

Homomorphe Verschlüsselung

Definition Homomorphe Verschlüsselung

Sei Π ein Verschlüsselungsverfahren mit $Enc : G \rightarrow G'$ für Gruppen G, G' . Π heißt *homomorph*, falls $Enc(m_1) \circ_{G'} Enc(m_2)$ eine gültige Verschlüsselung von $m_1 \circ_G m_2$ für alle $m_1, m_2 \in G$ ist.

Bsp:

- **Textbook-RSA** mit $Enc : (\mathbb{Z}_N^*, \cdot) \rightarrow (\mathbb{Z}_N^*, \cdot)$ und

$$m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e \text{ mod } N.$$

- **ElGamal** mit $Enc : (\mathbb{Z}_p^*, \cdot) \rightarrow (\mathbb{Z}_p^*, \cdot) \times (\mathbb{Z}_p^*, \cdot)$ und

$$(g^{y_1}, h^{y_1} m_1) \cdot (g^{y_2}, h^{y_2} m_2) = (g^{y_1+y_2}, h^{y_1+y_2} m_1 m_2).$$

Hier ist $\circ_{G'}$ die komponentenweise Multiplikation in $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$.

- **Goldwasser-Micali** mit $Enc : (\mathbb{Z}_2, +) \rightarrow (\mathbb{Z}_N^*, \cdot)$ und

$$z^{m_1} x_1^2 \cdot z^{m_2} x_2^2 = z^{m_1+m_2 \text{ mod } 2} (x_1 x_2)^2 \text{ mod } N.$$

E-voting mit Paillier

- **Paillier** mit $Enc : (\mathbb{Z}_N, +) \rightarrow (\mathbb{Z}_{N^2}^*, \cdot)$ und
$$(1 + N)^{m_1} r_1^N \cdot (1 + N)^{m_2} r_2^N = (1 + N)^{m_1 + m_2 \bmod N} (r_1 r_2)^N \bmod N^2.$$
Vorteil: $G = (\mathbb{Z}_N, +)$ ist additiv und groß.

Algorithmus E-voting mit Paillier

- Wahlleiter generiert öffentlichen RSA-Modul $N = pq$.
- Wähler $i \in [n]$ mit $n < N$ wählt $v_i = 0$ für NEIN, $v_i = 1$ für JA und sendet an alle anderen Wähler $c_i = (1 + N)^{v_i} r_i^N \bmod N^2$.
- Wähler aggregieren $c := \prod_{i=1}^n c_i \bmod N^2$.
- Wahlleiter erhält c und veröffentlicht $Dec(c) = \sum_{i=1}^n v_i$.

Eigenschaften: (falls alle Parteien sich an das Protokoll halten)

- Wahlleiter erhält c , ohne die einzelnen c_i kennenzulernen.
- Kein Wähler erhält Informationen über die v_i anderer Wähler.
- Berechnung von c ist öffentlich verifizierbar.

Voll homomorphe Verschlüsselung

Definition Voll homomorphe Verschlüsselung

Sei Π ein Verschlüsselungsverfahren mit $Enc : R \rightarrow R'$ für Ringe R, R' . Π heißt *voll homomorph*, falls

- 1 $Enc(m_1) + Enc(m_2)$ eine gültige Verschlüsselung von $m_1 + m_2$
 - 2 $Enc(m_1) \cdot Enc(m_2)$ eine gültige Verschlüsselung von $m_1 \cdot m_2$
- für alle $m_1, m_2 \in R$ ist.

Anwendung: Cloud Computing

- Sende verschlüsselt Algorithmus \mathcal{A} , Eingabe x an einen Server S .
- S berechnet daraus die verschlüsselte Ausgabe $Enc(\mathcal{A}(x))$.
- Erlaubt Auslagern von Berechnungen an S .
- S lernt nichts über das Programm \mathcal{A} oder die Eingabe x .

Erste voll homomorphe Verschlüsselung:

Gentry Verfahren (2009), basierend auf Problemen der Gittertheorie.

Digitale Signaturen

Funktionsweise von digitalen Signaturen:

- Schlüsselgenerierung erzeugt pk, sk .
- Signieren ist Funktion von sk .
- Verifikation ist Funktion von pk .

