Programm zu IDEA

Der Zahlenraum des Verfahrens enthält 65536 Zahlen. Wegen der notwendigen Umkehrbarkeit sowohl der Addition als auch der Multiplikation arbeitet man bei der

Addition mit 0, 1, 2, ... , 65535

Die Addition arbeitet also "normal" modulo $2^{16} = 65536$

Nähme man bei der Multiplikation denselben Zahlenraum, erhielte man zu der Zahl 0 kein Inverses. Im Sinne der Verschlüsselung: Beim Verschlüsseln ginge die Information verloren, da sie sich nicht mehr wieder herstellen ließe. Daher verwendet man für die

Multiplikation 1, 2, 3, ... , 65536

Bei der Multiplikation wird also mit der Zahl $2^{16} + 1 = 65537$ modulo gerechnet, statt der 0 verwendet man die 65536.

Bestimmung des Multiplikationsinversen

Ein Problem ist aber nicht nur die Existenz, sondern auch die Bestimmung des Multiplikationsinversen. Da aber 65537 Primzahl ist, kann keine der zu verschlüsselnden Zahlen ein Teiler sein. Daher ist die Bestimmung des Inversen mit dem „Kleinen Fermatschen Satz“ sehr einfach:

Nach dem „Kleinen Satz von Fermat“ gilt für m>0 und m teilerfremd zu a:

$$(a^{phi(m)}) \bmod m = 1$$

und damit ist

$$(a^{phi(m)-1}) \bmod m = a^{-1}$$

ein Multiplikationsinverses zu a.
Es sind dabei :
phi(m) = Anzahl der zu m teilerfremden Zahlen kleiner als m. Da m = 65537 in unserem Fall eine Primzahl ist, gilt phi(m) = m - 1 = 65536

Das Blockschema zum Programm:

Teil 1 : Schlüsselerzeugung :

Zunächst einmal muss der 128 – bit – Schlüssel erstellt werden, der die eigentliche Grundlage des Verschlüsselungsverfahrens ist. Dazu :

Erzeugen von 8 Schlüsselzahlen zwischen 0 und 65535. Die Zahlen können sowohl eingegeben, als auch zufällig erstellt werden.

Teil 2 : Erzeugen der Schlüsselliste :

Die Generierung der Schlüssel ist damit aber leider noch nicht fertig, da der Schlüssel in den 6 Verschlüsselungsrunden, die von IDEA durchlaufen werden, jeweils in veränderter Form verwendet wird: Im zweiten Achterblock werden die 128 um 25 bit nach links „rotiert“, im zweiten wieder 25 usw.
Benötigt wird daher eine Funktion zum Rotieren um 25 bit.

Rotiere die 128 bit um 25 bit.

25 bit sind mehr als die 16 bit eines der Teilschlüssel. Daher werden zunächst einmal die Teilschlüssel um einen Teilschlüssel nach links (also zu höherer Wertigkeit) verschoben. Außerdem werden weitere 25 – 16 = 9 bit innerhalb der Schlüssel verschoben. Wäre der Teilschlüssel nicht lang genug würden diese bit vorn „herausfallen“, bei 32-bit-Zahlen taucht das Problem allerdings nicht auf. Sie stehen allerdings an „unzulässigen Positionen, müssen daher vorn herausgeschnitten werden und dem vorderen Teilschlüssel hinzugefügt werden. Das geht glücklicherweise sehr einfach durch die Divisions, um den vorderen Teil zu bekommen und die modulo-Funktion, um den hinteren Teil zu bekommen.
Nun ist das Verfahren zum Erstellen der 52 Teilschlüssel für die 8 Verschlüsselungsrunden á 6 benötigten Teilschlüsseln zuzüglich der 4 Teilschlüssel für die Ausgangstransformation (alles soll symmetrisch bleiben!) fertig.

Teil 3 : Erzeugen der Umkehrschlüsselliste :

Nun haben wir zwar den Schlüssel, aber damit können wir die Daten nicht wieder herstellen. Dafür brauchen wir den Umkehrschlüssel !
Er wird aus dem ursprünglichen Schlüssel dadurch erzeugt, dass man jeweils die inverse Zahl der Operation für die jeweilige Schlüsselposition erstellt. Das sind an den Positionen, an denen ...

- multipliziert wurde, die Multiplikationsinversen (modulo 65537),
- addiert wurde, die Additionsinversen (modulo 65536),
- xor- verknüpft wurde, dieselben Zahlen (xor ist selbstinvers).

Benötigt wird daher eine Funktion zum Erzeugen dieser Umkehrschlüssel.

Teil 4 :Das eigentliche Verfahren :

Transformation Es werden die im Schema dargestellten Funktionen so verknüpft, dass zu den eingegebenen 4 Datenzahlen und 4 Schlüsselzahlen die 4 Werte bestimmt werden.

Entsprechendes gilt für die Ausgangstransformation.