Neues in der digitalen Kryptographie

(Ernst Erich Schnoor)

In der klassischen Kryptographie war Gegenstand der Verschlüsselung das einzelne Zeichen. Mit Einführung der Computer haben sich die Verfahren auf die digitale Technik verlagert. Aber Computer können grundsätzlich nur zwei Zustände unterscheiden: „vorhanden“ (eine Spannung) oder „nicht vorhanden“ (keine Spannung), in Ziffern des Zahlensystems zur Basis 2: „eins“ oder „null“. Dieser Vorgang wird bekanntlich als „**Bit**“ bezeichnet.

1 Systematisierung der Bitfolgen

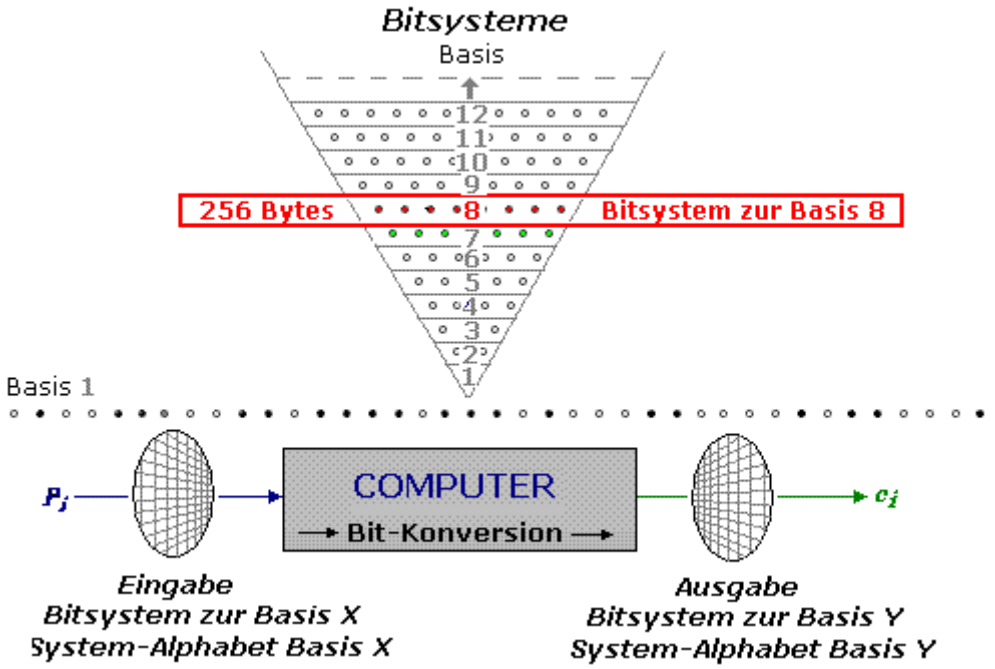
Eine Information ist mehrschichtig. Sie besteht aus mindestens zwei Bit, allgemein aus einer Folge von Bits. Die Bits sind in ihrer Menge unbegrenzt (1 bis ∞). Um mit Bitfolgen systematisch zu arbeiten, müssen sie systematisiert, d.h. skaliert und in feste Abschnitte (Units) geteilt werden. Es liegt ein vergleichbares Phänomen vor, wie bei der Menge aller Zahlen. Wie in der Zahlentheorie lassen sich auch Bitfolgen in einem Stellenwertsystem ordnen. Für 8-bit Folgen kann dann beispielsweise vom **„**Bitsystem zur **Basis 8“** gesprochen werden. Im Einzelnen:

				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

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

Der Aufbau des Stellenwertsystems wird am besten mit der kopfstehenden System-Pyramide dargestellt.



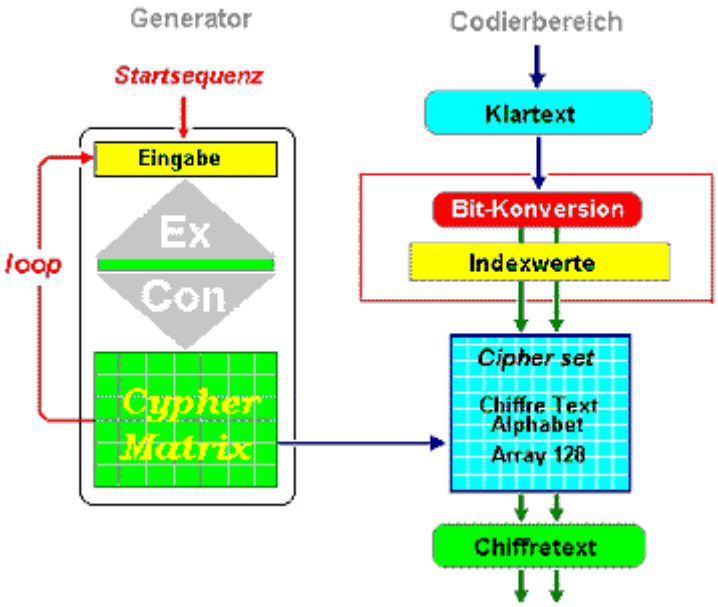
2 Das Ordnungssystem

In der aktuellen Kryptographie vollziehen sich fast alle Operationen in einem einheitlichen Bitsystem. Sowohl Eingaben als auch Ausgaben (verschlüsselte Ergebnisse, Chiffretext) werden im Bitsystem zur Basis 8 und mit einem System-Alphabet von 256 Zeichen (00 bis FF) verarbeitet. Außerdem wird gefordert, Klartext und Chiffretext müssten gleich lang sein [#1,#2). Dass zwischen Eingabe und Ausgabe vielfach auch Bitfolgen anderer Länge eingefügt sind, ist nicht entscheidend. Insoweit vollziehen sich alle Verschlüsselungs-operationen in einem einheitlichen **Ordnungsystem**, und zwar im Bitsystem zur **Basis 8**.

2 Das CypherMatrix Verfahren

Mit einem neuen Verfahren – vom Autor **CypherMatrix** genannt - können vor allem neue Verschlüsselungen und Hashwertberechnungen durchgeführt werden. Dabei ist die Verschlüsselung sehr einfach:

> Ein **Generator** erzeugt das erforderliche **System-Alphabet** und
> im **Codierbereich** wird der Chiffretext mit der **Bit-Konversion** geschrieben.



Beide Bereiche werden kombiniert, können aber auch getrennt verwendet werden.

2.1 “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.

2.1.1 ASCII-Zeichensatz

Das System-Alphabet im Bitsystem zur Basis 8 ist der ASCII-Zeichensatz in seiner jeweiligen Ausprägung. Für umfangreichere Zeichensätze – insbesondere fremde Sprachen – wird **Unicode** verwendet. Für jede Ausgabe des Computers ist ein sinnvolles System-Alphabet erforderlich. So für Bilder (Pixel, graphische Programme), Töne (digital/analog Wandler, Audio-Dateien), digitale Messgeräte, Strich-Code und für alle weiteren Ausgaben von skalierbaren und in Bitsystemen strukturierten Phänomenen.

