Urbild Angriff auf Inkrementelle Hashfunktionen

AdHash Konstruktion: (Bellare, Micciancio 1997)

- Hashe Nachricht $x = (x_1, \ldots, x_k)$ als
$$H(x) = \sum_{i=1}^{k} h(i, x_i) \bmod M.$$
- **Inkrementell:** Block x_i kann leicht durch x_i' ersetzt werden.
- NASD (Network-Attached Security Disks) Instantiierung: $M = 2^{256}$

Algorithmus: Urbild Angriff auf AdHash

EINGABE: Modul $M = 2^{256}$, Hashwert c

1. Generiere Listen $L_1, \ldots, L_k$ mit $|L_i| = 2^{\frac{n}{\log k + 1}}$.
2. Liste L_i enthält $y_j^{(i)} = h(i, x_j)$ für zufällig gewählte x_j.
3. k-Listen Algorithmus liefert $y_{j_1}^{(1)}, \ldots, y_{j_k}^{(k)}$ mit
$$y_{j_1}^{(1)} + \ldots + y_{j_k}^{(k)} = c \bmod 2^{256} \text{ und } y_{j_i}^{(i)} = h(i, x_{j_i}).$$

AUSGABE: $x = (x_{j_1}, \ldots, x_{j_k})$ mit $H(x) = c \bmod M$

Urbild Angriff auf Inkrementelle Hashfunktionen

Komplexität:

- Naive Urbildberechnung benötigt erwartet 2^{256} H-Auswertungen.
- Für unseren Angriff ist der k-Listen Algorithmus laufzeitbestimmend.
- Auswerten von $k \cdot 2^{\frac{n}{\log k+1}}$ für $k = 128$ liefert $2^7 \cdot 2^{32} = 2^{39}$.
- Allgemein: Erhalten einen Angriff mit Komplexität $\tilde{\mathcal{O}}(2^{2\sqrt{\log M}})$.
- D.h. für 80-Bit Sicherheit muss $M > 2^{1600}$ gewählt werden.

Fälschen von einfachen Ringsignaturen

Idee: Ringsignatur

- Sei $U = \{u_1, \ldots, u_k\}$ eine Menge von Usern.
- Ein User u_i möchte eine Unterschrift im Namen von U leisten.
- Eine Ringsignatur schützt die Anonymität von u_i in U.

Ringsignatur von Back (1997)

Sei H eine Hashfunktion.

1. **Gen:** Generiere RSA Schlüssel (N_i, e_i, d_i) für alle User u_i.
2. **Sign:** User u_i wählt $m_j \in_R \mathbb{Z}_{N_j}$, $j \neq i$, Nachricht m, und berechnet

$$m_i = \left(H(m) \oplus \bigoplus_{j \neq i} (m_j^{e_j} \bmod N_j)\right)^{d_i} \bmod N_i.$$

   Ausgabe von (m, σ) mit der Signatur $\sigma = (m_1, \ldots, m_k)$.
3. **Vrfy:** Prüfe für (m, σ) die Identität

$$\bigoplus_{i=1}^{k} (m_i^{e_i} \bmod N_i) \stackrel{?}{=} H(m).$$

Fälschen von Ringsignaturen

Algorithmus Universelles Fälschen von Ringsignaturen

EINGABE: Nachricht m, (N_i, e_i) für $i = 1, \ldots, k$

1. Berechne Listen L_i für $i = 1, \ldots, k$ mit Elementen
$$x_j^{(i)} = m_j^{e_i} \bmod N_i \text{ für } m_j \in_R \mathbb{Z}_{N_i}.$$
2. k-Listen Algorithmus liefert $x_{j_1}^{(1)}, \ldots, x_{j_k}^{(k)}$ mit
$$x_{j_1}^{(1)} \oplus \ldots \oplus x_{j_k}^{(k)} = H(m).$$

AUSGABE: (m, σ) mit $\sigma = (m_{j_1}, \ldots, m_{j_k})$.

Komplexität:

- Sei $N = \max_i\{N_i\}$. Wir erhalten Komplexität $\mathcal{O}(k \cdot 2^{\frac{\log N}{\log k + 1}})$.
- D.h. für $k = \theta(\log N)$ erhalten wir einen subexponentiellen Angriff.

Polynomielle Vielfache mit kleinem Gewicht

Definition Gewicht eines Polynoms

Sei $p(x) = \sum_{i=0}^{n} p_i x^i \in \mathbb{F}_2[x]$. Das *Gewicht* $w(p)$ von $p(x)$ ist definiert als das Hamminggewicht des Koeffizientenvektors von $p(x)$, d.h.

$$\mathrm{wt}(p) = \mathrm{wt}((p_0, \ldots, p_n)).$$

Anwendung: Bei sogenannten Korrelationsattacken auf Stromchiffren benötigt man Polynomvielfache sehr kleinen Gewichts.

