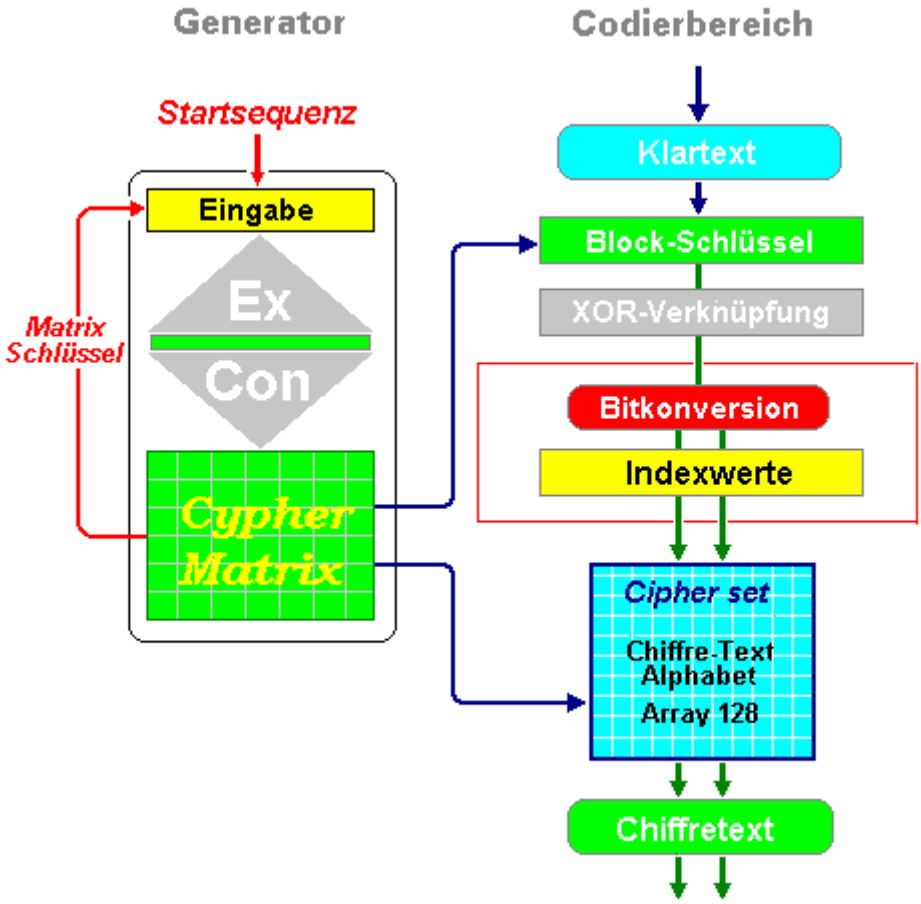
„Codegraphie“ als Teil der Kryptographie

(Ernst Erich Schnoor)

Mit dem **CypherMatrix** Verfahren – Bezeichnung vom Autor - werden neue Zusammenhänge in der Kryptographie aufgezeigt. Bedingt durch „Bitsysteme“ und „Bit-Konversionen“ entstehen neue Funktionsbereiche, die in vielen Fällen bisherige „primitives“ ersetzen [#1]. Die aktuelle Kryptographie ist eine reine Buchstabenverschlüsselung [#2] im Rahmen eines einheitlichen Ordnungssystem, dem Bitsystems zur Basis 8. Das CypherMatrix Verfahren geht über diesen Rahmen hinaus und erfasst alle Bitsysteme von zur Basis 1 bis zur Basis 16.

Die Codierung ist sehr einfach:

> Ein **Generator** erzeugt die erforderlichen Systemparameter
> und im **Codierbereich** wird der Chiffretext geschrieben.



Beide Bereiche werden kombiniert, können aber auch getrennt und eigenständig verwendet werden. Zur Erläuterung der Zusammenhänge muss allerdings auf die Grundzüge der digitalen Technik zurückgegangen werden.

1 Systematisierung der Bitfolgen

Um mit Bitfolgen systematisch zu arbeiten, müssen sie systematisiert, d.h. skaliert und in feste Abschnitte (Units) geteilt werden. Bisher sind Bitfolgen noch nicht skaliert. Wie in der Zahlentheorie lassen sich Bitfolgen auch in einem Stellenwertsystem ordnen.

System-Alphabet

Bitfolgen:	1-bit	=	Bitsystem zur Basis 1	=	2^1 Zeichen	= 2 Units
	2-bit	=	Bitsystem zur Basis 2	=	2^2 Zeichen	= 4 Units
	3-bit	=	Bitsystem zur Basis 3	=	2^3 Zeichen	= 8 Units
	4-bit	=	Bitsystem zur Basis 4	=	2^4 Zeichen	= 16 Units
	5-bit	=	Bitsystem zur Basis 5	=	2^5 Zeichen	= 32 Units
	6-bit	=	Bitsystem zur Basis 6	=	2^6 Zeichen	= 64 Units
	7-bit	=	Bitsystem zur Basis 7	=	2^7 Zeichen	= 128 Units
	8-bit	**=**	**Bitsystem zur Basis 8**	**=**	**2^8 Bytes**	**= 256 Bytes**
	9-bit	=	Bitsystem zur Basis 9	=	2^9 Zeichen	= 512 Units
	10-bit	=	Bitsystem zur Basis 10	=	2^10 Zeichen	= 1024 Units
	11-bit	=	Bitsystem zur Basis 11	=	2^11 Zeichen	= 2048 Units
	12-bit	=	Bitsystem zur Basis 12	=	2^12 Zeichen	= 4096 Units
	13-bit	=	Bitsystem zur Basis 13	=	2^13 Zeichen	= 8192 Units
	14 -bit	=	Bitsystem zur Basis 14	=	2^14 Zeichen	= 16384 Units
	15-bit	=	Bitsystem zur Basis 15	=	2^15 Zeichen	= 32768 Units
	16-bit	=	Bitsystem zur Basis 16	=	2^16 Zeichen	= 65536 Units
	32-bit	=	Bitsystem zur Basis 32	=	2^32 Zeichen	= 4294967296 Units

Ein Bitsystem besteht aus einer bestimmten Anzahl Zeichen, die im zugehörigen System-Alphabet zusammengefasst sind. Die zugrunde liegenden Zusammenhänge zeigt die folgende Übersicht:

Systembasis	System-alphabet	Bitfolgen in Einheiten Indizierung	Längen-verhältnis
Bitsystem zur Basis 1	2	0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0.	1:8
Bitsystem zur Basis 2	4	01 00 11 10 01 10 11 11 01 11 01 00 01 10 00 01 01 10 00. 1 0 3 2 1 2 3 3 1 3 1 0 1 2 0 1 1 2 0	1:4
Bitsystem zur basis 3	8	010 011 100 110 111 101 110 100 011 000 010 110 001 ... 2 3 4 6 7 5 6 4 3 0 2 6 1	1:2,66
Bitsystem zur Basis 4	16	0100 1110 0110 1111 0111 0100 0110 0001 0110 0010 ... 4 14 6 15 7 4 6 1 6 2	1:2
Bitsystem zur Basis 5	32	01001 11001 10111 10111 01000 11000 01011 00010 01.. 9 25 23 23 8 24 11 2	1:1,6
Bitsystem zur Basis 6	64	010011 100110 111101 110100 011000 010110 001001 ... 19 38 61 52 24 22 9	1:1,33
Bitsystem zur Basis 7	128	0100111 0011011 1101110 1000110 0001011 0001001 ... 39 27 110 70 11 9	1:1,143
Bitsystem zur Basis 8 *Bereiche*	256	01001110 01101111 01110100 01100001 01100010 011.. 78 111 116 97 98 4E 6F 74 61 62 N o t a b ... **Stromchiffre, Blockchiffre mit Längenkongruenz, Feistel-Netze ECB, CBC, CFB, OFB, DES, IDEA, AES, RSA, differenzielle und lineare Kryptanalyse, fast alle weiteren Programme**	1:1
Bitsystem zur Basis 9	512	010011100 110111101 110100011 000010110 001001100 156 445 419 22 76	1:1,79
Bitsystem zut Basis 10	1024	0100111001 1011110111 0100011000 0101100010 01100 313 759 280 354	1:1,6
Bitsystem zur Basis 11	2048	01001110011 01111011101 00011000010 11000100110 ... 627 989 194 1574	1:1,46
Bitsystem zur Basis 12	4096	010011100110 111101110100 011000010110 0010011001 1254 3956 1558 613	1:1,33
Bitsystem zur Basis xx	*div*	x x x x x	1: xxx

Im Längenverhältnis kommen die Längen von Klartext und Geheimtext zum Ausdruck.

2 System-Alphabet

Das System-Alphabet ist der wichtigste Bestandteil der Computertechnik. Es ist die Grundlage für die Visualisierung des Inhalts der Bitfolgen. Ohne die sachgerechte Definition eines Alphabets im jeweiligen Bitsystem könnte mit dem Computer gar nicht gearbeitet werden. Im Bitsystem zur Basis 8 ist das System-Alphabet der ASCII-Zeichensatz in seiner jeweiligen Ausprägung. Für jedes andere Bitsystem muss ein eigenes System-Alphabet zusätzlich definiert werden. Die System-Alphabete sind unabhängig voneinander.

3 Bit-Konversion

Bitkonversion ist die Umwandlung einer Bitfolge von einem Bitsystem in ein anderes Bitsystem. Dabei bleiben die Anzahl der Bits und ihre Reihenfolge gleich. Kein Bit wird hinzugefügt und kein Bit wird weggelassen. Nur die Anzahl der Bits in einer Einheit (Unit) ändert sich, und damit die Struktur der Bitfolge. Die dezimalen Werte der neuen Einheiten sind Indexwerte für das zugeordnete System-Alphabet.

Die Bitkonversion von Basis 1 nach Basis 7, 8 und 12 geschieht wie folgt:

Bitfolge Basis 1:
010011100110111101110100011000010110001001100101011011100110010 . . . . . .
Bitfolge Basis 7:
0100111 0011011 1101110 1000110 0001011 0001001 1001010 1101110 0110010 . . . . .
Index: 39 27 110 70 11 9 74 110 50
' ESC n F VT TAB J n 2

Bitfolge Basis 8:
01001110 01101111 01110100 01100001 01100010 01100101 01101110 01100101 . . . .
Index: 78 111 116 97 98 101 110 101
System-Alphabet Basis 8:
N o t a b e n e

Bitfolge Basis 12:
010011100110 111101110100 011000010110 001001100101 011011100110 0101 . . . . .
Index: 1254 3956 1558 613 1766
System-Alphabet Basis 12:
&ä }Æ Ó8 vâ ßä

Die ursprüngliche Bitfolge Basis 1 wird durch Bitkonversion in die entsprechenden Units (Abschnitte) der angestrebten Bitsysteme umgewandelt. Kein Bit wird hinzugefügt und kein Bit wird weggelassen. Die nicht strukturierte Bitfolge (Anzahl und Reihenfolge) bleibt gleich. Der Dezimalwert der entstehenden Bitsequenzen entspricht dem Index des Chiffrezeichens im zugeordneten System-Alphabet (Chiffre-Alphabet).

Bisher werden Umwandlungen von Bitfolgen im Verfahren „*Coding Base64*“ vorgenommen [#3]. Dabei werden Bytes im Bitsystem zur Basis 8 in eine Folge von 6-bit Sequenzen umgewandelt. Die dezimalen Werte dieser binären Sequenzen sind Indizes für ein Chiffre-Alphabet von 64 Zeichen. Das Chiffre-Alphabet (System-Alphabet) wird statisch vorgegeben.

