

„CypherMatrix“

Ein neues Verfahren in der Kryptographie

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

Das „CypherMatrix“ Verfahren zeigt einen neuen Weg, digital gespeicherte Informationen zu analysieren und systematisch zu bearbeiten. Das Verfahren kann in fast allen Bereichen der Informationstechnik eingesetzt werden. Die Basisfunktion liefert die erforderlichen Steuerungsparameter, während die kryptographischen Probleme in eigenständige Bereiche ausgegliedert werden. Mit dem Verfahren lassen sich viele Probleme lösen, insbesondere:

Durchführung von Verschlüsselungen,
Berechnung von Hashwerten,
einfache und erweiterte Signaturen,
Generierung unbegrenzter Byte- und Zahlenfolgen und
weitere noch zu erforschende Zusammenhänge.

Das Verfahren beginnt mit der Eingabe einer digitalen Information als **Startsequenz** und errechnet als Ergebnis eine eindeutige Abbildung der Eingabe in Form einer **Cypher Matrix** mit 16x16 Zeichen. Die CypherMatrix liefert die zur Lösung der kryptographischen Aufgabe erforderlichen Parameter. Nach den Gesetzen der Wahrscheinlichkeit entsteht die Wiederholung einer gleichen CypherMatrix erst nach $256!$ (Fakultät) = $8E+506$ Fällen. Die Zusammenhänge werden am Beispiel von Verschlüsselungen nach dem CypherMatrix Verfahren (Bezeichnung vom Autor).

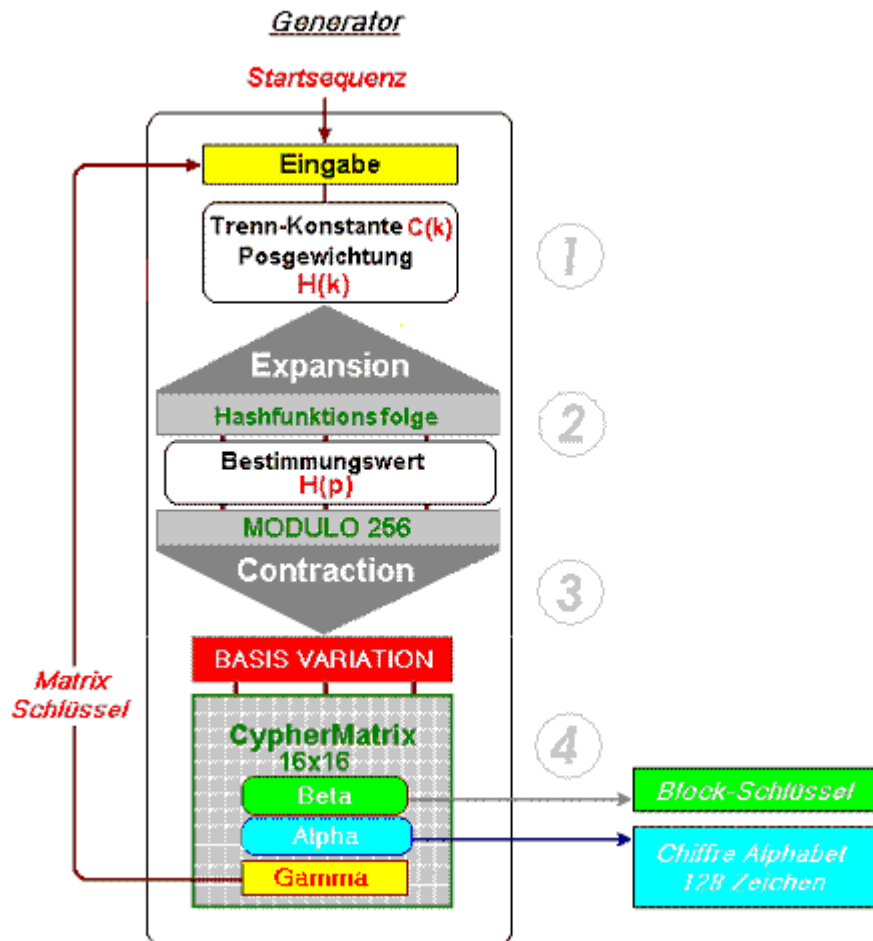
Im Gegensatz zu den heute üblicherweise verwendeten Algorithmen gehen CypherMatrix Verschlüsselungen eigene Wege. Die Verschlüsselung wird in zwei Bereiche aufgeteilt: **Basisfunktion** (Generator) und **Codierbereich**. Beide Bereiche werden kombiniert, können aber auch getrennt verwendet werden. Aufgabe der Basisfunktion (Generator) ist die Bereitstellung der zur Verschlüsselung erforderlichen Steuerungsparameter. Die eigentliche Verschlüsselung vollzieht sich im Codierbereich.