Idee: Es soll unmöglich sein, ein gültiges Paar von Nachricht m mit zugehöriger Signatur σ zu erzeugen, ohne sk zu kennen.

Eigenschaften digitaler Signaturen.

- **Integrität:** m kann nicht verändert werden, da man keine gültige Signatur zu einem $m' \neq m$ erstellen kann.
- **Authentizität:** Falls σ ein gültige Signatur zu m ist, so kommt die Signatur vom Besitzer des sk .
- **Transferierbarkeit:** Jeder kann die Gültigkeit von (m, σ) überprüfen. Insbesondere kann (m, σ) weitergereicht werden.
- **Nicht-Abstreitbarkeit:** Signierer kann nicht behaupten, dass eine andere Person eine gültige Signatur erzeugt hat.

Definition Signaturverfahren

Definition Signaturverfahren

Ein *Signaturverfahren* ist ein 3-Tupel $(Gen, Sign, Vrfy)$ von ppt Alg mit

- 1 **Gen:** $(pk, sk) \leftarrow Gen(1^n)$.
- 2 **Sign:** Für eine Nachricht $m \in \{0, 1\}^*$ berechne

$$\sigma := Sign_{sk}(m).$$

- 3 **Vrfy:** Für ein Tupel (m, σ) berechne

$$Vrfy_{pk}(m, \sigma) = \begin{cases} 1 & \text{falls } \sigma \text{ gültig ist für } m. \\ 0 & \text{sonst} \end{cases}.$$

Für alle $m \in \{0, 1\}^*$ gilt $Vrfy_{pk}(m, \sigma) = 1$ gdw $\sigma = Sign_{sk}(m)$.

Unfälschbarkeit von Signaturen

Spiel CMA-Spiel $Forge_{\mathcal{A},\Pi}(n)$

Sei Π ein Signaturverfahren mit Angreifer \mathcal{A} .

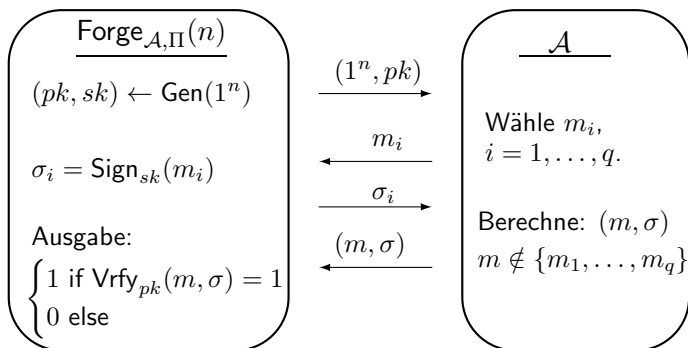
- 1 $(pk, sk) \leftarrow Gen(1^n)$
- 2 $(m, \sigma) \leftarrow \mathcal{A}^{Sign_{sk}(\cdot)}(pk)$, wobei $Sign_{sk}(\cdot)$ ein Signierorakel für beliebige Nachrichten $m' \neq m$ ist.
- 3 $Forge_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{falls } Vrfy_{pk}(m, \sigma) = 1, Sign_{sk}(m) \text{ nicht angefragt} \\ 0 & \text{sonst} \end{cases}$

Definition CMA-Sicherheit

Sei Π ein Signaturverfahren. Π heißt *existentiell unfälschbar* unter *Chosen Message Angriffen* (oder kurz *CMA-sicher*), falls für alle ppt Angreifer \mathcal{A} gilt

$$\text{Ws}[Forge_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

CMA Spiel Forge



Unsicherheit von Textbook RSA Signaturen

Algorithmus Textbook RSA Signaturen

- 1 **Gen:** $(N, e, d) \leftarrow \text{GenRSA}(1^n)$. Setze $pk = (N, e)$, $sk = (N, d)$.
- 2 **Sign:** Für $m \in \mathbb{Z}_N$ berechne $\sigma = m^d \bmod N$.
- 3 **Vrfy:** Für $(m, \sigma) \in \mathbb{Z}_N \times \mathbb{Z}_N$ Ausgabe $\begin{cases} 1 & \text{falls } \sigma^e \stackrel{?}{=} m \bmod N \\ 0 & \text{sonst} \end{cases}$.