4 Einheitliches Ordnungssystem

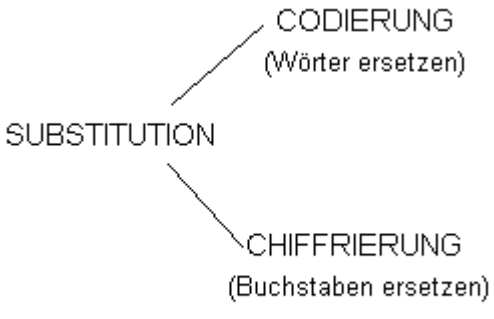
In der aktuellen Kryptographie werden **Eingaben** und **Ausgaben** (Klartexte, verschlüsselte Ergebnisse, Chiffretexte) in **Bytes** verarbeitet. Außerdem wird gefordert, Klartext und Chiffretext müssen gleich lang sein [#4,#5,#6]. Der Chiffretext soll an der gleichen Stelle wieder gespeichert werden, wo vorher der Klartext stand. Für jedes Klartextzeichen gibt es daher ein bestimmtes Geheimtextzeichen (Blendtexte ausgenommen). Sowohl Eingaben als auch Ausgaben werden im selben Chiffre-Alphabet dargestellt, fast ausschließlich im erweiterten ASCII-Zeichensatz. Alle Verschlüsselungsoperationen vollziehen sich somit im Bitsystem zur **Basis 8** und damit in einem einheitlichen **Ordnungsystem.**

Nur in einem einheitlichen Ordnungssystem können Wiederholungsmuster, Wortkombinationen, Häufigkeitsanalysen und Bigramme im Chiffretext zu vergleichbaren Merkmalen im Klartext führen [#7]. Die bekanntesten Angriffe, wie Strukturanalyse, „known plaintext attack“, „chosen plaintext attack“ und auch „differenzielle“ und „lineare“ Kryptoanalyse u.a. können nur dann Erfolg haben, wenn für jedes Chiffrezeichen auch ein bestimmtes Klartextzeichen vorhanden ist, also ein einheitliches Ordnungssystem vorliegt.

Bisher angewendete Techniken (Feistel-Netzwerke, Intermix, LFSR, S-Boxen, Betriebsarten EBC, CBC, CFB und OFB, Konfusion und Diffusion u.a.) setzen für die Beziehung von Klartext zum Chiffretext ein Verhältnis von **1:1** voraus, d.h. Arbeiten im Bitsystem zur Basis 8.
Im CypherMatrix Verfahren ist das nicht der Fall. Es arbeitet in allen Bitsystemen von zur Basis 1 bis zur Basis 16 (ausgenommen: Basis 8).

5 „Code-Verschlüsselung“

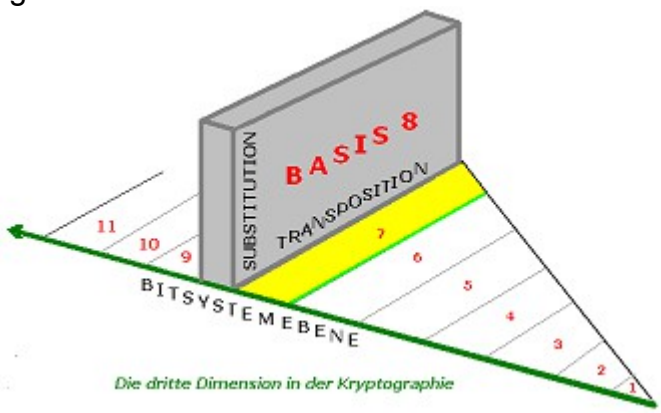
Mit der Zeit haben sich in der Kryptographie zwei Methoden entwickelt: „Substitution“ und „Transposition“. Als die Computer kamen, treten zwar die Bits an die Stelle der Buchstaben, aber im Ergebnis werden die Grundsätze der „Buchstabenverschlüsselung“ beibehalten. „Verschlüsselt wird nach wie vor gemäß den Grundsätzen der Substitution und Transposition“ [#8].



Simon Singh, Geheime Botschaften, S.48

Die Buchstabenverschlüsselung kann nunmehr durch eine „Code-Verschlüsselung“ ergänzt werden. Das bisher benutzte Bitsystem zur Basis 8 muss nur in ein anderes Bitsystem (Basis 1 bis Basis 16, ausgenommen Basis 8) umgewandelt werden. Die Bits repräsentieren dann keine Buchstaben, sondern eine digitale Information im betreffenden Bitsystem als Substitution für Wortsegmente. Vor allem können Chiffrezeichen nicht mehr durch Häufigkeitsanalysen und andere Angriffstechniken analysiert werden, weil eine Verbindung zu Buchstaben nicht mehr besteht,

Durch diese Neuorientierung können bestimmte kryptographische Vorgänge als ein eigener Bereich mit der Bezeichnung „**Codegraphie** als Teil der Kryptographie“ zusammengefasst werden. Damit bilden sie einen besonderen Teil der Kryptographie. Die bisherigen Bereiche „Substitution“ und „Transposition“ werden durch die „Bitsystemebene“ als dritte Dimension ergänzt.



Die dritte Dimension in der Kryptographie

Mit der „dritten Dimension“ lassen sich eine Reihe weiterer Codierungen gestalten.
Im Anhang A sind dazu einige vom Autor entwickelte Programme aufgeführt. Für jedes einzelne Programm kann der Quellcode per e-mail beim Autor angefordert werden (eschnoor@multi-matrix.de). Die Programme sind noch in WindowsXP geschrieben, sie müssten nach **C#** umgeschrieben werden [#9].

Nach meinen Erkenntnissen erstreckt sich der wissenschaftliche Bereich der Kryptografie offenbar nur auf das Feld von „Substitution / Transposition / Basis 8“. Alles darüber hinaus gehende wird erfahrungsgemäß als „Schlangen Öl“ ausgegrenzt.

6 Das CypherMatrix Verfahren

Das CypherMatrix Verfahren begründet einen eigenen Bereich. Es werden nur die folgenden Techniken angewendet: Startsequenz mit optimal 42 Zeichen, laufende Erzeugung von Blockschlüssel, Matrixschlüssel und unabhängige System-Alphabete in jedem Durchlauf sowie System-Wechsel durch Bitkonversion. Alle weiteren in aktuellen Verschlüsselungen angewendeten Schritte zur Abwehr von Angriffen sind nicht erforderlich. Das einheitliche Ordnungssystem zur Basis 8 fällt fort und Klartext und Chiffretext können nicht mehr sinnvoll verglichen werden. Damit entfällt auch die Basis für alle bekannten Angriffszenarien und wir können sie vergessen.

6.1 Der Generator

Die Funktion beginnt mit der Eingabe einer **Start Sequenz** und führt zu einer eindeutigen und kollisionsfreien Abbildung in Form einer **CypherMatrix** (16x16 Zeichen: GF(16^2)).
Mit einer “S-Box” ist die CypherMatrix nicht vergleichbar. Die CypherMatrix liefert die für kryptographische Lösungen erforderlichen Steuerungsparameter (Chiffre-Alphabet, Blockschlüssel, Matrixschlüssel als Eingangssequenz für den nächsten Durchlauf und weitere Faktoren). Nach den Gesetzen der Wahrscheinlichkeit entsteht eine Wiederholung der gleichen CypherMatrix erst in **256!** (Fakultät) = **8E+506** Fällen.

6.1.1. Verlauf der Funktion

Das Verfahren ist symmetrisch, weil Sender und Empfänger zur Initialisierung des Generators die gleiche Startsequenz eingeben müssen und es ist polyalphabetisch, weil der Generator für jeden Klartextblock in jedem Durchlauf neue Steuerungsparameter erzeugt. Eine Runde ist der Durchlauf des Generators von der Eingabe der Startsequenz (bzw. Matrixschlüssel) bis zur Generierung der CypherMatrix.

Der Generator wird mit der Startsequenz gestartet und ab der zweiten Runde mit dem Matrixschlüssel fortgeführt. So entsteht eine unbegrenzte Anzahl von Runden, bis ein Endeimpuls gesetzt wird. Eine beliebige Startsequenz (Passphrase) mit mindestens 36 Zeichen (optimal 42) steuert das gesamte Verfahren. Einige Beispiele:

Ein Fliegenpilz steigt in die Stratosphäre	[42 Bytes]
Die weiße Elster fließt in das schwarze Meer	[44 Bytes]
7 kangaroos jumping along the Times Square	[42 Bytes]
Yellow submarines diving in the Murray Mouth	[44 Bytes]

Die Startsequenz sollte ungewöhnlich sein und dennoch leicht zu behalten, so dass sie nicht aufgeschrieben werden muss aber auch nicht geraten werden kann. Wegen ihrer Länge kann sie weder durch Iteration noch durch Wörterbuchangriffe analysiert werden. Ein Angreifer kann auch nicht mit Erfolg versuchen, Teile des Schlüssels getrennt oder nacheinander zu brechen, da die Startsequenz nur in einem Durchgang als Ganzes gefunden werden kann, wenn überhaupt.

Folgende Arbeitsparameter steuern die Funktion:

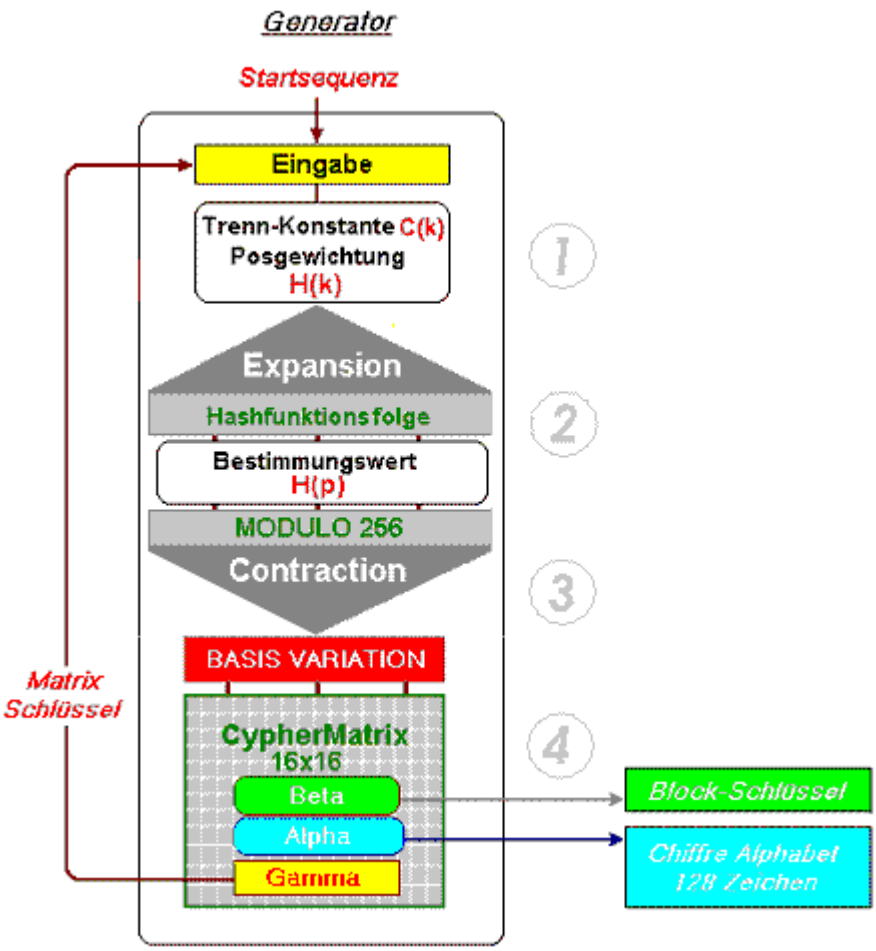
a) Persönliche Eingabe:

   1) Startsequenz (36-64, optimal 42),
   2) Anwendercode (1 bis 999, vorbelegt mit 1),

b) Fest verdrahted:

   1) Länge Matrixschlüssel (36-64, vorbelegt mit 42),
   2) Blocklänge (7-63, vorbelegt nach Erfordernis),
   3) Expansionsfaktor (36-128, vorbelegt mit 77).

Die Startsequenz kommt in ihrer Funktion dem “Spruchschlüssel” bei der **Enigma** gleich und die Arbeitsparameter entsprechen in ihrer Wirkung dem “Tagesschlüssel” [#15].
Mit jedem Durchlauf eines Klartextblocks entsteht ein neues Chiffre-Alphabet.
Bei der Enigma entsteht mit dem Eintippen eines Buchstabens und dem Fortschreiten der aktiven Schlüsselwalze ein neues Geheimtextalphabet [#16]. Der grundsätzliche Unterschied zwischen beiden Verschlüsselungen besteht jedoch darin, dass bei der Enigma sowohl im Klartext als auch im Geheimtext mit Buchstaben im Bitsystem zur Basis 8 gearbeitet wird, während im CypherMatrix Verfahren ein Systemwechsel von Basis 8 nach Basis 7 vorgenommen wird.



In jedem Durchlauf erzeugt der Generator die zur Verschlüsselung erforderlichen Steuerungsparameter:

1. Das Chiffre-Alphabet für die aktuelle Runde,
2. den Blockschlüssel für die XOR-Verknüpfung und
3. einen Matrixschlüssel als Startsequenz für die nächste Runde.

Den Abschluss jeder Runde bildet die **CypherMatrix** mit 16x16 Elementen, die alle Steuerungsparameter für die Verschlüsselung zur Verfügung stellt. Der Matrixschlüssel (42 Zeichen) wird auf den Anfang der Funktion zurückgeführt (loop). Er initialisiert den nächsten Durchgang. So entsteht eine unbegrenzte Anzahl von Runden, bis ein Endeimpuls gesetzt wird. Nach der Wahrscheinlichkeit entsteht die Wiederholung einer gleichen CypherMatrix erst in **256!** (Fakultät) = **8E+506** Fällen.