2.1.2 Variation des System-Alphabets

Mit Generator und Startsequenz „**Leonardo erobert Florenz mit Schneekanonen**“ wird ein permutiertes System-Alphabet erzeugt, das zu folgender Variation der Basis-Verschlüsselung führt:

```
  1  □#–¾—#Žñ□#Pä#ž#°□□º@ÎNB#¥ó#˜™²½ë„òc+úÒ¼ùàfS1\,H2Üöˆsaå6ká9ÝôêÛõ3  64
 65  R#û…šœl>ü€Þ³-TüÊxhtVì.´XâY#Æ!jZŸ ¤÷¡Dyz#§¢{£|¨ý}©]#†#‡‰ŠÐ#7À‹^eÏ  128
129  gª_b«mÔ4¬8þ";#æãn<Œµ®Aç¿¯ÇÕl#±·¸¹»ÁÂÖ####ºò#ôþãa®}#Îl V~Æx#™\ï¿  192
193  æÀk¡…ˆÊ÷h#yõ£Bp#EqË9#ÃHlÒ<Á#UƒcR·¤É„u#!YS†ý#J#ÂÚP-¨W×€å—?`CÄçdøÞ  256
```

Neues System-Alphabet :

```
Alphabet$(98)  =  Z
Alphabet$(101) =  ¤
Alphabet$(102) =  ÷
Alphabet$(103) =  ¡
Alphabet$(105) =  y
Alphabet$(108) =  §
Alphabet$(111) =  £
Alphabet$(116) =  ©
```

Der Klartext „Bitfolge“ ergibt verschlüsselt den Chiffretext: „**Zy©÷£§¡¤ „**

Dieses Beispiel zeigt die einfachste Art einer Verschlüsselung. Das System-Alphabet muss nur in Textblöcken von z.B. 16 Bytes variiert werden (neue Runde). Zum Testen können Sie sich das Programm: "System08.exe" herunter laden und selbst ausprobieren. Der Quellcode ist im Zip-Paket enthalten.

2.2 „Bit-Konversion“

Bisher werden Umwandlungen von Bitfolgen nur 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 Sequenzen sind Indizes für ein Chiffre-Alphabet von 64 Zeichen. Das Chiffre-Alphabet (System-Alphabet) wird statisch vorgegeben.

Bit-Konversion 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 Bit-Konversion von Basis 1 nach Basis 8 geschieht im Einzelnen wie folgt:

Bitfolge Basis 1:
01100010011010010111010001100110011011110110110001100111011001010110110
Bitfolge Basis 8:
01100010 01101001 01110100 01100110 01101111 01101100 01100111 01100101 . . . . .
Index: 98 105 116 102 111 108 103 101
System-Alphabet (ASCII-Zeichensatz):
b i t f o l g e
System-Alphabet (ASCII): dezimal hexadezimal

Alphabet$(98)	=	b	62
Alphabet$(101)	=	e	65
Alphabet$(102)	=	f	66
Alphabet$(103)	=	g	67
Alphabet$(105)	=	i	69
Alphabet$(108)	=	l	6C
Alphabet$(111)	=	o	6F
Alphabet$(116)	=	t	74

In der Klartextausgabe lautet die Folge: „**bitfolge**“
Die hexadezimale Ausgabe lautet: „**626974666F6C6765**“

Die zugrunde liegende Bitfolge im Bitsystem zur Basis 1 bleibt unverändert (kein Bit wird hinzugefügt und kein Bit wird weggelassen). Das Beispiel verdeutlicht, das nur das System-Alphabet gewechselt werden muss, während die nicht strukturierte Bitfolge (Anzahl und Reihenfolge) bei jeder Konversion grundsätzlich gleich bleibt. Eine Bit-Konversion kann für alle Bitsysteme von zur Basis 1 bis zur Basis 16 (und höher) durchgeführt werden.

Historisch gesehen wurde eine Bitkonversion bereits 1963 am Anfang der digitalen Technik durchgeführt. Die einzelnen Bits im Bitsystem zur Basis 1 wurden in eine 7-Bit-Zeichencodierung (ASCII als Standard-Code) konvertiert. Als 128 Zeichen nicht mehr ausreichten, kam der 8-Bit-Code mit dem erweiterten ASCII-Zeichensatz von 256 Elementen, der noch heute weitgehend als Standard verwendet wird.

2.3 Bit-Konversion von Basis 8 nach Basis 7

Die Bit-Konversion von Basis 8 nach Basis 7 ist der haupt Anwendungsfall für Verschlüsselungen. Im Folgenden wird die Umwandlung am Beispiel des Programms **Crypto07.exe** erläutert.

Bitfolge Basis 8:
01100010 01101001 01110100 01100110 01101111 01101100 01100111 01100101 . . . . .
Index: 98 105 116 102 111 108 103 101
System-Alphabet (ASCII):
b i t f o l g e
Bitfolge Basis 7:
0110001 0011010 0101110 1000110 0110011 0111101 1011000 1100111 0110010 . . . . .
Index: 49 26 46 70 51 61 88 103 50
(+1) 50 27 47 71 52 62 89 104 51

Chiffretext:

e ¡ À ® g þ E & Ï

„bitfolge“ wird im Bitsystem zur Basis 7 wie folgt dargestellt: **e¡À®gþE&Ï**

Das System-Alphabet zur Basis 7 mit 128 Units wurde mit dem Generator und der Start-sequenz: „Leonardo erobert Florenz mit Schneekanonen“ erzeugt. Die Indexwerte sind um (+1) erhöht, da das System-Array den Index „0“ nicht erkennt.

Chiffre-Alphabet (Basis 7)

1	> ü € ³ – T ¥ Ê x h t V ì . ´ X â Y Æ ! j Z Ÿ ¤ ÷ **¡** D y z § ¢	*32*
33	{ £ | ¨ ý } ©] † ‡ ‰ Š Đ 7 **À** ‹ ^ **e Ï g** ª _ b « m Ô 4 ¬ 8 **þ** ; æ	*64*
65	ã n < Œ µ ® A ç ¿ ¯ Ç I · ¸ ¹ » Á Â Ö # $ ' C **E** J Q o í O È ?	*96*
97	K å è Ë :) **&** *ƒ* É % , = Ì Ù Ú Ñ u / › î ` p [Ã ¦ w ï ø ’ ì Ä	*128*

Chiffre-Alphabet (hex)

1	3E	FC	80	B3	2D	54	9D	CA	78	68	74	56	EC	2E	B4	58	16
17	E2	59	C6	21	6A	5A	9F	A0	A4	F7	A1	44	79	7A	A7	A2	32
33	7B	A3	7C	A8	FD	7D	A9	5D	86	87	89	8A	D0	37	C0	8B	48
49	5E	65	CF	67	AA	5F	62	AB	6D	D4	34	AC	38	FE	3B	E6	64
65	E3	6E	3C	8C	B5	AD	AE	41	E7	BF	AF	C7	49	B7	B8	B9	80
81	BB	C1	C2	D6	23	24	27	43	45	4A	51	6F	ED	4F	C8	3F	96
97	4B	E5	E8	CB	3A	29	7F	26	83	C9	25	82	3D	CC	55	DA	112
113	D1	75	2F	9B	EE	60	70	5B	C3	A6	77	EF	F8	92	8D	C4	128

2.3 Folgen der Bit-Konversion

Als Folge der Bit-Konversion zeigt sich eine grundsätzliche Wirkung des Verfahrens: Der Chiffretext wird länger im Verhältnis der Länge der Zeichen im Bitsystem des Chiffretexts zur Länge der Zeichen im Bitsystem des Klartextes. Die Forderung Klartext und Chiffretext sollten gleiche Länge haben, wird im Prinzip nicht erfüllt.