A. Die Basisfunktion

Das Verfahren ist dynamisch, weil die Basisfunktion in jeder Runde eine neue CypherMatrix erzeugt und neue Steuerungsparameter berechnet. Das Verfahren ist symmetrisch, weil Sender und Empfänger zur Initialisierung der Funktion die gleiche Startsequenz eingeben müssen. Die Startsequenz (Passphrase) mit optimal 42 Zeichen steuert das gesamte Verfahren. Einige Beispiele:

Die Schildbürger fegen den Teutoburgerwald	[42 Bytes]
Der rote Main fließt in den blauen Pazifik	[42 Bytes]
Auf der Fischbachalm gibt es keine Heringe	[42 Bytes]

Die Startsequenz (Eingabe) 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 die Startsequenz weder durch Iteration noch durch einen Wörterbuchangriff 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. Die folgende Skizze veranschaulicht die Zusammenhänge:



1 Eingabe der Startsequenz

Als Beispiel wird die Startsequenz (Eingabe) gewählt mit:

"7 Nordlichter wandern über den Großen Belt" (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.

$$\mathbf{m} = \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3 + \dots + \mathbf{a}_i + \dots + \mathbf{a}_n$$

(Der einzelne Wert für "a;" wird um (+1) erhöht da sonst **a=0** nicht berücksichtigt wird)

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

$$\mathbf{m} = 4066$$

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

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 kartesisches Koordinatensystem) eindeutig bestimmt werden kann. Die Skalierung erfasst den Gegenstand, den Ort und die Zeit der digitalen Zeichenfolge. 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**. Die Dimensionswerte für Gegenstand, Ort und Zeit werden durch Multiplikation verknüpft und zum Zwischenwert **H(k)** addiert.

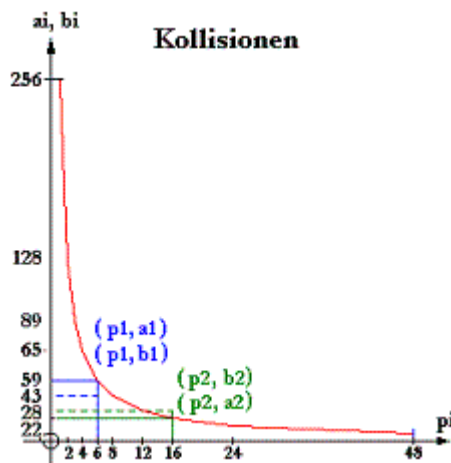
$$H(k) = \sum_{i=1}^n (a_i + 1) * p_i * t_i \quad t_i = 1$$