6.1.2 Eingabe der Startsequenz

Als Beispiel wird die folgende Startsequenz eingegeben:

Der schwarze Kater fängt immer graue Mäuse (n = 42).

Es gilt eine eindeutige Abbildung der Eingabe als Bestimmungsbasis für CypherMatrix und Steuerungsparameter zu finden.

Die Eingabe **m** ist eine Folge bestimmter Bytes **a**(**i**) mit der Länge **n**. Um die Folge als Sachverhalt zu analysieren, muss sie systematisiert (skaliert) werden. Dazu wird jedem Byte **a**(**i**) ein Index zugeordnet und alle **n** Bytes werden in sachgerechter Weise miteinander verknüpft (Addition).

$$\mathbf{m} = (a_1+1) + (a_2+1) + (a_3+1) + \ldots (a_i+1) + \ldots (a_n+1)$$

(Der einzelne Wert für "a_i" wird um (+1) erhöht da sonst ASCII-null (0) nicht berücksichtigt wird)

Um die einzelnen Bytes **a**(**i**) innerhalb der Zeichenfolge zu unterscheiden, müssen weitere Merkmale hinzukommen, da anderenfalls keine eindeutigen Ergebnisse erzielt werden können.

6.1.3 Erweiterung zur Positionsgewichtung

Jeder Sachverhalt, soweit er in seinen Dimensionen skalierbar ist, kann durch seine Koordinaten für **Gegenstand, Ort** und **Zeit** eindeutig bestimmt werden. Die Skalierung erfasst den Gegenstand, den Ort und die Zeit der digitalen Zeichen. Wir definieren:

Sachverhalt = **m** digitale Zeichenfolge der Länge **n**
Gegenstand = **a**(**i**) Element der Folge, Zeichen, Byte
Ort = **p**(**i**) Position von **a**(**i**) innerhalb der Folge
Zeit = **t** (**i**) Zeitpunkt von **a**(**i**) innerhalb der Folge

Damit sich die einzelnen Zeichen unterscheiden wird jedes Byte **a**(**i**) mit seinem Ort **p**(**i**) multipliziert, d.h. **positionsgewichtet**. Die Zeit ist nur von Bedeutung, wenn zwischen den einzelnen Bytes und der Prozessorfrequenz eine variable Funktion besteht, ansonsten:

t (**i**) = **1**. Um einen Wert für die Folge **m** zu erhalten, werden die Dimensionswerte für Gegenstand, Ort und Zeit durch Multiplikation verknüpft und zum Zwischenwert **H**(**k**) addiert.

$$\mathbf{H(k)} = \sum_{i=1}^{n} (a_i + 1) * p_i * t_i \qquad t_i = 1$$

H(k) = 88446

Mit der Positionsgewichtung unterscheiden sich zwar die Bytes **a**(**i**), aber Kollisionen als Folge des Austausches von Bytes innerhalb der Zeichenfolge sind noch nicht ausgeschlossen.

6.1.4 Ausschluss von Kollisionen

Eine Kollision entsteht unter folgenden Bedingungen:

Kollision: $\mathbf{H(k)}\ a_i = \mathbf{H(k)}\ b_i$

$$(a_1+1) * p_1 + (a_2+1) * p_2 = (b_1+1) * p_1 + (b_2+1) * p_2$$

Um Kollisionen zu vermeiden, wird die Positionsgewichtung in einen Bereich oberhalb der Länge (n) der Zeichenfolge verschoben, indem p(i) um einen konstanten Abstand **C** erweitert wird.

Die Ableitung des Abstands **C** wird im Einzelnen in telecypher.net/Kollfrei.pdf ausführlich dargelegt.

Der Faktor **C** ist allein von der Länge **n** der Eingabesequenz abhängig. Er hat außerdem die Eigenschaft, für gleiche Längen der Eingabesequenz gleich zu sein und die Zeichen der Positionsgewichtung in kollisionsfreie und kollisionsbelastete Abschnitte zu trennen. Der Faktor **C** erhält daher die Bezeichnung: Trenn-Konstante **C**(**t**).

Um die Funktion zu individualisieren wird zusätzlich ein Code – eine gewählte Zahl zwischen 1 und 99 – eingeführt. Wir setzen Code = 1.

$$\mathbf{C(t)} = \mathbf{n} * (\mathbf{n} - \mathbf{2}) + \mathbf{code}$$
$$\mathbf{C(t)} = 1681$$

Mit Runde wird jeder Durchlauf der Funktion gezählt. Nach Einbindung der Trenn-Konstante **C**(**t**) wird dann **H**(**k**) - als Zwischenwert - wie folgt ermittelt:

$$\mathbf{H(k)} = \sum_{i=1}^{n} (\mathbf{a}_i + \mathbf{1}) * (\mathbf{p}_i + \mathbf{C(t)} + \mathbf{Runde})$$
$$\mathbf{H(k)} = 6927458$$

Damit vermeidet das Ergebnis **H**(**k**) Kollisionen, ist aber immer noch zu niedrig, um unangreifbare Bestimmungswerte für die Funktion zu begründen. Es könnte lediglich als **MAC** für Nachrichten dienen.

6.1.5 Erweiterung zur Hashfunktionsfolge

Zur Erweiterung der Bestimmungsbasis wird eine **Hashfunktionsfolge** (**HF**) eingeführt, die die Eingangssequenz zu einer umfangreichen Folge in einem höherwertigen Zahlensystem expandiert. Das Zahlensystem der Expansion ist wählbar zwischen 64 bis 96. Hier wird die **Basis 77** festgelegt. Es kann auch jede andere Zahl im definierten Bereich gewählt werden. Für jedes Zeichen der Eingabeseqzenz wird der dezimalen Wert ($\mathbf{s}_i$) errechnet, der dann zu (**di**) **-** Ziffern im Zahlensystem zur Basis 77 - umgewandelt wird. Gleichzeitig ermittelt das Verfahren die Summe aller Einzelergebnisse ($\mathbf{s}_i$) als zusätzlichen Wert **H**(**p**) zur Bestimmung verschiedener Steuerungsparameter und akkumuliert die Ergebnisse ($\mathbf{d}_i$) seriell zur Hashfunktionsfolge (**HF**).

$$\mathbf{s}_i = (\mathbf{a}_i + \mathbf{1}) * \mathbf{p}_i * \mathbf{H(k)} + \mathbf{p}_i + \mathbf{code} + \mathbf{Runde}$$
$$\mathbf{s}_i \rightarrow \mathbf{d}_i \quad \text{(base 77)}$$

$$\mathbf{HF} = \mathbf{d}_1 + \mathbf{d}_2 + \mathbf{d}_3 + \ldots + \mathbf{d}_i + \ldots \mathbf{d}_m$$

(m = Anzahl der Zahlen im System zur Basis 77)

$$\mathbf{H(p)} = \sum_{i=1}^{n} \mathbf{S}_i$$
$$\mathbf{H(p)} = 612705951255$$

Das gewählte Zahlensystem zur Basis 77 umfasst die folgenden Ziffern:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz&#@àáâãäåæçèéêë
(definiert vom Autor, nicht standardisiert)

Die Hashfunktionsfolge **HF** umfasst 248 Ziffern im Zahlensystem zur Basis 77:

Dk0xBeFd7Vâë3@jQ0ê9S1bMèVh1fIXZO1â@æyi2ZE7Pp2J&dçr2çmCzH3Zmcbk3AG3iM17
fpnm2tqFEU3wqáiA4yä@Qæ4Xs&&u5MæN2T1khAPA5Kèh0f7BVAë26JI3uk69TQlK7ERw1â
28ik0n748s7i7kLèG&7âé7bU7hâáTë8#âcTl2lk4rB8d@KMH9ságHD8elèäFAhâá2à9UmA
t&39leSo7j7ä&2DLD@VDC6BgaJCDI&èzAéHp0ç

Bei der Generierung der Hashfunktionsfolge ergibt sich beispielsweise für die Teilsequenz "**Kater**" an den Positionen 14 bis 18 der Eingabesequenz folgende Berechnung:

char	pi		Hk	(ai+1)*pi*Hk		Si	Basis 77
	(ai+1)	(ai+1)*pi			pi+code+r		
..	...	..	.....	.....	.....	..	.....	.....
K	76	14	1064	6927458	7370815312	16	7370815328	2tqFEU
a	98	15	1470	6927458	10183363260	17	10183363277	3wqáiA
t	117	16	1872	6927458	12968201376	18	12968201394	4yä@Qæ
e	102	17	1734	6927458	12012212172	19	12012212191	4Xs&&u
r	115	18	2070	6927458	14339838060	20	14339838080	5MæN2T
..	...	..	.....	.....	.....	..	.....	.....

Die Variablen sind Ziffern (keine Zeichen) im Zahlensystem zur Basis 77. Es gibt keinen Weg zurück zur Startsequenz (erste Einweg-Funktion). Gleichzeitig errechnet das Programm die folgenden Bestimmungsfaktoren:

Trenn-Konstante C(t):	1681
Positionsgewichteter Wert (H(k)):	6927458
Bestimmungswert (H(p)):	612705951255
Gesamtwert (H(p)+H(k)):	612712878713

Aus den Bestimmungsfaktoren werden folgende Steuerungsparameter abgeleitet:

Variante	$(H_k \text{ MOD } 11) +1$	=	11	Beginn der Kontraktion
Alpha	$((H_k + H_p) \text{ MOD } 255) +1$	=	204	Offset Chiffre-Alphabet
Beta	$(H_k \text{ MOD } 169) +1$	=	149	Offset Blockschlüssel
Gamma	$((H_p + \text{code}) \text{ MOD } 196) +1$	=	141	Offset Matrixschlüssel
Delta	$((H_k + H_p) \text{ MOD } 155) +\text{code}$	=	44	dynamische Bitfolgen
Theta	$(H_k \text{ MOD } 32) +1$	=	3	Offset Rückrechnung
Omega	$(H_k \text{ MOD } 95) +1$	=	59	Beginn Doppelzeichen

Die Steuerungsparameter dienen zur Lösung verschiedener kryptographischer Aufgaben.

6.1.6 Verdichtung zur BASIS VARIATION

Um die Bestimmungsbasis auf dezimale Größen zurückzuführen wird eine **Kontraktionsfunktion** eingeführt. Für die Ziffern der Hashfunktionsfolge wird das Zahlensystem zur **Basis 78** (Expansions-basis +1) unterstellt. Jeweils drei Ziffern der Hashfunktionsfolge werden seriell durch **MODULO 256** in dezimale Zahlen 0 bis 255 (ohne Wiederholung) zurückgerechnet. Der Parameter **Theta** wird abgezogen.

Die Ergebnisse werden in der BASIS VARIATION gespeichert, einem Array von 16x16 Elementen. Eine rückwärts gerichtete Bestimmung vorhergehender Daten ist nicht möglich (zweite Einweg-Funktion).

Die ersten vier Rückrechnungen ab **Variante = 11** zeigen sich wie folgt:

3 Ziffern Basis 78	dezimal	Modulo 256	- Theta	Element
âë3	413559	119	3	116
ë3@	462682	90	3	87
3@j	23289	249	3	246
@jQ	392912	208	3	205
...	.....	...	..	...