3 Neue Technik

Die wichtigsten Funktionen im CypherMatrix Verfahren sind der **Generator** und die **Bitkonversion.** Der Generator erzeugt die zur Verschlüsselung erforderlichen Steuerungs parameter und im Codierbereich wird mit der Bitkonversion der Chiffretext geschrieben.

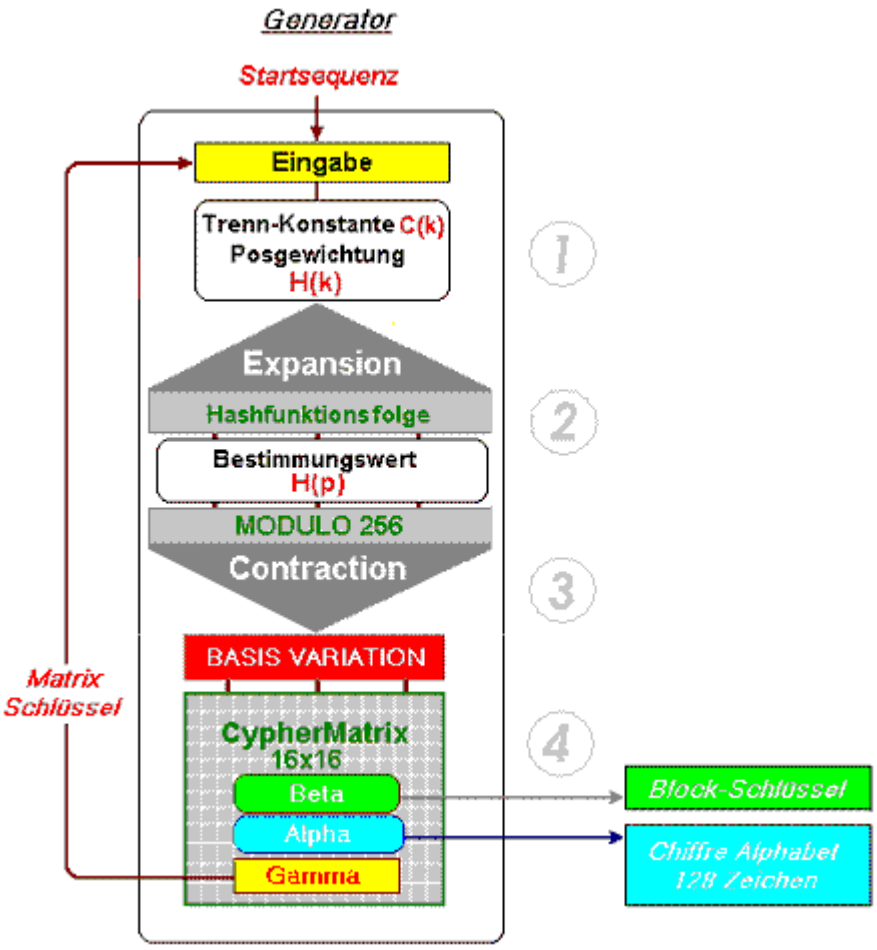
3.1 Der Generator

Das Verfahren ist als Block-Chiffre gestaltet. Es ist symmetrisch, weil Sender und Empfänger zur Initialisierung des Generators die gleiche Startsequenz eingeben müssen und es ist dynamisch, weil der Generator für jeden Klartextblock (zB 63 Bytes) in jeder Runde neue Steuerungsparameter erzeugt. 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]
Leonardo eroberte Florenz mit Schneekanonen	[43 Bytes]
7 kangaroos jumping along the Times Square	[43 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.

Jede Startsequenz erzeugt sowohl beim Sender als auch beim Empfänger einen identischen Ablauf des Verfahrens und identische Ergebnisse.



In jedem Durchlauf erzeugt der Generator die zur Verschlüsselung erforderlichen Runden-Parameter:

1. Das Chiffre-Alphabet (System-Alphabet) für die aktuelle Runde,
2. den Blockschlüssel für die XOR-Verknüpfung und
3. einen Matrix-Schlüssel als Startsequenz für die nächste Runde.

Den Abschluss jeder Runde bildet eine **CypherMatrix** mit 16x16 Elementen, die alle Steuerungsparameter für die Verschlüsselung zur Verfügung stellt. Der Matrix-Schlüssel (42 Zeichen) wird auf den Anfang der Funktion zurückgeführt. 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.

3.1.1 Eingabe der Startsequenz

Als Beispiel wird die folgende Startsequenz (Eingabe) gewählt:

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

Es gilt eine eindeutige Abbildung der Eingabe als Bestimmungsbasis für die Analyse 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):

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

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

$$\mathbf{m} = \sum_{i=1}^{n} (a_i + 1)$$

$$\mathbf{m} = 4066$$

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

3.1.2 Erweiterung zur Positionsgewichtung

Mit Besinnung auf **Renè Descartes** (1596 – 1650) wissen wir, dass jeder Sachverhalt, soweit er in seinen Dimensionen skalierbar ist, durch seine Koordinaten für **Gegenstand, Ort** und **Zeit** (analog kartesischem Koordinatensystem) eindeutig bestimmt werden kann. 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 dann von Bedeutung, wenn zwischen den einzelnen Bytes und der Prozessorfrequenz eine variable Funktion besteht, ansonsten: **t** (**i**) = **1**. Um einen bestimmten Wert für die Folge **m** zu erhalten, werden die Dimensionswerte für Gegenstand, Ort und Zeit durch Multiplikation verknüpft und zum Zwischenwert **H**(**k**) addiert.

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

$$H(k) = 88446$$

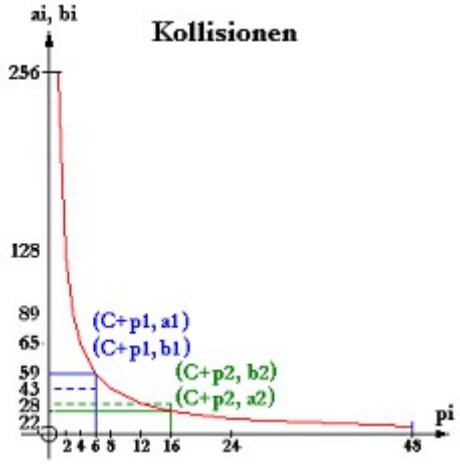
Mit der **Positionsgewichtung** unterscheiden sich zwar die Bytes **a**(**i**)**,** aber Kollisionen als Folge des Austausches von Bytes innerhalb der Zeichenfolge sind noch nicht ausgeschlossen.

