Anhang zu §2 Verschlüsselung von Botschaften mit invertierbaren Matrizen über Z_p

Bei der Matrix-Multiplikation ist der Wert an der Stelle i, j im Produkt von mehreren Matrix-Elementen abhängig. Verschlüsseln wir eine Botschaft, indem wir sie zuerst in einen Zahlencode übersetzen, diese Zahlen als Matrix arrangieren und dann mit einer Verschlüsselungsmatrix von links multiplizieren, so wird derselbe Buchstabe nicht jedesmal in denselben anderen umgesetzt wie etwa bei der Caesar-Methode oder einer beliebigen anderen monoalphabetischen Verschlüsselung. Damit hat man keine Chance, mit rein statistischen Methoden zB den häufigsten Buchstaben zu erkennen, der ja in deutschen Texten immer ein 'e' ist. Diesen grossen Vorteil zeigt schon das erste Beispiel ganz deutlich.

Zuerst wollen wir aber vereinbaren, wie wir einen Text in Zahlen umsetzen. 31 Zeichen erlauben es, mehr oder weniger alles bequem mitzuteilen. Wir rechnen also modulo 31 und benützen die folgende Tabelle:

0	!
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z
27	
28	.
29	,
30	?

Das folgende Beispiel soll alle Aspekte zeigen. Wir schreiben die Botschaft mit unserem Alphabet von 31 Zeichen und verteilen das Ganze auf mehrere gleich lange Zeilen:

```
I C H _ W I L L _ E I N
E N _ T E X T _ C O D I
E R E N ! _ G U B L E R
```

Unsere Übersetzungstabelle liefert die folgende 3 x 12 - Matrix :

$$\begin{matrix} 9 & 3 & 8 & 27 & 23 & 9 & 12 & 12 & 27 & 5 & 9 & 14 \\ 5 & 14 & 27 & 20 & 5 & 24 & 20 & 27 & 3 & 15 & 4 & 9 \\ 5 & 18 & 5 & 14 & 0 & 27 & 7 & 21 & 2 & 12 & 5 & 18 \end{matrix}$$

Diese Matrix nenen wir b (b wie 'Botschaft'). Nun brauchen wir eine invertierbare 3x3-Matrix c mit Koëffizienten aus Z_{31} . Wir nehmen die folgende:

$$\begin{matrix} 7 & -6 & 5 \\ 5 & 3 & -4 \\ 8 & 2 & 3 \end{matrix} = \begin{matrix} 7 & 25 & 5 \\ 5 & 3 & 27 \\ 8 & 2 & 3 \end{matrix} = c$$

Unser TR bestätigt mit c^{-1} oder rref(c) oder mod(det(c) , 31) , dass diese Matrix invertierbar ist und sich daher für unsere Zwecke eignet.

Nun berechnen wir die verschlüsselte Botschaft mit mod(c·b , 31) -> g (g wie 'geheim'). Es ist

$$g = \begin{matrix} 27 & 27 & 12 & 15 & 7 & 23 & 30 & 27 & 26 & 5 & 2 & 10 \\ 9 & 16 & 8 & 15 & 6 & 9 & 30 & 26 & 12 & 22 & 6 & 25 \\ 4 & 13 & 9 & 19 & 8 & 15 & 2 & 27 & 11 & 13 & 2 & 29 \end{matrix}$$

Wir müssen nicht preisgeben, dass wir das aus einer 3x12-Matrix erhalten haben. 36 könnte ja auch von 4 Zeilen à 9 Zeichen herkommen, was dann mit einer 4x4-Matrix verschlüsselt worden wäre. Wir schicken also den folgenden Geheimtext ab:

```
_ _ L O G W ? _ Z
E B J I P H O F I
? Z L V F Y D M I
S H O B _ K M B ,
```

Der Empfänger kann das wieder entschlüsseln, wenn er die Matrix c^{-1} kennt (gemeint ist damit natürlich die inverse Matrix über dem Zahlkörper Z_{31}). Er übersetzt den Text wieder in die 3x12-Matrix g und rechnet

$$c^{-1} \cdot g = c^{-1} \cdot (c \cdot b) = (c^{-1} \cdot c) \cdot b = b$$

und erhält damit wieder die Originalbotschaft.

c^{-1} könnten wir mit dem Gauss-Algorithmus bestimmen, wir müssen dabei einfach in Z_{31} rechnen. Unser TR hilft uns auf den ersten Blick nicht viel, er kennt zwar den Befehl rref(c) , aber er rechnet dabei in den reellen Zahlen. Dies tut er auch, wenn wir mit dem Befehl c^{-1} direkt die inverse Matrix berechnen lassen.