BASIS-VARIATION (256 Elemente)
Verteilung der Elemente

116 087 246 205 093 048 224 067 106 225 029 078 050 126 212 096
081 254 247 003 026 238 107 016 219 206 088 108 061 020 240 042
173 184 141 148 159 255 069 117 165 201 162 213 226 014 035 143
118 137 089 041 101 136 083 199 098 043 012 010 028 039 036 144
123 000 109 220 221 025 044 127 021 248 189 027 154 051 110 049
139 178 082 243 119 145 138 121 204 142 094 084 092 174 241 090
015 077 052 063 053 182 251 018 130 146 210 229 017 207 208 111
113 008 231 135 232 019 112 209 076 075 024 125 185 045 086 249
046 237 033 179 177 102 242 167 153 180 152 163 200 253 114 228
095 001 040 218 013 222 236 190 124 066 186 128 211 168 097 223
147 202 250 030 133 196 005 002 164 062 244 004 064 115 047 022
023 245 151 155 099 100 006 140 129 056 131 252 120 176 007 149
103 104 105 157 054 160 009 166 011 122 055 091 070 132 181 150
031 216 158 183 032 156 134 071 161 169 187 170 034 037 171 057
172 065 058 175 038 059 188 060 068 072 191 085 239 073 192 193
074 227 217 230 079 080 194 195 197 198 203 214 215 233 234 235

6.1.7 Berechnung der „CypherMatrix“

Für die Berechnung der CypherMatrix werden die Elemente der BASIS VARIATION in ihrer Verteilung 16x16 Zeichen direkt als Bestimmungsbasis verwendet. Dabei werden die Werte direkt auf die Indexwerte der Bytes von **0** bis **255** (vergleichbar: ASCII-Zeichensatz) bezogen. Für die Verteilung der Zeichen (16x16) in der CypherMatrix verwendet das Programm die folgende Ermittlung:

Die Indexwerte (**i, j**) werden in einem gesonderten Array (IndexFolge (2,16)) aus den Elementen der BASIS-VARIATION (VarFolge(), generiert.
Auszug aus dem Sourcecode:

```
SUB DynaFolge

  LOCAL DynA(),Z
  SHARED IndexFolge(),VarFolge(),Delta,Omega

  DIM DYNAMIC DynA(16)

  FOR B=1 TO 2
      IF B=1 THEN
         Z=Delta
      ELSEIF B=2 THEN
        Z=Omega
      END IF

      FOR C=1 TO 16
        INCR Z
        IF Z>256 THEN Z=1
        A=(VarFolge(Z) MOD 16)
        DynA(C)=A
        IF C>1 THEN
          L=0
          DO
             INCR L
             IF DynA(L)=A THEN
                INCR A
                A=(A MOD 16)
                DynA(C)=A
                L=0
             END IF
          LOOP UNTIL L=C-1
        END IF
        IndexFolge(B,C)=DynA(C)+1
    NEXT C
  NEXT B

  ERASE DynA
  END SUB

 N = 0
 FOR i = 1 TO 16
    FOR j = 1 TO 16
      INCR N
      x = IndexFolge(1,i)
      y = IndexFolge(2,j)
      Matrix$(x,y)= Variation$(N)          CypherMatrix Array
    NEXT j
  NEXT i
```

Die Berechnung ergibt folgende Indizes:

```
Index-Folge 1:  3  15  4  16  7  10  11  12  6  9  5  8  13  14  1  2
Index-Folge 2:  11  13  8  5  1  12  2  14  15  16  10  3  4  6  9  7
```

CypherMatrix (Variation: D)

1	16	F5	06	8C	2F	81	83	73	38	64	04	17	40	97	9B	63	*16*
17	95	68	09	A6	07	0B	37	B0	7A	A0	FC	67	78	69	9D	36	*32*
33	96	D8	86	47	B5	A1	BB	84	A9	9C	5B	1F	46	9E	B7	20	*48*
49	C1	E3	C2	C3	C0	C5	CB	49	C6	50	55	4A	EF	D9	E6	4F	*64*
65	6F	08	70	D1	D0	4C	18	CF	4B	13	E5	71	11	E7	87	E8	*80*
81	31	B2	8A	79	6E	CC	5E	33	8E	91	1B	8B	9A	52	F3	77	*96*
97	60	FE	6B	10	D4	DB	58	7E	CE	EE	4E	51	32	F7	03	1A	*112*
113	F9	ED	F2	A7	56	99	98	2D	B4	66	7D	2E	B9	21	B3	B1	*128*
129	5A	4D	FB	12	F1	82	D2	AE	92	B6	54	0F	5C	34	3F	35	*144*
145	2A	B8	45	75	F0	A5	A2	14	C9	FF	6C	AD	3D	8D	94	9F	*160*
161	8F	89	53	C7	23	62	0C	0E	2B	88	D5	76	E2	59	29	65	*176*
177	90	00	2C	7F	24	15	BD	27	F8	19	0A	7B	1C	6D	DC	DD	*192*
193	E4	01	EC	BE	72	7C	BA	FD	42	DE	A3	5F	C8	28	DA	0D	*208*
209	DF	CA	05	02	61	A4	F4	A8	3E	C4	80	93	D3	FA	1E	85	*224*
225	39	41	BC	3C	AB	44	BF	25	48	3B	AA	AC	22	3A	AF	26	*240*
241	EB	57	E0	43	EA	6A	1D	E9	E1	30	D6	74	D7	F6	CD	5D	*256*

CypherMatrix (Runde 1)

1	■	§	♠	î	/	ü	â	s	8	d	♦	↨	@	ù	ø	c	*16*
17	ò	h	o	ª	●	♂	7	▒	z	á	³	g	x	i	Ø	6	*32*
33	û	Ï	å	G	Á	í	╗	ä	®	£	[	▼	F	×	À	□	*48*
49	┴	Ò	┬	├	└	┼	╦	I	ã	P	U	J	´	┘	µ	O	*64*
65	o	◘	p	Đ	ð	L	↑	¤	K	‼	Õ	q	◄	þ	ç	Þ	*80*
81	1	▓	è	y	n	╠	^	3	Ä	æ	←	ï	Ü	R	¾	w	*96*
97	`	■	k	▶	È	▉	X	~	╬	¯	N	Q	2	¸	♥	→	*112*
113	¨	Ý	═	°	V	Ö	ÿ	-	┤	f	}	.	╣	!	▕	▒	*128*
129	Z	M	¹	↕	±	é	Ê	«	Æ	Â	T	☼	\	4	?	5	*144*
145	*	©	E	u		Ñ	ó	¶	╔	▯	l	¡	=	ì	ö	ƒ	*160*
161	Å	ë	S	Ã	#	b	♀	♬	+	ê	€	v	Ô	Y	)	e	*176*
177	É	□	,	⌂	$	§	¢	'	°	↓	◙	{	∟	m	■	¦	*192*
193	õ	☺	ý	¥	r	|	╟	²	B	Ì	ú	_	╚	(	┌	♪	*208*
209	■	╩	♣	☻	a	ñ	▕	¿	>	─	Ç	ô	Ë	·	▲	à	*224*
225	9	A	╝	<	½	D	┐	%	H	;	¬	¼	"	:	»	&	*240*
241	Ù	W	Ó	C	Û	j	↔	Ú	ß	0	Í	t	Î	÷	═	]	*256*

6.1.8 Steuerungsparameter

Ab Position Alpha = **204** werden 128 Zeichen als Chiffre-Alphabet des Bitsystems zur Basis 7 entnommen. Bestimmte Zeichen (Hex: 00 bis 20, 22, 2C, B0, B1, B2, D5, DB, DC, DD, DE, DF und weitere) werden ausgeklammert, weil sie in einigen Situationen noch ihre ursprünglichen Aufgaben wahrnehmen (zB. **1A** =ASCII26 = →) und die ordnungsgemäße Durchführung des Programms stören.

Entnahmen aus der Matrix

```
.. F5 .. 8C 2F 81 83 73 38 64 .. .. 40 97 9B 63
95 68 .. A6 .. .. 37 .. 7A A0 FC 67 78 69 9D 36
96 D8 86 47 B5 A1 BB 84 A9 9C 5B .. 46 9E B7 ..
C1 E3 C2 C3 C0 C5 CB 49 C6 50 55 4A EF D9 E6 4F
6F .. 70 D1 D0 4C .. CF 4B .. E5 71 .. E7 87 E8
31 .. 8A 79 6E CC 5E 33 8E 91 .. 8B 9A 52 F3 77
60 FE .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. 5F C8 28 DA ..
.. CA .. .. 61 A4 F4 A8 3E C4 80 93 D3 FA .. 85
39 41 BC 3C AB 44 BF 25 48 3B AA AC .. 3A AF 26
EB 57 E0 43 EA 6A .. E9 E1 30 D6 74 D7 F6 CD 5D
```

Chiffre-Alphabet (hex)

```
5F C8 28 DA CA 61 A4 F4 A8 3E C4 80 93 D3 FA 85
39 41 BC 3C AB 44 BF 25 48 3B AA AC 3A AF 26 EB
57 E0 43 EA 6A E9 E1 30 D6 74 D7 F6 CD 5D F5 8C
2F 81 83 73 38 64 40 97 9B 63 95 68 A6 37 7A A0
FC 67 78 69 9D 36 96 D8 86 47 B5 A1 BB 84 A9 9C
5B 46 9E B7 C1 E3 C2 C3 C0 C5 CB 49 C6 50 55 4A
EF D9 E6 4F 6F 70 D1 D0 4C CF 4B E5 71 E7 87 E8
31 8A 79 6E CC 5E 33 8E 91 8B 9A 52 F3 77 60 FE
```

Chiffre-Alphabet (Basis 7)

```
_ ╚ ( ┌ ╩ a ñ ⌠ ¿ > ─ Ç ô Ë · à
9 A ╝ < ½ D ┐ % H ; ¬ ¼ : » & Ù
W Ó C Û j Ú ß 0 Í t Î ÷ ═ ] § î
/ ü â s 8 d @ ù ø c ò h ª 7 z á
³ g x i Ø 6 û Ï å G Á í ╗ ä ® £
[ F × À ┴ Ò ┬ ├ └ ┼ ╦ I ã P U J
´ ┘ µ O o p Đ ð L ¤ K Õ q þ ç Þ
1 è y n ╠ ^ 3 Ä æ ï Ü R ¾ w ` ■
```

Blockschlüssel

Der ab Position **149** der CypherMatrix entnommene Blockschlüssel umfasst 42 Zeichen:

```
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. F0 A5 A2 14 C9 FF 6C AD 3D 8D 94 9F
8F 89 53 C7 23 62 0C 0E 2B 88 D5 76 E2 59 29 65
90 00 2C 7F 24 15 BD 27 F8 19 0A 7B 1C 6D .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
```

```
  Ñ ó ¶ ╓ □ l ¡ = ì ö ƒ Å ë S Ã # b ♀ ♬ +
ê € v Ô Y ) e É □ , ⌂ $ § ¢ ' ° ↓ ◙ { ∟ m
```

Matrixschlüssel

Als Startsequenz für die nächste Runde entnimmt das Verfahren ab Position **141** einen neuen Matrixschlüssel mit 42 Zeichen:

```
.. .. .. .. .. .. .. .. .. .. .. .. 5C 34 3F 35
2A B8 45 75 F0 A5 A2 14 C9 FF 6C AD 3D 8D 94 9F
8F 89 53 C7 23 62 0C 0E 2B 88 D5 76 E2 59 29 65
90 00 2C 7F 24 15 .. .. .. .. .. .. .. .. .. ..
```

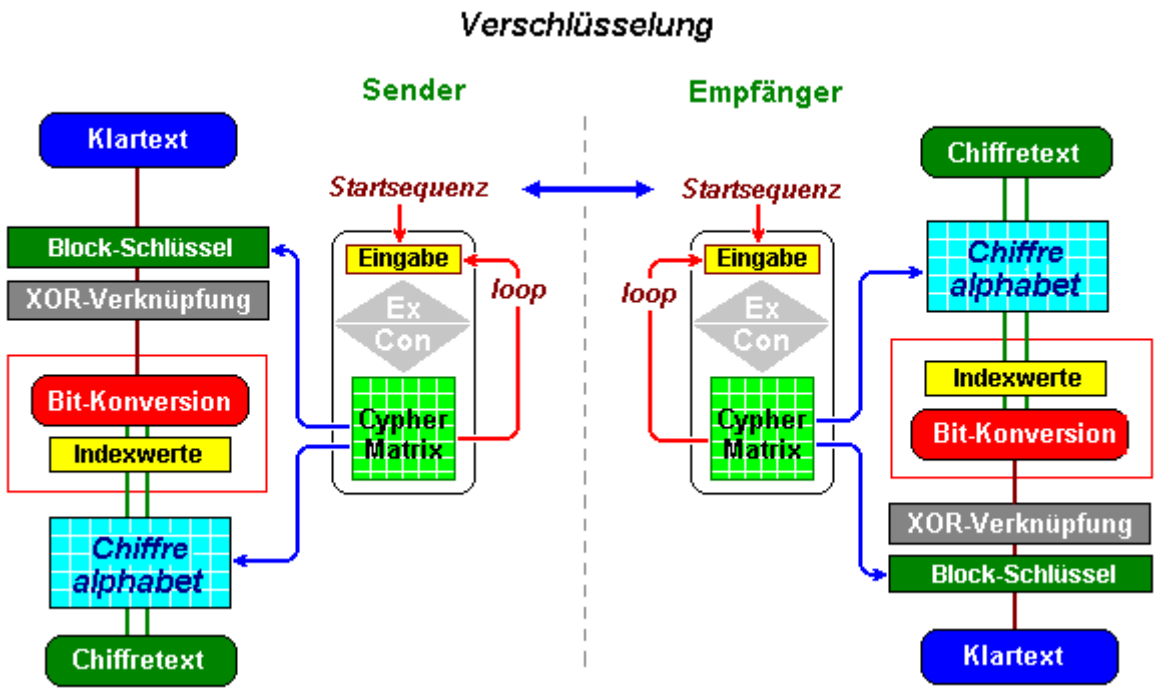
```
 \ 4 ? 5 * © E u   Ñ ó ¶ ╔ ▯ l ¡ = ì ö ƒ Å
ë S Ã # b ♀ ♬ + ê € v Ô Y ) e É □ , ⌂ $ Ş
```

Der Matrixschlüssel wird auf den Anfang der Funktion zurückgeführt (**loop**). Die jeweiligen Matrixschlüssel steuern den gesamten Ablauf des Verfahrens, inhaltsgleich, sowohl beim Sender als auch beim Empfänger.