3.1.3 Ausschluss von Kollisionen

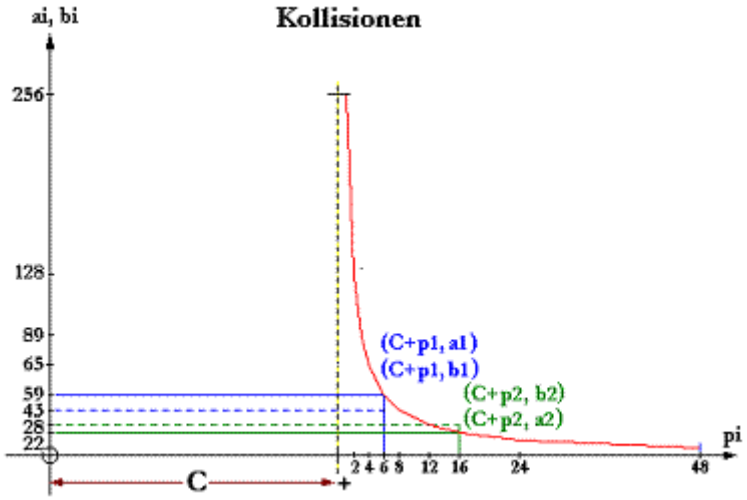
Eine Kollision entsteht unter folgenden Bedingungen:

$$\text{Kollision:} \quad H(k)\, a_i = H(k)\, b_i$$
$$(a_1 + 1) * p_1 + (a_2 + 1) * p_2 = (b_1 + 1) * p_1 + (b_2 + 1) * p_2$$

An der Stelle p_1 wird das Zeichen a_1 mit dem Zeichen b_1 und an der Position p_2 das Zeichen a_2 mit b_2 ausgetauscht. Die folgende Kurve zeigt die Zusammenhänge zwischen den einzelnen Zeichen a_i und p_i im Produkt $(a_i + 1) * p_i$:



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



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

Nach Umformung ergibt sich der folgende Veränderungsquotient: **Q**

$$\mathbf{Q} = \frac{(\mathbf{C} + \mathbf{p}_1)}{(\mathbf{C} + \mathbf{p}_2)} = \frac{(\mathbf{b}_2 - \mathbf{a}_2)}{(\mathbf{a}_1 - \mathbf{b}_1)}$$

Für den „Veränderungsquotienten“ - hier mit **Q** bezeichent – sind drei Fälle möglich:

Q > 1
Q = 1
Q < 1

Wenn **Q = 1,** dann müssen auch (**C** + $\mathbf{p}_1$) und (**C** + $\mathbf{p}_2$) gleich sein. Da der Austausch an derselben Position **p** geschieht, ist hier eine Kollision ausgeschlossen. Ist **Q > 1** oder **Q < 1,** dann sind auch ($\mathbf{b}_2$ **-** $\mathbf{a}_2$) und ($\mathbf{a}_1$**-** $\mathbf{b}_1$) verschieden. Da die Werte ($\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{b}_1$ und $\mathbf{b}_2$) Integerwerte sind, sind auch deren Differenzen ganzzahlig.

Die Positionen $\mathbf{p}_i$ in der Eingabesequenz mit **N** Bytes (Länge **n**) umfassen einen Bereich von **1** (*minimum*) bis **N** (*maximum*). Der Veränderungsquotient nach der obigen Formel erfasst daher die folgende Spanne:

$$\frac{\mathbf{C} + \mathbf{N}}{\mathbf{C} + \mathbf{1}} \quad \} \; \mathbf{Q} \; \{ \quad \frac{\mathbf{N}}{\mathbf{N} - \mathbf{1}}$$

Die weitere Entwicklung ist im Artikel

"Bestimmungsfaktoren für Kollisionsfreiheit"

ausführlich dargelegt. Das Ergebnis führt zur Formel: **C = N * (N – 2)**

Der Faktor **C** ist allein von der Länge **N** der Eingabesequenz abhängig. Er hat außerdem die Eigenschaften, 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**(**k**).

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

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

Nach Einbindung der Trenn-Konstante **C(k)** wird der Zwischenwert $\mathbf{H_k}$ wie folgt ermittelt:

$$\mathbf{H_k} = \sum_{i=1}^{n} \mathbf{(a_i + 1) * (p_i + C_k + Runde)}$$
$$\mathbf{H_k = } 6927458$$

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.

3.1.4 Erweiterung zur Hashfunktionsfolge

Zur Erweiterung der Bestimmungsbasis wird die **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. Für jedes Zeichen der Eingabeseqzenz errechnet das Verfahren den dezimalen Wert ($\mathbf{s_i}$), 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).

$$\mathbf{s_i = (a_i + 1) * p_i * H_k + p_i + code + Runde}$$
$$\mathbf{s_i \rightarrow d_i} \text{ (base 77)}$$

$$\mathbf{HF = d_1 + d_2 + d_3 + \ldots + d_i + \ldots d_m}$$
(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)

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

char	pi (ai+1)		Hk (ai+1)*pi		(ai+1)*pi*Hk	Si pi+code+r		Basis 77
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
						Summe:	612705951255	2çRprTD

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

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(k):	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 MOD 11) +1	=	11	Beginn der Kontraktion
Alpha	((H_k + H_p) MOD 255) +1	=	204	Offset Chiffre-Alphabet
Beta	(H_k MOD 169) +1	=	149	Offset Block-Schlüssel
Gamma	((H_p + code) MOD 196) +1	=	141	Offset Matrix-Schlüssel
Delta	((H_k + H_p) MOD 155) +code	=	44	dynamische Bitfolgen
Theta	(H_k MOD 32) +1	=	3	Offset Rückrechnung
Omega	(H_k MOD 95) +1	=	59	Beginn Doppelzeichen
Kappa	(H_k MOD 16384)+1	=	7704	Beginn Chiffre-Alphabet

Die Steuerungsparameter dienen zur Lösung verschiedener kryptographischer Aufgaben.

3.1.5 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

3.1.6 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 werden die Indexwerte in einem gesonderten Array **IndexFolge(2,16)** aus den Elementen der BASIS-VARIATION neu generiert (Variante D).

3.1.7 Die CypherMatrix

Nach allem ergibt sich die endgültige CypherMatrix wie folgt:

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

3.1.8 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** =ASCII-26) und die ordnungsgemäße Durchführung des Programms stören.

Chiffre-Alphabet (Basis 7)

1	ì ¾ | B º € Ó ¨ a “ Ê ú … Ä ¤ >	*16*
17	ô @ s / õ — › c d Œ □ 8 *ƒ* ü x •	*32*
33	g h i □ 6 ¦ z 7 [F „ µ – Ø ž	*48*
49	· œ † G ¡ © » ª % « 9 ¬ A : ¯ &	*64*
65	; ¼ < D H ¿ U ï I À Á J ã Ù æ O	*80*
81	P Â Ã Å Æ Ë Ö × é ê ë t W ö Í]	*96*
97	0 à C j á N 2 ~ Ô ` Q þ ÷ î k Î	*112*
113	X I = ð * ¬ ¸ □ ” Ÿ E u ¥ É ¢	*128*