Problem Polynomvielfache mit kleinem Gewicht

Gegeben: $p(x) \in \mathbb{F}_2[x]$ irreduzibel vom Grad n,
Gradschranke $d > n$, Gewicht k

Gesucht: $m(x) \in \mathbb{F}_2[x]$ mit $p(x) \mid m(x)$, Grad $\leq d$ und $\mathrm{wt}(m) \leq k$.

Konstruktion von Polynomvielfachen

Wir identifizieren Polynome in $\mathbb{F}_2[x]$ mit ihren Koeffizientenvektoren.

Algorithmus Polynomvielfache

EINGABE: $p(x) \in \mathbb{F}_2[x]$, Gewicht k

1. Setze die Gradschranke $d := 2^{\frac{n}{\log k + 1}}$
2. Generiere Listen L_i, $i = 1, \ldots, k$ mit Elementen der Form
$$y_j^{(i)} = x^{a_j} \bmod p(x) \text{ für zufällig gewählte } a_j \leq d.$$
3. k-Listen Algorithmus liefert $y_{j_1}^{(1)}, \ldots, y_{j_k}^{(k)}$ mit
$$y_{j_1}^{(1)} \oplus \ldots \oplus y_{j_k}^{(k)} = \mathbf{0}.$$

AUSGABE: $m(x) = x^{a_{j_1}} + \ldots + x^{a_{j_k}}$

Konstruktion von Polynomvielfachen

Korrektheit:

- Wir definieren $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/p(x)$. Addition zweier Polynome in $\mathbb{F}_{2^n}$ entspricht dem XOR ihrer Koeffizientenvektoren.
- Nach Konstruktion gilt $m(x) = x^{a_{j_1}} + \ldots + x^{a_{j_k}} = 0$ in $\mathbb{F}_{2^n}$.
- D.h. $p(x)$ muss $m(x)$ teilen.
- Wegen $a_j \leq d$ besitzt $m(x)$ Grad höchstens d.
- Ferner besteht $m(x)$ aus höchstens k Monomen.
- Damit besitzt $m(x)$ Gewicht höchstens k.
- Für die Listengröße im k-Listen Alg. benötigen wir $d = 2^{\frac{n}{\log k+1}}$.
- D.h. unser Algorithmus funktioniert nur für hinreichend großes d.

Komplexität:

- Der k-Listen Algorithmus liefert Komplexität $\tilde{\mathcal{O}}(k \cdot 2^{\frac{n}{\log k+1}})$.
- Bsp.: $\mathrm{grad}(p) = 120$ und wir suchen Vielfaches mit Gewicht $k = 4$.
- Wir wählen $d = 2^{\frac{n}{\log k+1}} = 2^{\frac{120}{3}} = 2^{40}$ erhalten $k \cdot 2^{\frac{n}{\log k+1}} = 2^{42}$.

k-Listen Problem über $\mathbb{F}_2^n$ für $k \geq n$

Problem Generalized Birthday für $k \geq n$

Gegeben: $L_1, \ldots, L_k$ mit Elementen aus $\mathbb{F}_2^n$, $|L_i| \geq 2$, $k \geq n$.
Gesucht: $\mathbf{x}_1 \in L_1, \ldots, \mathbf{x}_k \in L_k$ mit $\mathbf{x}_1 \oplus \ldots \oplus \mathbf{x}_k = \mathbf{0}$

Idee: (Algorithmus von Bellare, Micciancio 1997)

- ObdA $L_i = \{\mathbf{x}_{i,0}, \mathbf{x}_{i,1}\}$ für alle i, sonst entferne Elemente aus L_i.
- Definiere $b_i = \begin{cases} 0 & \text{falls } \mathbf{x}_{i,0} \text{ in } L_i \text{ ausgewählt wird.} \\ 1 & \text{falls } \mathbf{x}_{i,1} \text{ in } L_i \text{ ausgewählt wird.} \end{cases}$
- D.h. wird müssen $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{F}_2^n$ finden mit

$$b_1\mathbf{x}_{1,1} + (1-b_1)\mathbf{x}_{1,0} + \ldots + b_k\mathbf{x}_{k,1} + (1-b_k)\mathbf{x}_{k,0} = \mathbf{0}$$
$$\Leftrightarrow \quad b_1(\mathbf{x}_{1,1} - \mathbf{x}_{1,0}) + \ldots + b_k(\mathbf{x}_{k,1} - \mathbf{x}_{k,0}) = -(\mathbf{x}_{1,0} + \ldots + \mathbf{x}_{k,0})$$

- Dies ist ein lineares Gleichungssystem in den b_i.
- Falls die Matrix definiert durch die Vektoren $\mathbf{x}_{i,1} - \mathbf{x}_{i,0}$ vollen Rang besitzt, so können wir das System in Zeit $\mathcal{O}(n^3 + kn)$ lösen.