6.2 Der Codierbereich

Die Verschlüsselung – das Schreiben und Lesen von geheimen Informationen – findet ausschließlich im Codierbereich statt. Mit Eingabe der gleichen Startsequenz, sowohl beim Sender als auch beim Empfänger, werden im gesamten Verfahren ein identischer Verlauf und identische Steuerungsparameter erzeugt.

Das folgende Schema zeigt die Zusammenhänge:



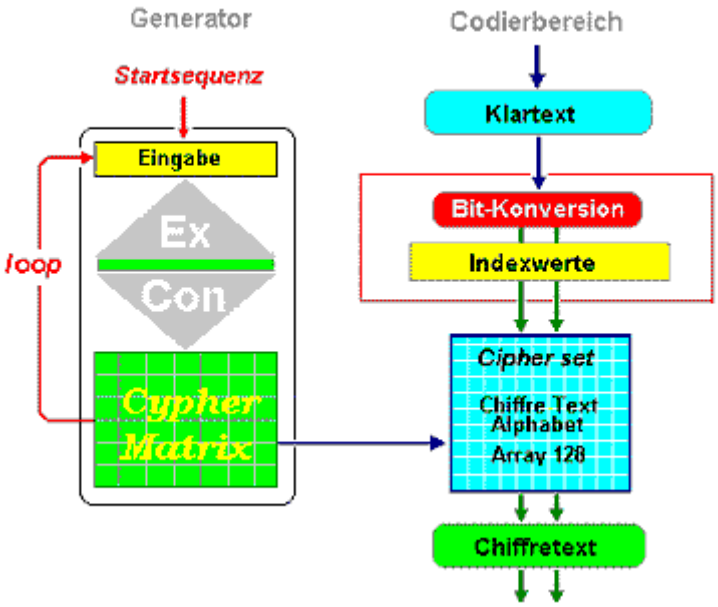
Die Verschlüsselung wird in folgenden Alternativen durchgeführt, und zwar:

1. **Basis-Coding:** Bit-Konversion allein ohne weitere Operationen oder
2. **Verbund-Coding:** Bit-Konversion mit zusätzlichen Operationen,

   a) mit XOR-Verknüpfung (voran- oder nachgestellt),
   b) verbunden mit weiteren Operationen (dyn24, exchange).

6.2.1 „Basis-Coding“

Im „Basis-Coding“ erfolgt die Bit-Konversion direkt vom Klartext im Bitsystem zur **Basis 8** zum Chiffretext im definierten Bitsystem von Basis 2 bis Basis 14 (ausgenommen Basis 8)**.** Aus der Bitfolge des eingelesenen Klartext-Blocks von 42 x 8 Bits (336 Bits) entsteht z.B. im Bitsystem zur Basis 7 eine Bitfolge von 48 x 7 Bits (336 Bits). Die Units im Bitsystem zur Basis 7 stellen die Indizes für die Chiffrezeichen im zugehörigen System-Alphabet dar.
Es ist die einfachste Art digitaler Verschlüsselung. Zusätzlich ist nur der Generator erforderlich. Die Indizes holen die Chiffrezeichen aus dem jeweiligen System-Alphabet, die dann zum Chiffretext verbunden werden. Die Zusammenhänge stellen sich schematisch wie folgt dar:



Als Beispiel wird das Programm **System11.**exe (vgl.Anhang A) genommen. Das Programm arbeitet im Bitsystem zur Basis 11. Die Verschlüsselung vollzieht sich in zwei Funktionen:

1. Bit-Konversion
   8-bit Klartextwerte → 11 bit Indexwerte (0 ...2048)
2. Bestimmung des Chiffretexts
   11-bit Indexwerte → Chiffre-Alphabet (0...2048) → Chiffretext.

Für das System-Alphabet zur **Basis 11** sind 2^11 Units = 2048 Zeichen erforderlich. Da für diesen Umfang keine Einzelzeichen zur Verfügung stehen, werden die Ziffern des **Zahlensystems** zur **Basis 128** – das sind 16384 Ziffern – als **Doppelzeichen** verwendet. Das System-Alphabet umfasst einen permutierten Abschnitt ab Kappa+1, d.h. von der Ziffer 14196+1 bis 16245 = 2048 Doppelzeichen.

Im Quellcode:

```
SUB Alphabet
      SHARED Alphabet$(),  Kappa
       Kappa = 14196,  ,  ,                                  (in jeder Runde neu berechnet)

       FOR C=1 TO 2048                                (Array für Zeichen des System-Alphabets)
          X# = C + Kappa                              (Index)
          CALL DezNachSystem (128, X#, Zeichen$)
          Digit$ = „00“+Zeichen$
          Digit$ = RIGHT$(Digit$,2)
          Alphabet$(C) = Digit$
       NEXT C
END SUB
```

„DezNachSystem“ ist eine Funktion, die dezimale Zahlen (X##) in Zahlen im System zur Basis 128 (Zeichen$) umwandelt. Beispiel: 7702 → @N.

Die Länge der Klartextblocks und Blockschlüssel sind mit 44 Zeichen festgelegt.
Als Startsequenz wird eingegeben:

Um 7:30 fährt ein Zug von Rom nach Irgendwo [43 Bytes]

Das Programm errechnet die folgenden Bestimmungsfaktoren:

Trenn-Konstante C(t):	1764
Positionsgewichteter Wert (H(k)):	6757511
Bestimmungswert (H(p)):	593491919629
Gesamtwert (H(p)+H(k)):	593498677140

Aus den Bestimmungsfaktoren werden folgende Steuerungsparameter abgeleitet:

Variante	(H_k MOD 11) +1	=	3	Beginn der Kontraktion
Alpha	((H_k + H_p) MOD 255) +1	=	181	Offset Chiffre-Alphabet
Beta	(H_k MOD 169) +1	=	149	Offset Blockschlüssel
Gamma	((H_p + code) MOD 196) +1	=	23	Offset Matrixschlüssel
Delta	((H_k + H_p) MOD 155) +code	=	76	dynamische Bitfolgen
Theta	(H_k MOD 32) +1	=	8	Offset Rückrechnung
Kappa	(H_k MOD 14334)+1	=	9169	Anfang Chiffre-Alphabet

Verschlüsselt wird ein Artikel aus „Zeit Wissen“ April/Mai 2014:

Die Namen von Gottlieb Daimler und Carl Benz kennt jeder - sie erfanden 1886 das Automobil. Andreas Flocken hingegen blieb unbekannt. Dabei entwickelte er mit seinen Ingenieuren zur selben Zeit ein Auto, dessen Konzept seit Kurzem als die Zukunft der individuellen Mobilität gilt: das erste Elektroauto. Anfangs waren stromgetriebene Autos sogar erfolgreicher als ihre knatternden, stinkenden Konkurrenten. Im Jahr 1900 fuhren in den USA 1575 Elektroautos und 936 mit Benzinmotor. Doch nur ein Teil der Bevölkerung hatte Strom. Und der Klimawamdel war kein Thema. Der Verbrennungsmotor setzte sich durch.

Elektroauto, Zeit Wissen NR.3 April/Mai 2014

Die Datei **EL-Auto.**txt umfasst 662 Bytes.

Als ersten Block des zu verschlüsselnden Klartextes werden 44 Bytes eingelesen:

Die Namen von Gottlieb Daimler und Carl Benz

44 69 65 20 4E 61 6D 65 6E 20 76 6F 6E 20 47 6F 74 74 6C 69 65 62
20 44 61 69 6D 6C 65 72 20 75 6E 64 20 43 61 72 6C 20 42 65 6E 7A

Der Klartext im Bitsystem zur **Basis 8** wird in Abschnitte des Bitsystems zur **Basis 11** umgewandelt. Die dezimalen Werte der umgewandelten Zeichen sind Indexwerte für die Positionen der Zeichen im Chiffre-Alphabet zur Basis 11. Die Indizes müssen um +1 erhöht werden, da der Index „**0**“ im Array des Chiffre-Alphabets nicht erkannt wird.

Klartext
Basis 8: D i e N a m e
01000100 01101001 01100101 00100000 01001110 01100001 01101101 01100101 . . .
Basis 11:
01000100011 01001011001 01001000000 10011100110 00010110110 10110010101. . .

Index:	547	601	576	1254	182	1429
(+1)	548	602	577	1255	183	1430
Chiffre:	Ûí	Ùh	ÙI	{u	Þ8	|ô

System-Alphabet zur Basis 11:

. . . . . .
Alphabet$(183) = Þ8
Alphabet$(548) = Ûí
Alphabet$(577) = ÙI
Alphabet$(602) = Ùh
Alphabet$(1255) = {u
Alphabet$(1430) = |ô
. . . . . .

Chiffretext:
ÛíÙhÙI{uÞ8|ô}MþþÝýÝåþîüÞÙCÛ3{½|qÞÇÞûýaüeÇ4}Q¯î|
´¼ÇÙømƒ6ƒ9°KØz£éóàóIá&ÖuØÖ°Åóïókóç×JƒuáÕÜú°DªoÜØ°qÖÅømøõƒJØnÜôñ½ÑQ
ñSßäÓWzX@Þz®#éÞHµcÔèyéÕTÚx#ªµåþ9zjÔGÔèÚFÞxÓ®zÕõIõkß£zý&lÚh&úµ×Ò×Þp
á3ƒûÖûññøKñ6£3Øµƒüó¼ªqÑOñKñJÑÜú2ƒDíûÜyÑÑ£×£lóFø
´ƒDƒûÖáªþáKñRúEúâÅyï4Éòù
´ìÄëgòLôLÄyï4Éòô¯ìëôÑÉrôIÅQêvÄJûøù#ëgærôSÅÒï4ÉÔëýôîôÄûíôPJõC@PÜPxG
qE{P|HUI¯IýDÆPùFqFrM´MÒHúM½QEQAIèDbLmNMEdG¼LÿPxIäNrP´NGV|
RÚXlYõeWSKZ®RÅXIYúQzc5bfQâSGT}UªYtQvdMSfS3WúZçVQUÙQ}cæQfa@aÇbFÙÕ¯Ç
{¬éX¯á{yédÛjÝÆ¯ÔÚXâ2Ú×}£|d}l¯ÝýKý¼üHéÖ{yþò}l¯s|3ÚXén¯ùÇ3 . . . . .
. . . . . .