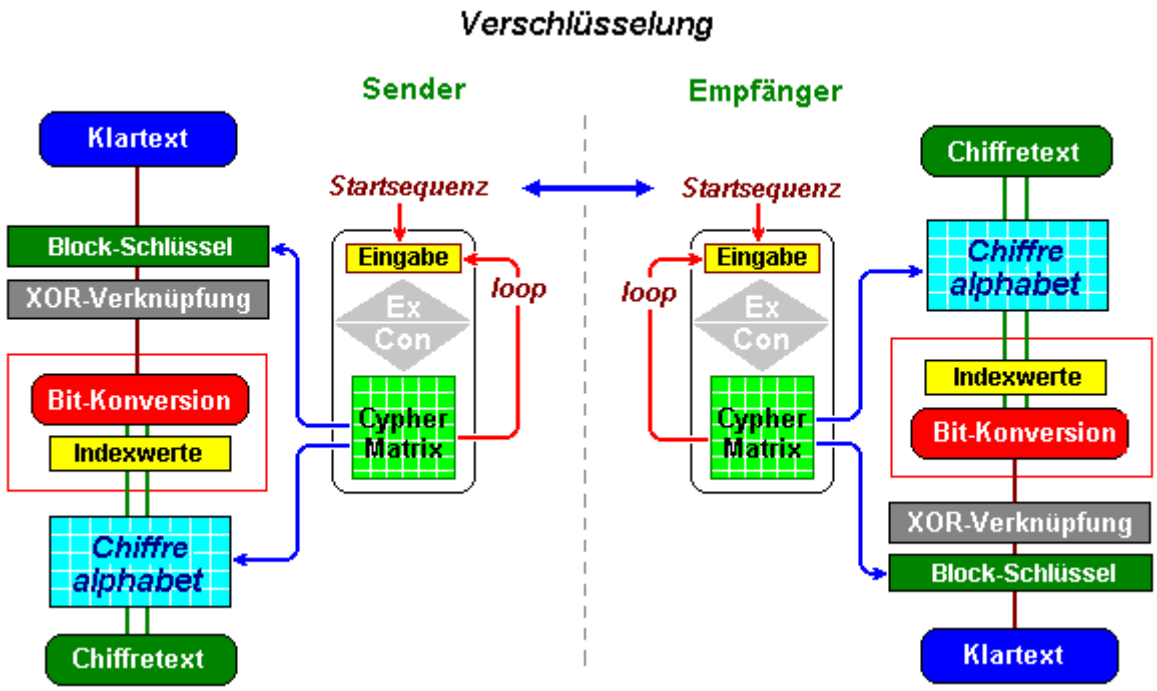
Chiffre-Alphabet (hex)

1	EC	BE	7C	42	BA	80	D3	A8	61	93	CA	FA	85	C4	A4	3E	*16*
17	F4	40	73	2F	F5	97	9B	63	64	8C	81	38	83	FC	78	95	*32*
33	67	68	69	9D	36	A0	A6	7A	37	5B	46	84	B5	96	D8	9E	*48*
49	B7	9C	86	47	A1	A9	BB	AA	25	AB	39	AC	41	3A	AF	26	*64*
65	3B	BC	3C	44	48	BF	55	EF	49	C0	C1	4A	E3	D9	E6	4F	*80*
81	50	C2	C3	C5	C6	CB	D6	D7	E9	EA	EB	74	57	F6	CD	5D	*96*
97	30	E0	43	6A	E1	4E	32	7E	D4	60	51	FE	F7	EE	6B	CE	*112*
113	58	6C	3D	F0	2A	AD	B8	8D	94	9F	FF	45	75	A5	C9	A2	*128*

4 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:

Basis-Coding: Bit-Konversion allein ohne weitere Operationen oder
Verbund-Coding: Bit-Konversion mit zusätzlichen Operationen, und zwar:

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

4.1 Basis-Coding

Im „Basis-Coding“ Modus erfolgt die Bit-Konversion direkt vom bisherigen Bitsystem zum angestrebten Bitsystem. Zum Beispiel von Basis 8 nach Basis 7:

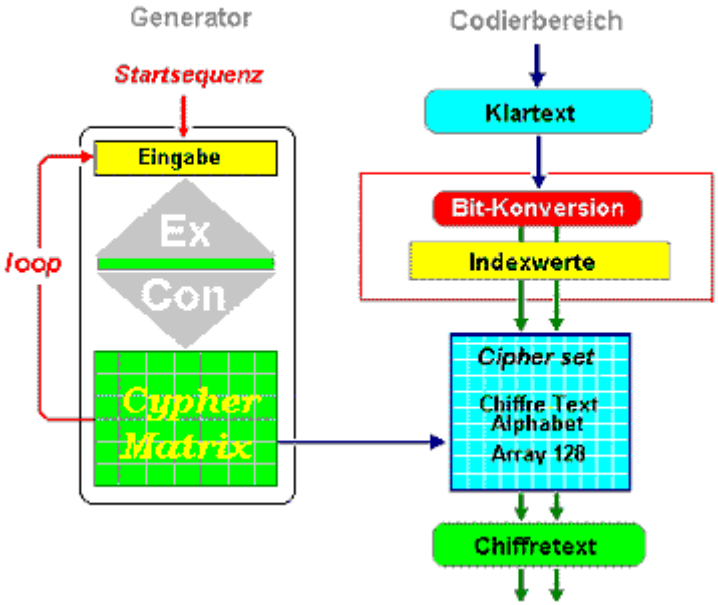
Klartext

N	o	t	a	b	e	n	e	
01001110	01101111	01110100	01100001	01100010	01100101	01101110	01100101	
0100111	0011011	1101110	1000110	0001011	0001001	1001010	1101110	0110010	1
39	27	110	70	11	9	74	110	50	

Indexe

Die Umwandlung vom Klartext zum codierten Text geschieht mit zwei Funktionen:

1. Bit-Konversion
   8-bit Klartextwerte → 7 bit Indexwerte (0 ...127)
2. Bestimmung des Chiffretexts
   7-bit Indexwerte → Chiffre-Alphabet (0...127) → Chiffretext.



Als Beispiel für **Basis-Coding** wird ein Gedicht von Matthias Claudius verschlüsselt mit dem Programm: **System12.exe** und der Start-Sequenz: „Die weiße Elster fließt in das schwarze Meer“ (Datei: **Claudius.txt** / Umfang: 215 Bytes).

Der Mond ist aufgegangen,
Die goldnen Sternlein prangen
Am Himmel hell und klar;
Der Wald steht schwarz und schweiget,
und aus den Wiesen steiget
Der weiße Nebel wunderbar.
Matthias Claudius, Reinfeld 1812

Als ersten Block des zu verschlüsselnden Klartextes werden 36 Bytes eingegeben:

Der Mond ist aufgegangen,♪¤Die goldn

44 65 72 20 4D 6F 6E 64 20 69 73 74 20 61 75 66 67 65 67 61 6E 67 65 6E 2C
0D 0A 44 69 65 20 67 6F 6C 64 6E

Der Klartext im Bitsystem zur **Basis 8** (36x8=288) wird in Abschnitte des angestrebten Bitsystems zur **Basis 12** (288:12=24) umgewandelt.

Basis 8: D e r M o n d
01000100 01100101 01110010 00100000 01001101 01101111 01101110 01100100 . . . .
Basis 12:
010001000110 010101110010 001000000100 110101101111 011011100110 0100 . . . .

