Rechnersicherheit - Übung 04

Dennis Hägler und Martin Görick
Tutor: Stefan Pfeiffer

13. Mai 2015

Aufgabe 1

Zuerst wird mittels einer Blockchiffre aus dem geheimen Schlüssel K zwei Schlüssel K_1 und K_2 generiert. Danach wird die MAC nach Abbildung 1 generiert.

CHIP ist die Blockchiffre mit dem geheimen Schlüssel K MBS_{Tlen} ist die Länge der MAC

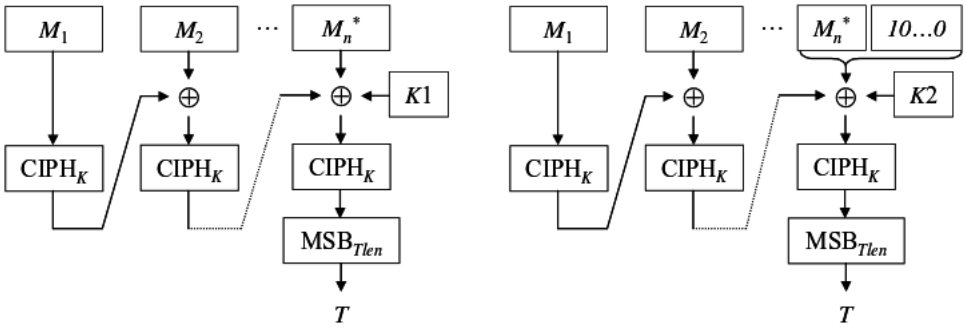

Abbildung 1: MAC generierung

Zur Verifikation wird aus dem K, M ein T' berechnet und mit dem übermittelten T verglichen.

Quelle Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Morris Dworkin, NIST Special Publication 800-38B, May 2005.

Aufgabe 2

1 Ein Empfänger welcher den Schlüssel zum vergleichen hat, könnte mit einer zufälligen Signatur, zufällige aussehende Texte erstellen. Damit kann verhindert werden, dass der Sender zufällige Texte wie Schlüssel signieren kann, da nicht sicher ist ob die zufälligen Texte vom Sender sind.

2 Ein Empfänger will aus einem Text m und einer Signatur s ein eigenes Dokument m'' mit der gültigen Signatur s'' erzeugen. Unter einem Vorwand soll der Sender ein Dokument $m' = m^{-1} * m''$ signieren. Der Sender soll keinen Verdacht schöpfen, da m' unsinnig ist. Mit der erhaltenen Signatur s' unter mod n gerechnet und mit dem privaten Schlüssel d erhält der Angreifer ein s'' wie folgt.

$$s'' = s * s' = m^d * m'^d = (m * m')^d = (m * m6{-}1 * m'')^d = m''^d$$

Quelle RSA-Signatur,Hans Werner Lang, 2014, `http://www.iti.fh-flensburg.de/lang/krypto/protokolle/rsa-signatur.htm`

Aufgabe 3

a) In der Menge $\mathbb{Z}_p^*(g)$ sind alle Möglichkeiten enthalten, den geheimen Schlüssel zu wählen. Dieser wird aus 4. a gewählt wie beschrieben geht a von 1 bis $q-1$. Theoretisch könnte jedoch auch q als geheimer Schlüssel gewählt werden. Diese Zahl ist jedoch bekannt, weshalb diese nicht verwendet wird. Also sind in der Menge $\mathbb{Z}_p^*(g)$ genau q Elemente mit allen möglichen Elementen von A sowie g^q.

b) q muss mindestens eine Länge von 224 Bit und p 2048 Bit Länge besitzen.

Aufgabe 4

- G ist ein fester Erzeuger der n-Torsionsuntergruppe der Kurve
- n ist die Ordnung des Punktes G
- L_n die Bitlänge der Gruppenordnung n

Schlüsselerzeugung

- privat key (sk) ist ein zufällige Ganzzahl zwischen 1 und $n-1$
- public key $pk = sk * G$

Signaturerzeugung

Signatur für Nachricht m erzeugen mit Hashfunktion H.

1. berechne $e = H(m)$ und definiere z mit den L_n hochwertigsten Bits von e
2. wähle zufällige Ganzzahl k von 1 bis n-1
3. berechne $r = x_1 (\mathrm{mod}\ n)$, wobei $(x_1, y_1) = k * G$, wenn $r = 0$ gehe zu 2.
4. berechne $s = k^{-1}(z + r * sk)(\mathrm{mod}\ n)$($z$ wird in eine Ganzzahl umgewandelt), wenn $s = 0$ gehe zu 2.
5. Signatur ist das Paar (r, s)

Überprüfen der Signatur

Wenn nicht sicher ist ob der pk richtig erzeugt wurde muss dieser überprüft werden.

- Überprüfen ob pk nicht das neutrale Element der Kurve ist und es valide Koordinaten.
- Überprüfen ob pk auf der Kurve liegt.
- überprüfe ob n*pk = neutrales Element, es wird überprüft ob pk ein Vielfaches von G ist.

Signatur überprüfen in den Folgenden Schritten:

- überprüfen ob r und s ganze Zahlen und im Intervall von 1 bis $n-1$, wenn nicht Signatur ungültig
- berechne $e = H(m)$ und definiere z mit den L_n hochwertigsten Bits von e
- berechne $w = s^{-1} (\mathrm{mod}\ n)$
- berechne $u_1 = z * w (\mathrm{mod}\ n)$ und $u_2 = r * w (\mathrm{mod}\ n)$
- berechne $(x_1, y_1) = u_1 G + u_2 pk$
- Signatur ist gültig, wenn $r = x_1 (\mathrm{mod}\ n)$

Quelle Wikipedia `http://de.wikipedia.org/wiki/Elliptic_Curve_DSA`