EA A1 EB 68 EB 49 7B 75 E8 38 7C 93 7D 4D E7 E7 ED EC ED 86 E7 8C 81 E8 EB
43 EA 33 7B AB 7C 71 E8 80 E8 96 EC 61 81 65 80 34 7D 51 EE 8C 7C 40 EC 9F
E6 8E 81 4B 81 A4 7D 8E E8 87 EF AC 80 EB 9B 6D 9F 36 9F 39 A7 4B 9D 7A 9C
82 A2 85 A2 49 A0 26 99 75 9D 99 A7 8F A2 8B A2 6B A2 87 9E 4A 9F 75 A0 E5
9A A3 A7 44 A6 6F 9A 9D A7 71 99 8F 9B 6D 9B E4 9F 4A 9D 6E 9A 93 A4 AB A5
51 A4 53 E1 84 E0 57 7A 58 40 E8 7A A9 23 82 E8 48 E6 63 E2 8A 79 82 E5 54
E9 78 23 A6 E6 86 E7 39 7A 6A E2 47 E2 8A E9 46 E8 78 E0 A9 7A E5 E4 49 E4
6B E1 9C 7A EC 26 6C E9 68 26 A3 E6 9E E3 9E E8 70 A0 33 9F 96 99 96 A4 A4
9B 4B A4 36 9C 33 9D E6 9F 81 A2 AC A6 71 A5 4F A4 4B A4 4A A5 9A A3 32 9F
44 A1 96 9A 79 A5 A5 9C 9E 9C 6C A2 46 9B EF 9F 44 9F 96 99 A0 A6 E7 0D 0A
A0 4B A4 52 A3 45 A3 83 8F 79 8B 34 90 95 97 EF 8D 8E 89 67 95 4C 93 4C 8E

```
79 8B 34 90 95 93 EE 8D 89 93 A5 90 72 93 49 8F 51 88 76 8E 4A 96 9B 97 23
89 67 91 72 93 53 8F E3 8B 34 90 E2 89 EC 93 8C 93 8E 96 A1 93 50 4A E4 43
40 50 9A 50 78 47 71 45 7B 50 7C 48 55 49 EE 49 98 44 92 50 97 46 71 46 72
4D EF 4D E3 48 A3 4D AB 51 45 51 41 49 8A 44 62 4C 6D 4E 4D 45 64 47 AC 4C
98 50 78 49 84 4E 72 50 EF 4E 47 56 7C 52 E9 58 6C 59 E4 65 57 53 4B 5A A9
52 8F 58 49 59 A3 51 7A 63 35 62 66 51 83 53 47 54 7D 55 A6 59 74 51 76 64
4D 53 66 53 33 57 A3 5A 87 56 51 55 EB 51 7D 63 91 51 66 61 40 61 80 62 46
EB E5 EE 80 7B AA 82 58 EE A0 7B 79 82 64 EA 6A ED 92 EE E2 E9 58 83 32 E9
9E 7D 9C 7C 64 7D 6C EE ED EC 4B EC AC 81 48 82 99 7B 79 E7 95 7D 6C 0D 0A
EE 73 7C 33 E9 58 82 6E EE 97 80 33 .. .. .. .. .. .. .. .. .. .. .. .. ..
```

Der Chiffretext **EL-Auto.**ctx umfasst 1032 Zeichen des System-Alphabets zur Basis 11.

6.2.2 „Verbund-Coding“

Beim Verbund-Coding werden verschiedene Operationen seriell verbunden, z.B. XOR-Operation mit Bit-Konversion und weiteren Operationen (dyn24, exchange).

Mit vorgeschalteter XOR-Verknüpfung vollzieht sich die Verschlüsselung in drei Funktionen:

1. Partielles dynamisches „One-time-pad“
   Klartext-Block → Blockschlüssel → 8-bit XOR-Verknüpfung
2. Bit-Konversion
   8-bit XOR-Verknüpfung → 7 bit Indexwerte (0 ...127)
3. Bestimmung des Chiffretexts
   7-bit Indexwerte → Chiffre-Alphabet (0...127) → Chiffretext.

6.2.3 Bestimmungsfaktoren

Als Beispiel werden die Verschlüsselungsschritte im Programm **DataCode.**exe (vgl. Anhang A) erläutert. Die Länge der Klartextblöcke und Blockschlüssel sind mit 42 Zeichen festgelegt. Als Startsequenz wird eingegeben:

Auf der Fischbachalm gibt es keine Heringe [42 Bytes]

Das Programm errechnet die folgenden Bestimmungsfaktoren:

Trenn-Konstante C(t):	1681
Positionsgewichteter Wert (H(k)):	6655148
Bestimmungswert (H(p)):	67391298875
Gesamtwert (H(p)+H(k)):	67397954023

Aus den Bestimmungsfaktoren werden folgende Steuerungsparameter abgeleitet:

Variante	$(H_k \text{ MOD } 11) +1$	=	6	Beginn der Kontraktion
Alpha	$((H_k + H_p) \text{ MOD } 255) +1$	=	199	Offset Chiffre-Alphabet
Beta	$(H_k \text{ MOD } 169) +1$	=	98	Offset Blockschlüssel
Gamma	$((H_p + \text{code}) \text{ MOD } 196) +1$	=	137	Offset Matrixschlüssel
Delta	$((H_k + H_p) \text{ MOD } 155) +\text{code}$	=	69	dynamische Bitfolgen
Theta	$(H_k \text{ MOD } 32) +1$	=	13	Offset Rückrechnung
Omega	$(H_k \text{ MOD } 95) +1$	=	19	Parameter Chiffre-Alphabet

CypherMatrix (Variation: D)

1	FA	7E	DE	FD	F0	05	CD	FC	E9	7D	D2	CF	52	22	C9	99	*16*
17	0D	0B	16	A1	4F	2B	B7	2E	7A	1F	A2	C3	CE	E0	81	83	*32*
33	EA	55	64	7F	97	D0	F8	30	14	74	EF	AA	20	93	75	F6	*48*
49	67	D4	70	DB	46	0F	ED	F9	7C	C2	11	FF	56	B2	6E	D3	*64*
65	B3	A6	90	C6	15	4D	FB	01	BE	D5	77	40	1C	BB	28	5C	*80*
81	C4	F1	A9	82	79	17	B6	45	BC	9B	02	73	FE	78	60	3A	*96*
97	31	53	12	32	9C	88	DD	95	EE	D7	BF	B8	9F	87	84	39	*112*
113	DC	E5	92	13	E2	72	06	21	C1	DA	85	65	5A	6D	6B	E4	*128*
129	CC	E8	24	EC	51	34	76	27	C5	AF	AE	98	4C	10	89	26	*144*
145	5D	D1	86	BD	80	1A	3F	F7	CB	48	2A	EB	23	91	D6	1E	*160*
161	B0	F5	29	0C	9A	A8	AD	DF	5B	F4	B1	63	5F	4E	E7	62	*176*
177	44	4A	4B	E1	6A	A0	59	A4	50	A3	6C	A5	43	69	A7	E3	*192*
193	6F	F2	04	AB	B5	03	B9	8B	2F	8A	33	8F	00	D8	B4	47	*208*
209	AC	94	19	3B	18	8C	36	38	3E	54	09	1B	07	35	5E	66	*224*
225	E6	61	F3	8E	0E	8D	1D	2C	0A	CA	C7	C8	25	7B	2D	57	*240*
241	9E	D9	3D	42	71	9D	41	C0	58	49	BA	68	37	96	08	3C	*256*

CypherMatrix (Runde 1)

1	·	~	Ì	²	≡	♣	═	³	Ú	}	Ê	¤	R	"	╔	Ö	*16*
17	♪	◙	─	í	O	+	À	.	z	▼	ó	╞	╬	Ó	ü	â	*32*
33	Û	U	d	▯	ù	ð	°	0	¶	t	´	¬	□	ô	u	÷	*48*
49	g	È	p	▉	F	☼	Ý	¨	|	┬	◄	⌂	V	▓	n	Ë	*64*
65	▏	ª	É	ã	Ş	M	¹	☺	¥	€	w	@	└	╗	(	\	*80*
81	─	±	®	é	y	↨	Â	E	╝	ø	☻	s	■	x	`	:	*96*
97	1	S	↕	2	£	ê	¦	ò	¯	Î	┐	©	ƒ	ç	ä	9	*112*
113	▄	Õ	Æ	‼	Ô	r	♠	!	┴	┌	à	e	Z	m	k	õ	*128*
129	╠	Þ	Ş	ý	Q	4	v	'	┼	»	«	ÿ	L	►	ë	&	*144*
145	]	Ð	å	¢	Ç	→	?	¸	╦	H	*	Ù	#	æ	Í	▲	*160*
161	▒	Ş	)	Ü	¿	¡	■	[	▕	▒	c	_	N	þ	b	D	*176*
177	D	J	K	ß	j	á	Y	ñ	P	ú	l	Ñ	C	i	°	Ò	*192*
193	o	≥	♦	½	Á	♥	╣	ï	/	è	3	Å	□	Ï	┤	G	*208*
209	¼	ö	↓	;	↑	î	6	8	>	T	o	←	●	5	^	f	*224*
225	µ	a	¾	Ä	♬	ì	↔	,	◙	╩	Ã	╚	%	{	-	W	*240*
241	×	┘	=	B	q	Ø	A	└	X	I	╟	h	7	û	◘	<	*256*

Chiffre-Alphabet (hex)

1	B9	8B	2F	8A	33	8F	D8	B4	47	AC	94	3B	8C	36	38	3E	*16*
17	54	35	5E	66	E6	61	F3	8E	8D	CA	C7	C8	25	7B	2D	57	*32*
33	9E	D9	3D	42	71	9D	41	C0	58	49	BA	68	37	96	3C	FA	*48*
49	7E	FD	F0	CD	FC	E9	7D	D2	CF	52	C9	99	A1	4F	2B	B7	*64*
65	2E	7A	A2	C3	CE	E0	81	83	EA	55	64	7F	97	D0	F8	30	*80*
81	74	EF	AA	93	75	F6	67	D4	70	46	ED	F9	7C	C2	FF	56	*96*
97	6E	D3	B3	A6	90	C6	4D	FB	BE	77	40	BB	28	5C	C4	F1	*112*
113	A9	82	79	B6	45	BC	9B	73	FE	78	60	3A	31	53	32	9C	*128*

Chiffre-Alphabet (Basis 7)

1	╣	ï	/	è	3	Å	Ï	┤	G	¼	ö	;	î	6	8	>	*16*
17	T	5	^	f	µ	a	¾	Ä	ì	╩	Ã	╚	%	{	-	W	*32*
33	×	┘	=	B	q	Ø	A	└	X	I	╠	h	7	û	<	·	*48*
49	~	²	≡	═	³	Ú	}	Ê	¤	R	╒	Ö	í	O	+	À	*64*
65	.	z	ó	├	╬	Ó	ü	â	Û	U	d	▯	ù	ð	°	0	*80*
81	t	´	¬	ô	u	÷	g	È	p	F	Ý	¨	|	┬	▯	V	*96*
97	n	Ë	┃	ª	É	ã	M	¹	¥	w	@	╗	(	\	─	±	*112*
113	®	é	y	Â	E	╝	¢	s	■	x	`	:	1	S	2	£	*128*

Zur Erläuterung wird die Datei **„Reiter.**txt**“** (1034 Bytes) verschlüsselt:

Was ich zu berichten beabsichtige, ist mir vor reichlich einem halben Jahrhundert im Hause meiner Urgroßmutter, der alten Frau Senator Feddersen, kundgeworden, während ich, an ihrem Lehnstuhl sitzend, mich mit dem Lesen eines in blaue Pappe eingebundenen Zeitschriftenheftes beschäftigte; ich vermag mich nicht mehr zu entsinnen, ob von den "Leipziger" oder von "Pappes Hamburger Lesefrüchten". Noch fühl ich es gleich einem Schauer, wie dabei die linde Hand der über Achtzigjährigen mitunter liebkosend über das Haupthaar ihres Urenkels hinglitt. Sie selbst und jene Zeit sind längst begraben; vergebens auch habe ich seitdem jenen Blättern nachgeforscht, und ich kann daher um so weniger weder die Wahrheit der Tatsachen verbürgen, als, wenn jemand sie bestreiten wollte, dafür aufstehen; nur so viel kann ich versichern, daß ich sie seit jener Zeit, obgleich sie durch keinen äußeren Anlaß in mir aufs neue belebt wurden, niemals aus dem Gedächtnis verloren habe.
Theodor Storm: Der Schimmelreiter, Novelle (1888)

In jeder Runde werden Klartextblöcke von 42 Bytes Länge mit gleich langen Blockschlüssel XOR-verknüpft.

Als ersten Klartextblock liest das Programm folgende 42 Bytes ein:

Was ich zu berichten beabsichtige, ist mir

57 61 73 20 69 63 68 20 7A 75 20 62 65 72 69 63 68 74 65 6E 20
62 65 61 62 73 69 63 68 74 69 67 65 2C 20 69 73 74 20 6D 69 72

01010111 01100001 01110011 00100000 01101001 01100011 01101000 00100000
01111010 01110101 00100000 01100010 01100101 01110010 01101001 01100011 . . .

Der ab Position **98** der CypherMatrix entnommene Blockschlüssel umfasst:

S↕2£ê¦ò¯Î┐ ©ƒçä9▄ÕÆ‼Ôr♠!┴ ┌àeZmkõ╠Þ$ýQ4v'┼»«

53 12 32 9C 88 DD 95 EE D7 BF B8 9F 87 84 39 DC E5 92 13 E2 72
06 21 C1 DA 85 65 5A 6D 6B E4 CC E8 24 EC 51 34 76 27 C5 AF AE

01010011 00010010 00110010 10011100 10001000 11011101 10010101 11101110
11010111 10111111 10111000 10011111 10000111 10000100 00111001 11011100

XOR-Verknüpfung:
Klartext:

	W	a	s		i	c	h	
01010111	01100001	01110011	00100000	01101001	01100011	01101000	00100000	. . . .