Index:	1094	1394	516	3439	1766
(+1)	1095	1395	517	3440	1767

System-Alphabet Basis 12:

Alphabet$(517) = ‰s
Alphabet$(1095) = ì¤
Alphabet$(1395) = Éa
Alphabet$(1767) = “O
Alphabet$(3440) = X

Chiffretext: ì¤ Éa ‰s X “O

Der Chiffretext lautet wie folgt:

ì¤Éa‰s X“Oì|’èŒc‰uˆd’O”N’eˆW’e‰WŠžŸyì¤˜N‰u”X’¤ŽWx–êqvêt¥zwë0xá2pRm6–
xgê§x–êXmuo1pO#ªy–è¥y,l©xç0äWë7ì‡á4í‹ãBçè‡Zè‹ëBäUî¦í‹èvîïè«íWèvîîç1ïFãBïx–
áEVnP—W1OpWàSqUàS#NBh□WXklL{K@VªIlUXj—SàamW1SvLO#ejk&ekZ□S§a|
yf¥SíejluefSbjWyfPSígjuxekWvdšW5TdI□Gkf—T{NžSWHªJERXPšSXP . . . . . .

8D A4 90 61 89 73 A0 58 93 4F 8D 7C 92 E8 8C 63 89 75 88 64 92 4F
94 4E 92 65 88 57 92 65 90 57 8A 9E 9F 79 8D A4 98 4E 89 75 94 58
92 A4 8E 57 78 96 EA 71 76 EA 74 A5 7A 77 EB 30 78 96 E1 32 70 52
6D 36 78 67 EA A7 78 96 EA 58 6D 75 6F 31 70 4F 23 AA 79 96 E8 A5
79 82 6C A9 78 96 E7 30 E4 57 EB 37 EC 87 E1 34 ED 8B E3 42 E7 E8
. . . . . .

Infolge der Bit-Konversion entfallen auf 12 Zeichen im Bitsystem zur Basis 8 insgesamt 96 Bits, die dann auf 8 Zeichen im Bitsystem zur Basis 12 verteilt werden. Insoweit ergibt sich wegen der Doppelzeichen ein Längenverhältnis von 1:1,333 .

Für das System-Alphabet zur Basis 12 werden 4096 Zeichen benötigt, die als Einzelzeichen nicht mehr zur Verfügung stehen. Sie werden als Doppelzeichen mit Hilfe der Ziffern des **Zahlensystems** zur **Basis 128** – das sind 16384 Ziffern - generiert:

Im Quellcode:

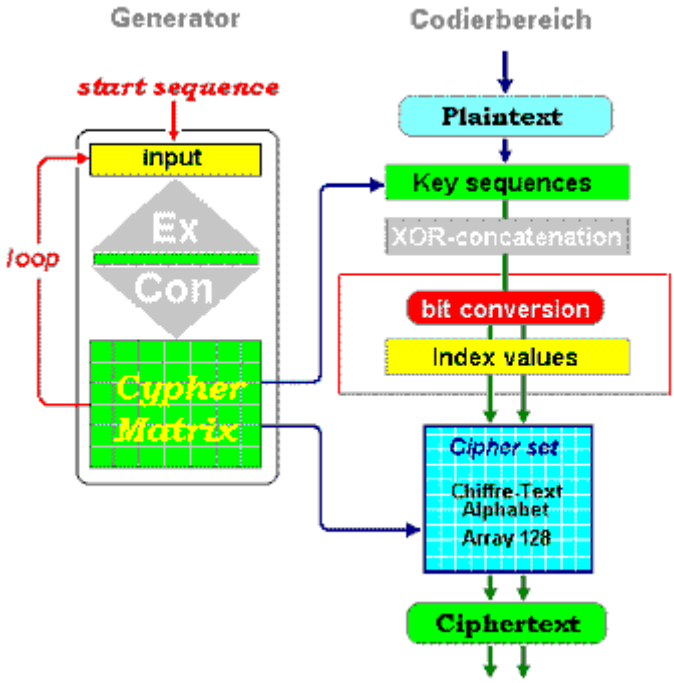
```
SUB Alphabet
      SHARED Alphabet$(),  Kappa
       Kappa = 11441, 6035, 8344, 2250  (in jeder Runde neu berechnet)

       FOR C=1 TO 4096
         X## = C + Kappa
         CALL DezNachSystem (128, X##, Zeichen$)
         Digit$ = „00“+Zeichen$
         Digit$ = RIGHT$(Digit$,2)
         Alphabet$(C) = Digit$
      NEXT C
END SUB
```

4.2 „Verbund-Coding“

Beim Verbund-Coding werden verschiedene Operationen seriell verbunden, z.B. XOR-Verknüpfung 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 → Block-Schlüssel → 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.



Im Quellcode werden die Module nacheinander durchlaufen:

```
. . . . . .
CALL  XORGenerator (InputData$,TransData$)          für „XOR“
CALL BitConversion (TransData$,DataTrans$)          für „Bit-Konversion“
CALL Substitution (DataTrans$,ReportData$)          für „dyn24“
CALL ByteWechsel (ReportData$,OutputData$)          für „exchange“
. . . . . .
```

4.2.1 Verschlüsselung von Basis 8 nach Basis 7

Als Beispiel werden die Verschlüsselungsschritte im Programm **DataCode.exe** und der Startsequenz: „**Im Elbsandsteingebirge gibt es keinen Sand**“ für die Datei **„Garten.txt“** erläutert (573 Bytes) .

> *Ein Steingarten ist eine Zierde für jeden Garten. Schon allein die Steine, aus denen sich eine solche Anlage aufbaut -ob es sich nun um Kalksteine, Sandsteine, Schiefer, Granit, Basalt, Tuffsteine oder auch nur um große Kieselsteine handelt - verleihen dem Garten einen besonderen Akzent. Man kann sie im rohen Zustand verwenden, gerade so, wie man sie vielleicht auf einer Wanderung durch bergige oder gebirgige Gegenden findet, doch eine ebenso große Wirkung kann man natürlich mit behauenen Steinen erzielen.*
>
> Steingartenpflanzen, ADAC-Ratgeber, Stuttgart 1961

In jeder Runde wird ein Klartext-Block von 42 Bytes (42x8 = 336 Bits) mit einem Block-Schlüssel von gleicher Länge XOR-Funktion verknüpft. Als ersten Klartext-Block liest das Programm folgende 42 Bytes ein:

Ein Steingarten ist eine Zierde für jeden□ (42 Bytes)

45 69 6E 20 53 74 65 69 6E 67 61 72 74 65 6E 20 69 73 74 20 65
69 6E 65 20 5A 69 65 72 64 65 20 66 81 72 20 6A 65 64 65 6E 20

Der ab Position **158** der CypherMatrix entnommene Block-Schlüssel umfasst:

yu5ˆÉDW"È¹´GÐ‡Á½H,(#„Ìsí^Ëº¿ö-q<ÊÅ#ÂðþØ¥¤+

79 75 35 88 C9 44 57 22 C8 B9 B4 47 D0 87 C1 BD 48 2C 28 03 84
CC 73 ED 5E CB BA BF F6 2D 71 3C CA 8F 15 C2 F0 FE D8 A5 A4 2B

XOR-Verknüpfung:
Klartext:
01000101 01101001 01101110 00100000 01010011 01110100 01100101 01101001 . . . .
Schlüssel:
01111001 01110101 00110101 10001000 11001001 01000100 01010111 00100010 . . . .
XOR:
00111100 00011100 01011011 10101000 10011010 00110000 00110010 01001011 . . . .

