RUHR UNIVERSITÄT BOCHUM

RUB

Lehrstuhl für Kryptologie und IT-Sicherheit
Prof. Dr. Alexander May
Mathias Herrmann, Enrico Thomae

Hausübungen zur Vorlesung

Kryptanalyse

WS 2010/2011

Blatt 11 / 22. Dezember 2010 / Abgabe bis spätestens 12. Januar 2011, 10 Uhr (vor der Übung)

AUFGABE 1 (4 Punkte)**:**
Geben Sie einen Algorithmus zum Berechnen des diskreten Logarithmus an, der den k-Listen Algorithmus verwendet. Sei dazu p eine n-Bit Primzahl und $b \in \langle g \rangle$ mit $g \in \mathbb{Z}_p^*$. Ihr Algorithmus soll $\text{dlog}_g b$ ausgeben. Bestimmen Sie die optimale Listentiefe k und geben Sie die Komplexität ihres Algorithmus in Abhängigkeit von n an.

AUFGABE 2 (4 Punkte)**:**
Bestimmen Sie die Determinante folgender Matrix für $a \neq b \in \mathbb{Z}$.

$$\begin{pmatrix} a & b & b & \cdots & b \\ b & a & b & \cdots & b \\ b & b & a & \cdots & b \\ \vdots & \vdots & \vdots & & \vdots \\ b & b & b & \cdots & a \end{pmatrix}$$

AUFGABE 3 (10 Punkte)**:**
Implementieren Sie den in der Vorlesung beschriebenen Angriff auf die AdHash Funktion. Zur Erinnerung: Für eine Nachricht $x = (x_1, \ldots, x_k)$ berechnet sich der AdHash als

$$H(x) = \sum_{i=1}^{k} h(i, x_i) \bmod M,$$

für einen Modulus M und eine Kompressionsfunktion h. In der folgenden Implementierung wird als Modulus 2^{30} und als Kompressionsfunktion MD5 verwendet. Um MD5 in `sage` zu verwenden:

```
## Import der Hashfunktion
from Crypto.Hash import MD5;

## Erzeugen eines MD5 Objekts
m = MD5.new("test");

## Ausgabe des Hashwertes als Integer
val = int(m.hexdigest(),16)
```

Wir werden den k-Listen Algorithmus für $k = 4$ implementieren. Hier einige Meilensteine um den Angriff auf AdHash durchzuführen. Ziel ist es eine Urbild für den Wert $c = 621541643$ zu finden.
(Sie müssen den Meilensteinen nicht folgen. Gerne können Sie den Algorithmus nach Ihren eigenen Ideen implementieren!)

1. Schreiben Sie eine Funktion `adHash` die als Parameter eine Liste `x` der x_i, sowie einen Modulus M erhält und den Wert $H(x)$ zurückgibt.

2. Innerhalb des join Operator im k-Listen Algorithmus muss eine der beiden Listen sortiert werden. Diese Sortierung ist jedoch modulo 2^ℓ für einen Parameter ℓ. Schreiben Sie eine Funktion, die eine Liste so sortiert oder übergeben Sie einen entsprechenden Parameter an die `sort` Methode der Liste.

3. Eine sortierte Liste hat den Vorteil, dass eine binäre Suche durchgeführt werden kann. Implementieren Sie eine Funktion `binary_search` die als Parameter eine Liste, ein gesuchtes Element und den Parameter ℓ der Sortierung erhält.

4. Implementieren Sie nun den join Operator, der zwei Listen L_1, L_2 und den Kürzungsparameter ℓ erhält. Beachten Sie, dass die binäre Suche nur einen Treffer zurückgibt, Sie jedoch unter Umständen mehrere Treffer haben.

5. Verändern Sie Ihren join Operator so, dass am Ende nicht als Summe Null, sondern der Zielwert entsteht.

6. Erzeugen Sie 4 Listen L_i indem Sie zufällige r_j wählen und $h(i, r_j)$ berechnen und führen Sie den implementierten k-Listen Algorithmus aus um ein Urbild für $c = 621541643$ zu finden.

Bonus:[5 Punkte] Implementieren Sie den Algorithmus für beliebiges k.

Hinweis: Sie können die `sage` Programm-Codes per Email direkt an **ilya.ozerov@rub.de** schicken.