Die folgenden Betrachtungen sollen Sie auf eine Spur führen, wie wir die Berechnung der inversen Matrix über einem Zahlkörper Z_p mit einer neuen Funktion "matinver (m , p)" automatisieren könnten.

Wir suchen also die Inverse Matrix zu c in $M^{3x3}(Z_{31})$:

$$c = \begin{pmatrix} 7 & 25 & 5 \\ 5 & 3 & 27 \\ 8 & 2 & 3 \end{pmatrix}$$

Unser TR liefert uns nach der Eingabe von c^{-1} :

$$c^{-1} = 331^{-1} \cdot \begin{pmatrix} 17 & 28 & 9 \\ -47 & -19 & 53 \\ -14 & -62 & 51 \end{pmatrix}$$

Was ist aber 331^{-1} in Z_{31} ? Dafür haben wir bereits eine Funktion programmiert:

euklidin (331 , 31) liefert den Wert 3

Es ist also $3 \cdot 331 = 1$ modulo 31 , wir dürfen somit die Matrix c^{-1} mit $3 \cdot 331$ multiplizieren, ohne dass wir ihren Wert in Z_{31} verändern! Damit wird sie aber ganzzahlig, und wir müssen diese ganzzahligen Werte nur noch modulo 31 nehmen:

mod ($3 \cdot 331 \cdot c^{-1}$, 31) --> d

liefert uns die inverse Matrix $d = c^{-1}$, die wir für das Entschlüsseln brauchen ! Es ist

$$d = \begin{pmatrix} 20 & 22 & 27 \\ 14 & 5 & 4 \\ 20 & 0 & 29 \end{pmatrix}$$

Testen Sie dieses Ergebnis mit der Eingabe von mod (d · c , 31) !

d · g liefert also wieder die Originalbotschaft b .

Der folgende Text basiert auf unserem Code modulo 31 :

```
B Q Z _ B . V L A Z ? Y I Y W ,
G ? ? D _ H Y O V W _ N M Q ! J
O S . X D W J S H A S I H T F Z
A W B
```

Verschlüsselt wurde er mit der Matrix

```
18  26  25
 2   9  25
27   6  14
```

Aufträge:

1. Entschlüsseln Sie den obenstehenden Text. Setzen Sie dabei soweit wie möglich unseren Luxus-TR ein.
2. Erzeugen Sie selber eine invertierbare 3x3-Matrix über Z_{31} und verschlüsseln Sie damit einen eigenen Text, der eine Länge von mindestens 21 Zeichen aufweisen soll. Halten Sie Ihren Originaltext, die Botschaft b und die Verschlüsselungsmatrix c auf einem A4-Blatt fest.
3. Notieren Sie den verschlüsselten Text und die verwendete Verschlüsselungsmatrix auf ein separates A4-Blatt (ähnlich wie oben auf dieser Seite). Versehen Sie das Blatt vor der Abgabe noch mit Ihrem Namen.
4. Berechnen Sie noch die Inverse ihrer Verschlüsselungsmatrix in $M^{3x3}(Z_{31})$. Notieren Sie $d = c^{-1}$ auf dem Blatt mit dem Originaltext und den Matrizen b und c .
5. Warum ist die Länge des obigen Textes ungeschickt gewählt ? Wie hätte man das leicht verbessern können ?

```
Define LibPub randmodmat(n,p) =
Func
:Local m
:mod(randMat(n,n),p) → m
:While mod(det(m),p) = 0
:   mod(randMat(n,n),p) → m
:EndWhile
:Return m
:EndFunc
```

Input:

n Kantenlänge der Matrix

p rechne modulo p

Output:

eine invertierbare Zufallsmatrix m

```
Define LibPub matinver(m,p) =
Func
:Local d,c
:mod(m,p) → m
:det(m)→d
:If  mod(d,p) = 0  Then
:   Return 0*m
:EndIf
:euklidin(d,p) → c
:d*c*m^(–1) → m
:Return mod(m,p)
:EndFunc
```

Input:

m eine quadratische Matrix

p rechne modulo p

Output:

die inverse Matrix modulo p falls m invertierbar ist, die Nullmatrix sonst

```
Define LibPub randseedmat(n,p,s) =
Func
:RandSeed s
:Local m
:randMat(n,n) → m
:Local i,j
:Loop
:  For i,1,n,1
:    For j,1,n,1
:      randInt(–1,p) → m[i,j]
:    EndFor
:  EndFor
:  If  mod(det(m),p) ≠ 0  Then
:     Exit
:  EndIf
:EndLoop
:Return m
:EndFunc
```

Input:

n Kantenlänge der Matrix

p rechne modulo p

s seed für RandSeed

Output:

eine invertierbare Zufallsmatrix m