Blockschlüssel:

01010011	00010010	00110010	10011100	10001000	11011101	10010101	11101110	. . . .

XOR:

00000100	01110011	01000001	10111100	11100001	10111110	11111101	11001110	

hex:	04	73	41	BC	E1	BE	FD	CE
ASCII:	♦	s	A	╝	ß	¥	²	╬

♦sA╝ß¥²╬¡╧ÿ²Ô÷P┐ ìµvîRdDá©÷9♣▼ì½ì◘╠8G☺●¿ã▄

04 73 41 BC E1 BE FD CE AD CA 98 FD E2 F6 50 BF 8D E6 76 8C 52
64 44 A0 B8 F6 0C 39 05 1F 8D AB 8D 08 CC 38 47 02 07 A8 C6 DC

Als Ergebnis entsteht ein partielles **„one-time-pad“**. Klartext und Schlüssel sind gleich lang und der Schlüssel wird auch nicht wiederholt. In jeder Runde wird ein anderer Schlüssel aus der betreffenden CypherMatrix entnommen. Da in jeder Runde dasselbe geschieht, entsteht auf diese Weise eine Kette von „one-time-pads“, die in ihrer Wirkung wie eine einheitliche Funktion den gesamten Klartext verschlüsseln.

6.2.4 Bit-Konversion

Das Ergebnis der XOR-Verknüpfung im Bitsystem zur Basis 8 wird in Zeichen des Bitsystems zur Basis 7 umgewandelt. Die dezimalen Werte der umgewandelten Zeichen (Basis 7) sind Indexwerte für die Positionen der Zeichen im Chiffre-Alphabet. Die Indizes für das Chiffre-Alphabet müssen um +1 erhöht werden, da der Index „**0**“ im Array des Chiffre-Alphabets nicht erkannt wird.

Bit-Konversion:
Ergebnis XOR Basis 8:

00000100	01110011	01000001	10111100	11100001	10111110	11111101	11001110	. . . .

Konversion (Basis 8 nach Basis 7):

0000010	0011100	1101000	0011011	1100111	0000110	1111101	1111101	1100111	0 . . . .

Index:	2	28	104	27	103	6	125	125	103
(+1)	3	29	105	28	104	7	126	126	104

Chiffre Basis 7:

	/	%	¥	╚	¹	Ï	S	S	¹ . . .

Als Ergebnis der Bit-Konversion entsteht der Chiffretext der Datei **Reiter.**ctx (1253 Zeichen).

/%¥╚¹ÏSS¹hRIâsÓøX·é□═Ýì¬5µ;âpìR/âéÝ|³5ù%5n×OB6|
B</«Ð²wöèD{=Ð°«ªÀÞï┬¨¨N£³½¥B$bJÕkN╣¨┼êo3ìÜ│Àh□¢t
D─┐D¹©Ü$ýÆì╩└┼øc6ç[aµY±É¯áHõ¢_óë¸┐ýíú?ÿÕ┐ll(IXØ
ö{z/├°ÐÔ#├□KL³nÖÎçC╔ÚWÔ%&C{GAÅ╩³Q¬~rU¥ÆÃ├Èê+eqIã
ä┼0lplOéåùT!^üùlJ×X+¿{X╟°a/§┐VtTk\Éò*>jÃv]Ç╬î©ô
s┐Òoz<F5*Å»╩Çâ*æp┐fãP>gTgËlñFlo╚├§?┴qÏóÁz»F-±>h~
3èÈ°âÆ:àdñìÈÙSxO/¨%4W╬╦KpÃö╣}ù┤F_═È╝G│K±─t┤~åÙk©

zÝ^□h&¨bß8O┐;G¨W(Cò±ã?¥ÖUWìêÈ8å7»è□à>ÇøóøM□>ê*Z6
■ÿèß{N╝ P:╝|)╟╗ëî-╟┬#JÖ┤ìN¢QJUf{(ÏJqâ¡àú¯F¤½fP└Â
V~╝)°:1╜dBÿØå┴Ð©ƒb╝□L□DLÑbÄjÝ`.ƒ{)╣÷È□Ð╚Ãç\1*ÝÂ=
=╣╗╠¬■©©99Üï>]´Ö3╬-Òn3ÐË*i¾&─<□┼┴@Íü╣Ä■+╦6Ú©o¬
×÷j#Z^ý$vÅ«³aÊ¤B-Ê®─X¥&.ë╣2╝|ù┴Îü┤Ç}┴-╟X§÷u#|╨8┴
QÆÓëÊZ[pfúÒ[¹}dýtîKéÕöä|tÍì□└rCîr─nx├└ç╦Rräï7b
¯ª°ÍÀÊi╦┐ó~®i²vû¼3}¿ø┐.¸=ýX÷╚°âÀE┌õMÐö¹Åüè╚@²÷┌
F¸³Ä+ÜÒcu=²╣ZvvÙÛa╠Z□.<3ZãQ_YÈJ3Ä┴/éeçíBvç■┬Mw
╟n5|dt×.@@öMI4┬3LƒD±DL[<¤7Sw°úù4~╔s5oû\ü±l&Go╗ƒÇ
moD9}^Uî®Pƒ┼mƒx─l|╝*╟┌S2ôiÐé¤ƒ]:UECàür8Ú\Ý¡Fâë¡s
Bm╨ê©Í'RÍÃÂÄÑÐ¿JJýû%╨ïjlR╨¹!Áé@9Tí¾┘q.┘╣áÐgûî®ú
íÔC¥ÙÁÒV═Ú$«çÂÄ3°ý╠ûäéä:╬âY┌#t?ÜÚ¯¯8êR{í5äƒ¸Ý`F´
mÙ¿Â├ÎƒÐ*¶¤┼çàmn¹Â¤³(┐r│¿n±NäzMòÃ╟üê§»v¾&çx!Ú(½÷
M╝z0┘Ð¸oâÄäP®î┐AÅÂøKrÐoJÿ═PÇ┌»Ã¬+cÎÿÃïÔO┤¸╠ùä╦
ß|G═╗k²815Ô¸2>Ïì╨ÿåMªÐç¼µígLáÚo╠2jÊ4U¾┤l³nLkçt°õ
A~1kÿ6.=CO#@à£µa¸ei¥%t£·óè³ðöû└a┼Åa@®9@AÞè=S??Ö┐
³öa5;□Ñû¿é&^²¶╔F¨8¢é}╟ôõO¼┴BQ╠«ÑO╟'e├SÆÚ·î¸'WT5Ú
@ÀÖÈu>Y«»øé┬toC°╨¾!]È¬|6>□╔O¢÷ª|]3#Í□jOoé─ardõq╝
¶ÓÖ

7 Entschlüsselung

Für die Entschlüsselung erzeugt der Generator einen inhaltsgleichen Ablauf wie bei der Verschlüsselung. Die Entschlüsselung wird im Codierbereich abgearbeitet, nur in der umgekehrten Reihenfolge:

1. Analyse des Chiffretextes
   Chiffretext → Chiffre-Alphabet (0...127) → 7 bit Indexwerte
2. Bit-Konversion
   7 bit Indexwerte (0 ...127) → 8-bit XOR-Verknüpfung
3. XOR-Verknüpfung
   8-bit XOR-Verknüpfung → Blockschlüssel → Klartext-Block

Aus Blöcken von **48** Zeichen Chiffretext sucht das Verfahren im identisch erzeugten Chiffre-Alphabet die dezimalen Index-Werte der einzelnen Zeichen und verbindet deren binäre Zahlen zu einer Bitfolge von 336 Bits. Diese Bitfolge wird wiederum in **42** 8-bit Sequenzen (336 Bits) im Bitsystem zur Basis 8 aufgeteilt und mit dem entsprechenden Blockschlüssel XOR-verknüpft.

Chiffretext (Basis 7):
Index: 3 29 105 28 104 7 126 126 104
(-1): 2 28 104 27 103 6 125 125 103
0000010 0011100 1101000 0011011 1100111 0000110 1111101 1111101 1100111 0 . . . .
Konversion (Basis 7 nach Basis 8):
00000100 01110011 01000001 10111100 11100001 10111110 11111101 11001110 . . . .
Blockschlüssel (Basis 8):
01010011 00010010 00110010 10011100 10001000 11011101 10010101 11101110 . . . .
Klartext (Basis 8 / XOR-verknüpft):
01010111 01100001 01110011 00100000 01101001 01100011 01101000 00100000 . . . .
W a s i c h

Als Ergebnis erscheint der ursprüngliche Klartext.

8 Sicherheit des Verfahrens

Zu den bekanntesten Angriffen gehören Strukturanalyse, „known plaintext attack“ und „chosen plaintext attack“, eventuell auch noch „differenzielle“ und „lineare“ Kryptoanalyse. Mit diesen Angriffen sollen aus dem Chiffretext statistisch erfassbare Regelmäßigkeiten herausgefiltert werden, die möglicherweise einen Weg zum Klartext aufzeigen. Zu den Auffälligkeiten der Sprache zählen Wiederholungsmuster und Wortkombinationen, Häufigkeitsstrukturen und Bigramme [#10].

Eine Analyse aller dieser Merkmale setzt allerdings voraus, dass Klartext und Chiffretext sich auch vergleichen lassen. Insoweit muss ein einheitliches Ordnungssystem bestehen, das sowohl im Klartext als auch im Chiffretext wirksam ist. Im CypherMatrix Verfahren ist dass nicht der Fall. Bei Verschlüsselungen findet ein Wechsel im Bitsystem statt mit der Folge, dass Klartext und Chiffretext sich nicht mehr vergleichen lassen. Durch die Umwandlung der Zeichen vom Bitsystem zur Basis 8 in Zeichen zur Basis 7 entfällt auf jedes Klartextzeichen ein Chiffrezeichen das um den Faktor 1,143 (8/7) länger ist als das zugehörige Klartextzeichen. Es fehlt die Basis für alle herkömmlichen Angriffszenarien. Sie sind wirkungslos und daher wir können sie vergessen.

Die **Startsequenz** (Passphrase) zum Initialisieren des Generators kann zwischen 36 und 64 Zeichen (optimal 42 Bytes) festgelegt werden. Bei Eingabe über die Tastatur mit etwa 100 Zeichen umfasst die Startsequenz dann folgenden Schlüsselraum:

36 Bytes →	100^36 = 1E+ 72	Länge: 240 Bit	Entropie: 239.188
42 Bytes →	100^42 = 1E+ 84	Länge: 280 Bit	Entropie: 279.042
64 Bytes →	100^64 = 1E+128	Länge: 426 Bit	Entropie: 425.207

Die Start-Sequenz wird nur einmal zum Start des Verfahrens verwendet und nicht auf der Festplatte gespeichert.

Der **Matrixschlüssel** aus der CypherMatrix mit 42 Zeichen entnommen entfaltet maximal folgenden Schlüsselraum:

42 Bytes →	256^42 = 1.4E+101	Länge: 336 Bit	Entropie: 336.000

Bei einem Angriff auf den Matrixschlüssel mit 42 Bytes Länge ergibt sich eine Entropie von 336 und eine exponentielle Komplexität von O(2^336) = **1.4E+101** [#11].
Angenommen in einem „brute force“ Angriff dauere eine Exhaustionsrunde 1 nano sec, dann würden 5E+89 Jahre notwendig sein, den Schlüssel mit Sicherheit zu finden.

Die für die laufende Sicherheit des Generators wichtigen Schritte sind vor allem:

a) Vermeidung von Kollisionen,
b) erste Einwegfunktion: „Expansion“ zur Hashfunktionsreihe und
c) zweite Einwegfunktion: „Kontraktion“ zur BASIS VARIATION.

Mit Einbindung dieser Funktionen ist eine rückwärts gerichtete Bestimmung vorhergehender Daten nicht möglich.

Die **CypherMatrix** (16x16 Zeichen) wird in jeder Runde neu generiert. Eine Wiederholung der gleichen Matrix tritt erst nach 256! (Fakultät) = 8.57E+506 Fällen auf (2*10^1684).

Das **System-Alphabet** mit 128 Zeichen nimmt 212 Bytes aus der CypherMatrix von 256 Zeichen. 44 Zeichen bleiben unberücksichtigt.

212 Bytes → 212^128 = 5.9E+297 Umfang: 989 Bit Entropie: 989,173

8.1 „Verbund-Coding“

Beim Verbund-Coding werden verschiedene Operationen seriell verbunden, z.B. XOR-Verknüpfung mit Bit-Konversion und weiteren Operationen (dyn24, exchange).