Hex.	3C	1C	5B	A8	9A	30	32	4B
Dez:	60	28	91	168	154	48	50	75
ASCII:	<	#	[	¨	š	0	2	K

<#[¨š02K¦ÞÕ5¤â¯¥!_\#á¥#ˆ~‘ÓÚ„l##¬#gâš›¼ÀÊ&#=

3C 1C 5B A8 9A 30 32 4B A6 DE D5 35 A4 E2 AF 9D 21 5F 5C 23
E1 A5 1D 88 7E 91 D3 DA 84 49 14 1C AC 0E 67 E2 9A 9B BC C0

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.

4.2.2 Bit-Konversion

Das Ergebnis der XOR-Verknüpfung im Bitsystem zur Basis 8 (42x8 = 336 Bits) wird in Zeichen des Bitsystems zur Basis 7 (336 Bits = 48x7) 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:
XOR Basis 8:
00111100 00011100 01011011 10101000 10011010 00110000 00110010 01001011
Basis 7:
0011110 0000111 0001011 0111010 1000100 1101000 1100000 0110010 0100101 1

Index:	30	7	11	58	68	104	96	50	37
(+1)	31	8	12	59	69	105	97	51	38
Alphabet Basis 7:									
	k	T	?	q	+	•	Š	s	W

System-Alphabet zur Basis 7

1	ƒ™êš¢U¦Tnñ?¡‰h9˜Q>MLié8K7Ï€èYky	32
33	u5ˆÉDWÈ¹´GÐ‡Á½H(„Ìsí^Ëº¿ö-q<Ê¥Âð	64
65	þØ¥¤+[:ã#{·CåS/ª‘Bx@ä3À§)0«deV!Ÿ	96
97	Š‹Œıℓ˛Î¨c³&• ž¯£—l›®.’%$ýÄ\OÔ]ûÆü	128

System-Alphabet (hex)

```
83 99 EA 9A A2 55 A6 54 7D 6E F1 3F A1 89 68 39
98 51 3E 4D 4C 69 E9 38 4B 37 CF 80 E8 59 6B 79
75 35 88 C9 44 57 C8 B9 B4 47 D0 87 C1 BD 48 28
84 CC 73 ED 5E CB BA BF F6 2D 71 3C CA 8F C2 F0
FE D8 A5 A4 2B 5B 3A E3 23 7B B7 43 E5 53 2F AA
91 42 78 40 E4 33 C0 A7 29 30 AB 64 65 56 21 9F
8A 8B 8C 8D A8 63 B3 26 95 A0 9E AF A3 97 6C 9B
AE 2E 92 AD 7F 25 24 FD C4 5C 4F D4 5D FB C6 FC
```

Als Ergebnis der Bit-Konversion entsteht der Chiffretext der Datei **Garten.ctx** (672 Zeichen) wie folgt:

kT?q+•ŠsW d—GÀ{Œ§&DiO®ã‹xãÌTÿã¹«¥>ˆØc„è&.È@<³šLèfÅRó~÷([GI½hO‡@~?R E`((sMGfÉw`¢¤âúþzÈÅv9ú2[úíWn½~ýe~«□å”½YY¼ýÀþK”gO¾À“<¶æÏ¼×•-”ÌI}r½«öC-A‘˜8¯='Gd'ý¿‰MD·Cm`.dš§záÓÖ(¾Sæ¢’óó+F¦¿¹gôæ:F \óž¾zBéÈÓ7<îÏfïwîOnÉ½E×Æ ÷›Ïûî¼Ð´<”ÏÅ¸□Ï³g6@5®HV N”D…ªb=GÎc·yÀSÆÃK1KO6tî¤—«†À}¼1Ép6†ÏÃáŸTàîLêI –©yT&“&0<ýHî“ö†«œÀ-Åó¬«4¿(ñmHôní™ŠòôYŽ12÷qslqm¾jØÃôr–Gii¬sÀaÈP'—ÔËP —ê}Ç]Ôîzµ¢□úoA‡ãV]¸m$Ë©*üêu<cK’Ì¼ù§hœÈVSÎB)ÒZÒîežú«GM+BsVL&?úŒd”«å! Ú•Ãe¤ÈâB{û³®ãæ□ú¦eª©z+Çµ¢º¬XHjw*f*¶-Z*5k§“¶Kô5™F»áQ\2Ì¦†À†7´E™ÍOûXl-´EE ™.ÍOûXl-.GF\v1=´Ò*ÊEMðì¯÷æ>h>XECãŠ†ì2©JÃê(ìHæ‰*º´+%Á´ácZ… <*ƒ*MNüœº` ÓbY…kÀ¤kxc|‡|þŽ‹@1dø)§/àÀõÓ¿zAn}^™„s zÉH¤=þë€*Ø7áHáÇ¼m¼KAK¡%g’ðÚ ! 7áHáÇ¼m¼KAK¡%g’ðÚ–’ðÚÇ@Têú´§ëwgP›□`džsÈ□aö¡³gc□ú.“>£ZŽ[m‰_GZÔfî ‰_GZÔf5ï6ÌàòÂM;ï□ü»7õê(þWÇîÀ□à6ÂØø

5 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 → Block-Schlüssel → 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 Block-Schlüssel XOR-verknüpft. Als Ergebnis erscheint der ursprüngliche Klartext.

Als Beispiel die Entschlüsselung des Chiffretextes **Garten.ctx:**
Das Programm liest als ersten Chiffretext-Block folgende 48 Zeichen (Basis 7) ein:

kT?q+•ŠsW d—GÀ{Œ§&DiO®ã‹xãÌTÿã¹«¥>ˆØc„è&.È@<³šLèfÅRó~÷([GI½hO‡@~?

Index:	31	8	12	59	69	105	97	51	38
(-1)	30	7	11	58	68	104	96	50	37

Basis 7:
0011110 0000111 0001011 0111010 1000100 1101000 1100000 0110010 0100101 1 . . .
Bit-Konversion nach Basis 8:
00111100 00011100 01011011 10101000 10011010 00110000 00110010 01001011 . . . .
01111001 01110101 00110101 10001000 11001001 01000100 01010111 00100010 . . . .
XOR-Verknüpfung mit Schlüssel:
01000101 01101001 01101110 00100000 01010011 01110100 01100101 01101001 . . . .
Index Basis 8:

	69	105	110	32	83	116	101	105
Zeichen:	E	i	n		S	t	e	i

Als Klartext ergibt sich: **Ein Stei**ngarten ist eine Zierde für jeden□ . . . . . .