Unsicherheit: gegenüber CMA-Angriffen

- Wähle beliebiges $\sigma \in \mathbb{Z}_N$. Berechne $m := \sigma^e \bmod N$.
- Offenbar ist σ eine gültige Signatur für m .
- Angreifer besitzt keine Kontrolle über m (existentielle Fälschung).

Fälschen einer Signatur für ein gewähltes $m \in \mathbb{Z}_N$:

- Wähle $m_1 \in_R \mathbb{Z}_N^* \setminus \{1, m\}$. Berechne $m_2 = \frac{m}{m_1} \bmod N$.
- Lasse m_1, m_2 vom Orakel $\text{Sign}_{sk}(\cdot)$ unterschreiben.
- Seien σ_1, σ_2 die Signaturen. Dann ist
$$\sigma = \sigma_1 \cdot \sigma_2 = m_1^d \cdot m_2^d = (m_1 m_2)^d = m^d \bmod N$$
 gültig für m .

Hashfunktionen und Kollisionen

Definition Hashfunktion

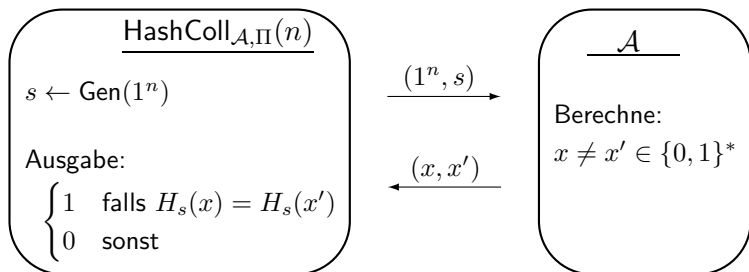
Eine *Hashfunktion* ist ein Paar (Gen, H) von pt Algorithmen mit

- 1 **Gen:** $s \leftarrow Gen(1^n)$. *Gen* ist probabilistisch.
- 2 **H:** Für einen Index s und ein Argument $x \in \{0, 1\}^*$ berechne $H_s(x)$, wobei $H_s : \{0, 1\}^* \rightarrow \{0, 1\}^n, x \mapsto H_s(x)$.

Spiel $HashColl_{\mathcal{A}, \Pi}(n)$

- 1 $s \leftarrow Gen(1^n)$
- 2 $(x, x') \leftarrow \mathcal{A}(s)$
- 3 $HashColl_{\mathcal{A}, \Pi} = \begin{cases} 1 & \text{falls } H_s(x) = H_s(x') \text{ und } x \neq x' \\ 0 & \text{sonst} \end{cases}$.

Kollisionsresistente Hashfunktionen



Definition Kollisionsresistenz

Eine Hashfunktion Π heißt *kollisionsresistent*, falls für alle ppt \mathcal{A} gilt $\mathbb{W}_s[\text{HashColl}_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n)$.

Hashed RSA

Algorithmus Hashed RSA

- 1 Gen:** $(N, e, d, H) \leftarrow \text{GenHashRSA}(1^n)$ mit $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$.
Ausgabe $pk = (N, e, H)$, $sk = (N, d, H)$.
- 2 Sign:** Für $m \in \{0, 1\}^*$ berechne $\sigma = H(m)^d \bmod N$.
- 3 Vrfy:** Für $(m, \sigma) \in \mathbb{Z}_N^2$ Ausgabe 1 gdw $\sigma^e \stackrel{?}{=} H(m) \bmod N$.

Einfacher Angriff:

- Sei $m_1 \neq m_2$ eine Kollision für H ist, d.h. $H(m_1) = H(m_2)$.
- Frage (m_1, σ) an. Dann ist (m_2, σ) eine gültige Fälschung.
- D.h. wir benötigen für H Kollisionsresistenz.

Anmerkung: Sicherheit gegen unsere Angriffe für Textbook RSA

- 1** Wähle $\sigma \in \mathbb{Z}_N$, berechne σ^e . Müssen $m \in H^{-1}(\sigma^e)$ bestimmen.
Übung: Urbildbestimmung ist schwer für kollisionsresistentes H .
- 2** Für ein $m \in \mathbb{Z}_N^*$ benötigen wir m_1, m_2 mit $H(m) = H(m_1) \cdot H(m_2)$ in \mathbb{Z}_N . Scheint Invertierbarkeit von H zu erfordern.