Beim „Verbund-Coding“ besteht für jeden Blockschlüssel ein partielles „**one-time-pad**“ [#12]. Klartextblock und Blockschlüssel sind gleich lang und der Schlüssel wird auch nicht wiederholt. In jeder Runde wird ein anderer Schlüssel aus der betreffenden CypherMatrix entnommen. Das ergibt für den gesamten Verschlüsselungsvorgang eine **Kette** als zusammenhängende „one-time-pad“-Funktion. Nach derzeitiger Auffassung wird dadurch eine absolute Sicherheit erreicht [#13].

8.2 „brute force“ Angriff

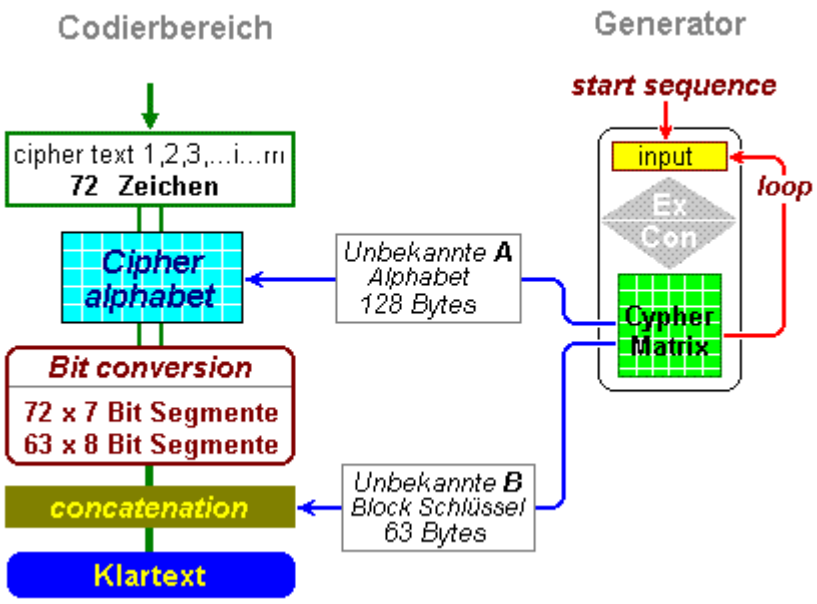
Immerhin bleibt noch die Möglichkeit einer „brut force“ Attacke [#14]. Einem Angreifer sind grundsätzlich nur der Chiffretext und das CypherMatrix Verfahren bekannt. Das jeweilige Programm und die einzelnen Steuerungsparameter, einschließlich der Startsequenz, kennt er nicht. Ihm bleiben die folgende Angriffszenarien:

1. Am Beginn die „Startsequenz“ zu suchen,
2. in das laufende Verfahren einzusteigen oder
3. vom Ende her den Weg vom Chiffretext zum Klartext zu finden.

Ein Versuch, aus dem laufenden Verfahren vom Chiffretext auf die Startsequenz oder den Matrix Schlüssel zu schließen, muss aussichtslos bleiben. Zwischen beiden gibt es keine Verbindung und beide werden für die Verschlüsselung nicht verwendet.

Weiterhin bleibt die Möglichkeit, vom Chiffretext auszugehen und den Vorgang vom Ende her aufzurollen. Dazu müsste das letzte System-Alphabet im Bitsystem zur Basis 7 und die Verteilung der 128 Alphabet-Zeichen herausgefunden werden. Die richtige Verteilung könnte nur durch eine Iteration gesucht werden. Für die Verteilung besteht ein Kombinationsraum von 128! (Fakultät) = **2.85E+215** Möglichkeiten.

Wenn für eine bestimmte Runde, aus welchen Gründen auch immer, das System-Alphabet zur Basis 7 dennoch bekannt werden sollte, bleibt die Aufgabe, den jeweiligen Blockschlüssel herauszufinden. Die Entschlüsselung geschieht in jedem Durchgang wie folgt:



Das Verfahren enthält drei Funktionen:

1. Klartextblock --> **Blockschlüssel** --> -8 bit XOR-Sequenzen
2. 8-bit XOR Sequenzen --> 7-bit Index-Werte
3. 7-bit Index-Werte --> **Chiffre-Alphabet (128)** --> Chiffretext

In diesen Funktionen sind die Parameter **Blockschlüssel** und **Chiffre-Alphabet** zwei voneinander unabhängige Variable. Es gelten:

cm = f [f1 (an, **k1**), f2 (b1, b2), f3 (b2, **k2**)]
an = f [f3 (cm, **k2**), f2 (b2, b1), f1 (b1, **k1**)]

fx = funktionale Verbindung
an = Klartext
k1 = **Blockschlüssel**
b1 = 8-bit Sequenz
b2 = 7-bit Index-Wert
k2 = **Chiffre-Alphabet (128)**
cm = Chiffretext

Die Ermittlung des Chiffretextes „**cm**“ und die retrograde Suche nach dem Klartext „**an**“ zeigen sich somit als Gleichungen mit zwei unbekannten Veränderlichen: **k1** und **k2**. Das führt bekanntlich nur dann zu einer eindeutigen Lösung, wenn eine Unbekannte aus der anderen abgeleitet werden kann oder wenn zwei Gleichungen mit denselben Unbekannten vorhanden sind.

Aber zwischen dem jeweiligen Blockschlüssel = k1 und dem in derselben Runde generierten Chiffre-Alphabet (128) = k2 gibt es keine Verbindung. Beide sind zwar aus der aktuellen CypherMatrix entnommen, haben aber keine funktionale Beziehung: (k1 --> (**Hk MOD 169**)**+1** und k2 --> ((**Hk +Hp**) **MOD 255**)**+1**). Die Runden CypherMatrix selbst ist aus der ursprünglichen Start-Sequenz hergeleitet. Dahin führt jedoch kein Weg zurück (zwei Einwegfunktionen stehen dagegen). Insoweit kann auch hier kein erfolgversprechender Weg gefunden werden.

9 Beispiele zum Testen

Im Anhang A sind einige vom Autor entwickelte Programme aufgeführt. Für jedes einzelne Programm kann der Quellcode per e-mail beim Autor angefordert werden (eschnoor@multi-matrix.de).

10 Hinweise

[#1] Algorithms, Key Sizes and Parameters Report, 3. Primitives, www.enisa.europa.eu
[#2] Singh, Simon, Geheime Botschaften. München 2004, S.298
[#3] Morin, Charles, www.kbcafe.com/articles/HowTo.Base64.pdf
[#4] Swoboda J., Spitz St., Pramateftakis M., Krytographie und IT-Sicherheit, Wiesbaden 2008, S.22
[#5] Schmeh K., Safer Net, Heidelberg1998, S.61
[#6] Paar Ch., Pelzl J., Understanding Cryptography, Berlin-Heidelberg, 2010, S.124
[#7] Strukturvergleich Klartext und Geheimtext, www.telecypher.net/Equilang.pdf
[#8] Singh S.,a.a.O., S.48
[#9] Schäfer Michal B., Changeset 4524 for trunk/Templates/Experimental/CypherMatrix https://www.cryptoll.oeg/trac/CrypRool2/changeset/.../CypherMatrix.xml
[#10] Bauer F.L., Entzifferte Geheimnisse, Berlin Heidelberg NewVork 1995, S.36, 78
[#11] Schneier B., Angewandte Kryptographie (dt.Ausgabe), Bonn … 1996, S.278

[#12] Schneier B., a.a.O., S. 17
[#13] Schneier B., a.a.O., S.275
[#14] Schneier B., a.a.O., S.177
[#15] Singh S., a.a.O., Sn.183 -193.
[#16] Singh S., a.a.O., S.161

München, im März 2014

Anhang A

Nach den Grundsätzen der „Codegraphie“ hat der Autor eine Reihe von Programmen entwickelt, die in der folgenden Liste zusammengestellt sind.

System Basis	System Alphabet	Basis-Coding (ohne XOR-Funktion) einfache Matrix	Verbund-Coding (mit XOR-Funktion) einfache Matrix	Längen-verhältnis
1	2	Crypto01	MonoCode	1:8
2	4	Crypto02	ZweiCode	1:4
3	8	Crypto03	DreiCode	1:2,66
4	16	Crypto04	VierCode	1:2
5	32	Crypto05	QuinCode	1:1,6
6	64	Crypto06	CM64Code	1:1,33
7	128	Crypto07	DataCode DynaCryp CodeData[1)] QuadCode[2)]	1:1,143 1:1,143 1:1,143 1:1,143
8	256	Crypto08 CMCode8D System08	PlanCode MyCode08	1:1 1:1 1:1
9	512	Crypto09 System09	NeunCode MyCode09	1:1,79 1:1,79
10	1024	Crypto10 System10	ZehnCode MyCode10	1:1,6 1:1,6
11	2048	Crypt11A Crypt11B System11	ElvaCode MyCode11	1:1,46 1:1,46
12	4096	Crypto12 System12	MegaCodA MegaCodB MyCode12	1:1,33 1:1,33 1:1,33
13	8192	System 13	MyCode13	1:1,23
14	16384	System14	MyCode14	1:1,143

[1)] Programm mit drei Operationen (XOR – bit conversion - exchange),
[2)] Programm mit vier Operationen (dyn24 – XOR – bit conversion – exchange).

Für jedes einzelne Programm kann der Quellcode per e-mail beim Autor angefordert werden (eschnoor@multi-matrix.de).

Um die Sicherheit des Verfahrens einmal selbst zu testen, können Sie sich das Programm **telecypher.net/securita.7z** herunter laden und alles ausprobieren.

Die Klartextdatei *„Beispiel.txt“* wird mit dem Programm **„DataCode.**exe**“** verschlüsselt. Ein Teil vom Anfang des Klartextes ist bekannt. Außer einem „*ciphertext only*“ Angriff können auch ein *„chosen-plaintext“* Angriff und weitere Analysen vorgenommen werden. Die Startsequenz und der restliche Klartext sind zu suchen. Im Erfolgsfall erbittet der Autor eine entsprechende Nachricht per e-mail (eschnoor@multi-matrix.de).

Alle Programme sind DOS-Programme und laufen nur noch unter Windows XP. Sie müssen auf eine aktuelle Programmiersprache umgeschrieben werden. Im Rahmen der **CMLizenz** (vgl. Anhang B) können die Programme getestet und weiter entwickelt werden.

Unter Leitung von Prof. **Bernhard Esslinger** (Uni Siegen) und seinem CrypTool-Team [#1] (insbesondere: **Michael B. Schäfer**) sind die Programme **DataCode** und **DynaCode** bereits auf C# umgewandelt worden. Dabei hat sich ergeben, dass die in C# geschriebenen Programme etwa 25x schneller laufen als vorher unter Windows XP. Alle weiteren Programme warten allerdings noch auf eine entsprechende Bearbeitung.

[#1] Esslinger, Bernhard, Uni Siegen, http://www.cryptool,org/de/

Anhang B

L I Z E N Z

zur Weiterentwicklung und Nutzung der Software
des „CypherMatrix“ Verfahrens

CypherMatrix®

I.

„CypherMatrix“ Verfahren ist die vom Autor *Ernst Erich Schnoor, München*, gewählte Bezeichnung einer neuen Basisfunktion der Kryptographie mit folgenden Zweckbestimmungen:

1. Durchführung von Verschlüsselungen,
2. Berechnung von Hashwerten,
3. einfache und erweiterte Signaturen und
4. weitere im Einzelnen noch zu erforschende Aufgaben.

II.

„Software zur Gestaltung des Verfahrens“ ist jeder Quellcode, gleich welcher Art, der den vorstehenden Zweckbestimmungen direkt oder indirekt dient oder zu dienen bestimmt ist, im Folgenden kurz: *„Software“* genannt.

III.

Jeder Anwender kann die Software:

1. nach eigenem Ermessen anwenden,
2. die Funktionsweise uneingeschränkt studieren und an eigene Vorstellungen anpassen,
3. Kopien der Software umsonst weiter geben,
4. sowie die Software verbessern und diese Verbesserungen zur öffentlichen Diskussion stellen.

IV.

Jeder Anwender erhält das Recht, die Marke „CypherMatrix“ für seine Tätigkeiten nach vorstehendem Abschnitt III zu benutzen.

V.

Für kommerzielle Verwendung der Software ist die vorherige schriftliche Zustimmung des Autors erforderlich. Als kommerzielle Verwendung gilt jede entgeltliche Übertragung der Software, in welchem Umfang auch immer, und der mit Hilfe der Software hergestellten Produkte, ganz oder in Teilen. Ein Verstoß gegen diese Auflage hat eine Schadensersatzzahlung des Verursachers zur Folge.

München, den 27. Mai 2011

Ernst Erich Schnoor
(eschnoor@multi-matrix.de)