Dokumentation Verschlüsslungsverfahren

Christopher Besch und Katharina Libner

12. März 2021

Inhaltsverzeichnis

1 Abstract

Es wird ein symmetrisches Verschlüsselungsverfahren vorgestellt. Mit Hilfe einer Stromverschlüsslung wird die Bitanzahl eines Hashwerts erweitert und anschließend ein Schlüsselfluss erzeugt, der den KLartext mit XOR verschlüsselt.

2 Einleitung

Symmetrische Verschlüsselungsverfahren sind dadurch gekennzeichnet, dass der Schlüssel zur Ver- und Entschlüsslung gleich bleibt. Der Schlüssel soll daher gut bearbeitet und für einen Angriff wenig Fläche schaffen. Zunächst wird das Passwort auf einen Hashwert abgebildet. Dessen Bitanzahl wird durch eine Stromverschlüsselung erweitert und auf einen Seed dargestellt. Aus diesem Seed entsteht ein pseudo zufälliger Schlüsselstrom, der den endgültigen Schlüssel für XOR liefert.

3 Verschlüsslungsablauf

Jegliche Eingabe wird in Binär konvertiert.

3.1 Hashfunktion

3.2 Stromverschlüsselung

3.3 Pseudozufallsgenerator

Ziel ist es einen „zufälligen“ Schlüssel zu erzeugen, der unendlich lang werden kann, um unbeschränkt große Bitanzahlen zu verschlüsseln.

3.3.1 Linearer Kongruenzgenerator

$$x_i = ((x_{i-1} \cdot a) + c) \bmod 2^n$$

x_i ist ein neuer „Zufallswert“, der dem Schlüssel zugeordnet wird. x_{i-1} ist dementsprechend der vorherige Wert, der bestimmt wurde. Da jeder Wert aus der Menge aller x_i erst durch einen vorherigen Wert entsteht, muss es einen Startwert geben: $x_0 = seed$. n sei aus $\mathbb{N}$ und markiert die Bitintervallgrenze in der x_i liegt. Intervall: $[0, n]$. c sei aus $\mathbb{N} < 2\ n$ und dient als Summand. a sei in $\mathbb{N} < 2\ n$ und dienst als Multiplikator.

3.3.2 Enstehung des Schlüssel.

Natürlich erkennt man schnell, dass durch mehrere bekannten Werte aus der Menge der x_i und ein wenig geschickte Mathematik alle Parameter und damit auch der *seed* gefunden werden können. Um dem aus dem Weg zu gehen, wird dem Schlüssel nur die rechte Bithälfte von jedem Wert x_i zugeordnet. Der Generator arbeitet jedoch mit dem ganzen Werten von x_i weiter. Beispiel:

$$B_{x_i} = 10011111$$

$$B_{x_s} = 1111$$

Der Schlüssel s besteht demnach aus allen x_s, die aneinandergereiht werden.

$$s = x_{s_1}, x_{s_2}, x_{s_3}, x_{s_4}, ...$$

3.3.3 Abscheinden der linken Bithälfte

Der Wert von x_s entsteht durch eine Art Maske oder Folie, die über x_i gelegt wird. Diese ist so aufgebaut, dass durch sie nur die rechte Bitseite von x_i durchschimmert und x_s zugeordnet wird. Im Programm wird mit dem Vergleichsinstrument «and»gearbeitet. Es werden die Bits von der Maske mit den der von x_i verglichen. Wenn beide an der gleichen Stelle den gleichen Bitwert haben, wird dieser übernommen, ist dies nicht der Fall gilt für x_s an der Stelle 0. Um nur die rechte Seite zu übernehmen, muss also eine gleichlange Bitmaske m erzeugt werden, die von links bis zur Mitte aus Nullen und von der Mitte bis rechts aus Einsen besteht. Beispiel:

$$B_{x_i} = 11111001SS$$

$$B_{x_i} = 00001111$$

$$B_{x_s} = 00001001$$

3.3.4 Pseudozufallsgenerator als Klasse

Der Generator wird als Klasse erzeugt. Als Eingabe bekommt er die Parameter a, b, n und den *seed*. Der Vorteil an einer Klasse ist, dass jeder letzte Wert von x_i gespeichert wird. Sodass man innerhalb der Klasse mit Hilfe einer Methode jeden nächsten Teilschlüssel k_i mit der Bitanzahl n erzeugen kann. Und die Möglichkeit besteht den zusammengesetzten Schlüssel k aus den Teilschlüsseln k_i unbeschränkt lang zu wählen.

4 XOR

XOR verschlüsselt die Bits des Textes mit den des gleichlangen Schlüssels. Dabei werden Schlüssel und Text gegenübergestellt. Dabei kann die „Bithierarchie“ durch folgende Wahrheitstabelle veranschaulicht werden.

Text	Schlüssel	Geheimtext
0	0	0
1	1	0
1	0	1

Aus 011 wird also 001. Möchte man den Geheimtext wieder Entschlüsseln wird Gleiches verwendet.

Geheimtext	Schlüssel	Text
0	0	0
0	1	1
1	0	1

5 Fazit

6 Beispiele