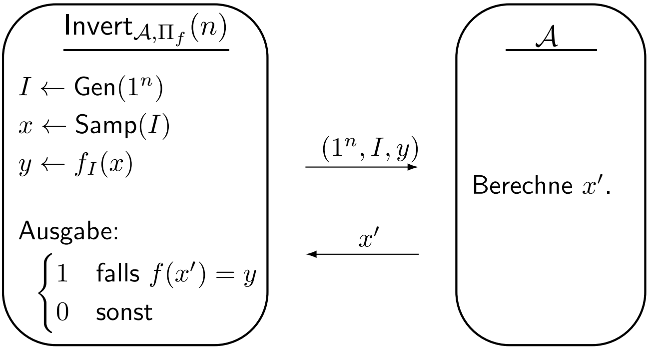
Trapdoor-Permutationsfamilie $\Pi_f = (Gen, Samp, f, Inv)$ besteht aus

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 $x \in D$.

Spiel Invertieren einer Permutation $Invert_{\mathcal{A},\Pi_f}(n)$

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

1. $I \leftarrow Gen(1^n)$, $x \leftarrow Samp(I)$ und $y \leftarrow f(I,x)$.
2. $x' \leftarrow \mathcal{A}(I,y)$.
3. $Invert_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{falls } f(x') = y \\ 0 & \text{sonst} \end{cases}.$



Konstruktion einer Trapdoor-Einwegpermutation

Definition Einweg-Permutation

Eine (Trapdoor-)Permutationsfamilie heißt *(Td-)Einwegpermutation* falls für alle ppt Algorithmen $\mathcal{A}$ gilt $\mathrm{Ws}[Invert_{\mathcal{A},\Pi_f}(n) = 1] \leq \mathrm{negl}(n)$.

Bsp: Trapdoor-Einwegpermutation unter RSA-Annahme

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

Hardcore-Prädikat

Ziel: Destilliere Komplexität des Invertierens auf ein Bit.

Definition Hardcore-Prädikat

Sei Π_f eine Einwegpermutation. 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:

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

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

Spiel zum Berechnen des Hardcore-Prädikats

$\mathsf{CompHC}_{\mathcal{A},\Pi_f}(n)$

$I \leftarrow \mathsf{Gen}(1^n)$

$x \leftarrow \mathsf{Samp}(I)$

$y \leftarrow f_I(x)$

$\xrightarrow{(1^n, I, y)}$

$\mathcal{A}$

Berechne $hc(x)'$.

$\xleftarrow{hc(x)'}$

Ausgabe:

$$\begin{cases} 1 & \text{falls } hc(x) = hc(x)' \\ 0 & \text{sonst} \end{cases}$$

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

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

Goldreich-Levin Hardcore-Prädikat

Satz von Goldreich-Levin

Sei Π_f eine Einwegpermutation. Dann existiert eine Einwegpermutation Π_g mit Hardcoreprädikat hc.

Konstruktion: (ohne Beweis)

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

Sei Π_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 Π_{cpa}^{rsa}

Sei Π_{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
$$m \leftarrow c_2 \oplus hc(r).$$