6 Kryptanalyse

Die Kryptanalyse umfasst bekanntlich alle Versuche, aus dem Chiffretext irgendwelche Rückschlüsse auf den Klartext zu gewinnen. Zu den Auffälligkeiten der Sprache zählen Wiederholungsmuster und Wortkombinationen, Häufigkeitsstrukturen und Bigramme [#4].
Diese Sachverhalte setzen voraus, dass Klartext und Chiffretext zueinander im Verhältnis 1:1 stehen. Zu den bekanntesten Angriffen gehören Strukturanalyse, "known plaintext attack", "chosen plaintext attack" und "brute force attack", eventuell noch "differenzielle" und "lineare" Kryptoanalyse. Mit diesen Angriffen sollen aus dem Chiffretext statistisch erfassbare Regelmäßigkeiten herausgefiltert werden, die mölicherweise einen Weg zum Klartext aufzeigen. Eine Analyse anhand dieser Merkmale, setzt allerdings voraus, dass Klartext und Chiffretext Strukturen enthalten, die sich auch vergleichen lassen.
Daher muss ein einheitliches **Ordnungssystem** bestehen, das sowohl im Klartext als auch im Chiffretext wirksam ist.

6.1 Aktuelle Verfahren

Bei fast allen herkömmlichen Verfahren haben Klartext und Chiffretext die gleiche Länge: **"Längenkongruenz"**. Daraus folgt, dass für jedes Klartextzeichen auch ein bestimmtes Chiffrezeichen vorhanden sein muss. Beide Bereiche - Eingaben und Ausgaben - arbeiten im selben System-Alphabet (ASCII-Zeichensatz), und damit auch im gleichen Bitsystem zur **Basis 8.** Mithin besteht auch ein einheitliches Ordnungssystem.
Ein Systemwechsel findet nicht statt. Eigenheiten im Klartext müssen sich in irgendeiner Form auch im Chiffretext wiederfinden lassen.

6.2 „CypherMatrix“ Verfahren

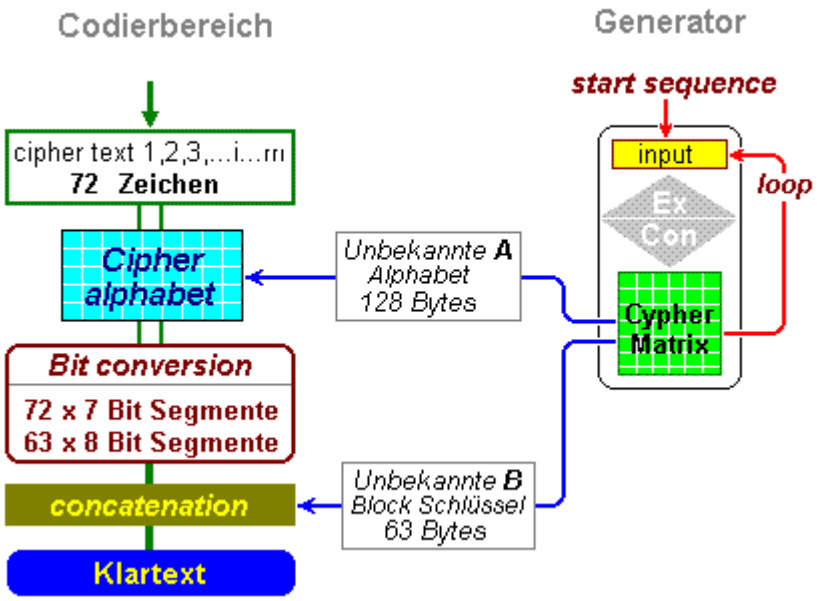
Diese Bedingungen sind im CypherMatrix Verfahren nicht erfüllt. Infolge **Bit-Konversion** entfällt das einheitliche Ordnungssystem zwischen Klartext und Chiffretext mit der Folge, dass beide Bereiche sich nicht mehr vergleichen lassen. Es fehlt die Längenkongruenz. Durch die Umwandlung der Zeichen vom 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 verursachende Klartextzeichen. Außerdem enthält das Chiffretext-Alphabet nur 128 Zeichen, während das Klartext-Alphabet 256 Zeichen umfasst.

Eine Kryptanalse anhand bestimmter Merkmale, wie beispielsweise Wiederholungsmuster, Häufigkeitsstrukturen, Bigramme, differenzielle und lineare Verfahren - setzt allerdings voraus, dass Klartext und Chiffretext zueinander im Verhältnis 1:1 stehen. Die hierauf gründenden Angriffe können mithin auch keinen Erfolg mehr versprechen. Es fehlt ein einheitliches Ordnungssystem für beide Bereiche und damit auch die Basis für alle oben genannten Angriffszenarien. Sie sind **wirkungslos** und wir können sie vergessen. Um dies zu testen, können Sie sich einige mit CypherMatrix Programmen verschlüsselte Nachrichten als ZIP-Datei auf Ihren Rechner holen und versuchen, die Chiffre mit heute gängigen Angriffen zu brechen.

6.3 Sicherheit des Verfahrens

Immerhin bleibt noch die Möglichkeit einer „brut force“ Attacke. 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. Er könnte also nur eine Iteration aller Möglichkeiten versuchen. Bei einem Angriff auf die Startsequenz mit 42 Bytes Länge ergibt sich eine Entropie von **336** und eine exponentielle Komplexität von **O(2^336)** = 1.4E+101. Ein Angreifer kann dann versuchen vom Chiffretext ausgehend, retrograd durch Iteration die einzelnen Stufen der Entschlüsselung herauszufinden.

Die Entschlüsselung geschieht in jedem Durchgang wie folgt:



Das Verfahren enthält drei Funktionen:

1. Klartextblock --> **Block-Schlüssel** --> -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 **Block-Schlüssel** 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 = **Block-Schlüssel**
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 Block-Schlüssel = 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).

Es gibt somit viele Paare **Chiffre-Alphabet** / **Block-Schlüssel**, die nach einem versuchten "brute force" Angriff irgendwelche lesbaren Texte liefern, von denen man aber nicht weiß, welcher der Richtige ist: Angriff mit "brute force" . Somit kann auch „brute force“ keinen Erfolg haben.

7 Zusammenfassung

Die vorstehenden Erläuterungen lassen einige alternative Grundsätze erkennen:

1. Ein einheitliches Ordnungssystem (Struktur der Bitfolgen und System-Alphabete) ist bisher nicht definiert,
2. Die Darstellung der Kryptographie beschränkt sich auf Verschlüsselungen im Bitsystem zur Basis 8 (Eingaben und Ausgaben) und
3. Fast alle Angriffszenarien setzen für Klartext und Chiffretext ein vergleichbares einheitliches Ordnungsystem voraus (Bitsystem zur Basis 8 und System-Alphabet mit 256 Zeichen).

Weitere Einzelheiten zum **CypherMatrix** Verfahren unter: www.telecypher.net/.
Wer sich intensiver mit dem Verfahren beschäftigen möchte, kann einzelne Programme mit oder ohne Source-Code beim Autor per e-mail anfordern und unter der CMLizenz damit arbeiten (eschnoor@multi-matrix.de).

München, im September 2013

Hinweise

[#1] Schneier, Bruce, Angewandte Kryptographie (dt. Ausgabe), Bonn … 1996, S.229
[#2] Schmeh, Klaus, Safer Net, Kryptografie im Internet und Intranet, Heidelberg 1998, S.61,
[#3] Morin, Charles, www-lehre.inf.uos.de/~rspier/referat/feist.html
[#4] Bauer, Friedrich L., Entzifferte Geheimnisse – Methoden und Maximen der Krytologie, Berlin Heidelberg New York, 1995, S.186 ff.