$$H(k) = 89247$$

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 Ausschluss von Kollisionen

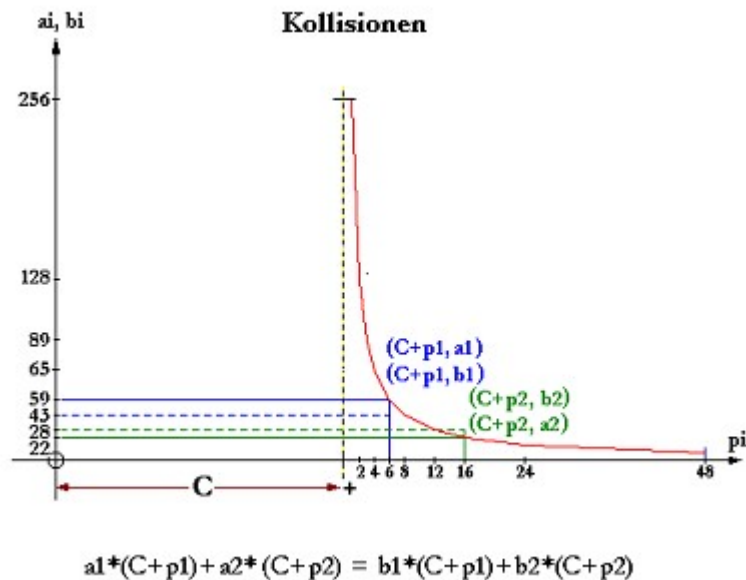
Eine Kollision entsteht, wenn an der Stelle **p₁** das Zeichen **a₁** mit dem Zeichen **b₁** und an der Position **p₂** das Zeichen **a₂** mit **b₂** ausgetauscht werden. Die folgende Kurve zeigt die Zusammenhänge zwischen den einzelnen Zeichen **a_i** und **p_i** im Produkt **(a_i+1) * p_i**. [Sch07a]:



Kollision: $H(k) a_i = H(k) b_i$
 $(a_1+1) * p_1 + (a_2+1) * p_2 = (b_1+1) * p_1 + (b_2+1) * p_2$

Die Summe der neuen Teilprodukte entspricht der Summe der bisherigen Teilprodukte, so dass in dieser Konstellation der Wert **H(k)** gleich bleibt, d. h. es entsteht eine **Kollision**.

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



Die Ableitung im Einzelnen wird im Artikel ["Bestimmungsfaktoren für Kollisionsfreiheit"](#) dargelegt.

Der Faktor **C** ist von der Länge **n** der Zeichenfolge und von einem individuell festzulegenden **code** abhängig. Er berechnet sich wie folgt:

$$n = \text{Länge der Zeichenfolge}$$

$$C_k = [(n - (n/(n-1))) / (n/(n-1) - 1)] + \text{code}$$

$$C_k = n * (n - 2) + \text{code}$$

$$C_k = 1681$$

Der Faktor **C** ist außerdem für gleiche Längen der Eingabesequenz gleich und trennt die Zeichen der Positionsgewichtung in kollisionsfreie und kollisionsbelastete Abschnitte. Der Faktor **C** erhält daher die Bezeichnung: Trenn-Konstante **C(k)**.

Zusätzlich wird ein Code zwischen 1 und 99 eingeführt. Wir setzen Code = 1.

Nach Einbindung der Trenn-Konstante **C(k)** ergibt sich der Zwischenwert **H_k** wie folgt:

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

$$H_k = 6928259$$

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.

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. Hier wird die **Basis 77** verwendet. Für jedes Zeichen der Eingabesequenz errechnet das Verfahren den dezimalen Wert (**s_i**), der dann zu (**d_i**) - Ziffern im Zahlensystem zur Basis 77 - umgewandelt wird. Gleichzeitig ermittelt das Verfahren die Summe aller Einzelergebnisse (**s_i**) als zusätzlichen Wert **H(p)** zur Bestimmung verschiedener Steuerparameter und akkumuliert die Ergebnisse (**d_i**) seriell zur Hashfunktionsfolge (HF).

$$s_i = (a_i + 1) * p_i * H_k + p_i + \text{code} + \text{Runde}$$

$$s_i \rightarrow d_i \text{ (base 77)}$$

$$HF = d_1 + d_2 + d_3 + \dots + d_i + \dots + d_m$$

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

$$H_p = \sum_{i=1}^n S_i$$

$$H_p = 618326331960$$

Das gewählte Zahlensystem zur Basis 77 umfasst die folgenden Ziffern:

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

In unserem Beispiel enthält die Hashfunktionsfolge 249 Ziffern im Zahlensystem zur Basis 77:

B2ã7áD0kaxkspUV1BMw&m1aP7691gXfvG1èTBAU2DA6#e2NTHxz2qçkVJ3MoKzk3AIEç53
 #n@kp1E4HQG4kwZge412pàB4#äwOY4oNmHi4èètuQ5ãNSOU5éV&èb1á6nJD7oMqGK66LuD
 46eiOrá7oMqGN2Lknç7ISFAG7hë40â8eNYé#2llää05ã75a89sèFæ29vdbâ8KIépàN9Usi&BAdY4
 mn3GBa576qëUhjjAY9O9CBXysGuCibçW9

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):	6928259
Bestimmungswert (H _p):	618326331960
Gesamtwert (H _p +H _k):	618333260219

Aus den Bestimmungsfaktoren werden die für Verschlüsselungen notwendigen Steuerungsparameter abgeleitet:

Variante	(H _k MOD 11) + 1	=	9	Beginn der Kontraktion
Alpha	((H _k + H _p) MOD 255) + 1	=	150	Offset Chiffre-Alphabet
Beta	(H _k MOD 169) + 1	=	105	Offset Block-Schlüssel
Gamma	((H _p + code) MOD 196) + 1	=	94	Offset Matrix-Schlüssel
Theta	(H _k MOD 32) + 1	=	4	dynamische Zahlenfolgen

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** (Expansionsbasis +1) unterstellt. Jeweils drei Ziffern der Hashfunktionsfolge werden seriell durch **MODULO 256** in dezimale Zahlen 0 bis 255 (ohne Wiederholung) zurückgerechnet und in der BASIS VARIATION gespeichert, einem Array von 16x16 Elementen. Eine rückwärts gerichtete Bestimmung vorhergehender Daten ist nicht möglich (zweite Einweg-Funktion).

Die Rückrechnung auf dezimale Zahlen durch MODULO 256 beginnt beim Parameter **Variante = 9**:

.....axkspU.....

Ziffern Basis 78	dezimal	MODULO 256	- Theta	Element
axk	223672	184	4	180
xks	362598	102	4	98
ksp	284127	223	4	219
spU	332544	0	4	252 (0 +256) - 4
...

BASIS-VARIATION (256 Elemente)

Verteilung der Elemente

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

Die Werte der **BASIS VARIATION** werden direkt auf die Indexwerte von **a_i** (Bytes) bezogen mit Werten von **0** bis **255** (vergleichbar: ASCII-Zeichensatz).

6 Berechnung der „CypherMatrix“

Für die Berechnung der **CypherMatrix** werden die Elemente der BASIS VARIATION in ihrer Verteilung 16x16 Zeichen entweder direkt als Bestimmungsbasis verwendet oder zur Erhöhung der Sicherheit zusätzlich einer dreifachen Permutation unterworfen (CM3).

Endgültige CypherMatrix (CM3) als Ergebnis der dreifachen Permutation

1	CB	6B	D7	AF	2D	5C	68	3F	92	15	20	82	61	41	A1	3E	16
17	9E	12	63	45	3A	73	6C	D5	AB	EA	94	78	1F	77	FB	74	32
33	1A	E4	ED	07	E3	18	36	E7	16	59	F9	A4	C1	BE	0F	75	48
49	0E	EC	72	BB	70	D4	6F	4C	7D	57	49	BC	A5	80	1B	56	64
65	D3	30	0D	EB	B6	47	8D	06	52	40	26	7B	76	38	A9	C0	80
81	50	37	81	11	01	29	B5	7A	13	F8	65	C7	5D	C4	5F	03	96
97	6E	4A	32	99	CE	67	4B	00	27	B1	46	DE	14	C6	C3	86	112
113	C2	DC	BD	A2	66	7F	F7	5B	FC	F6	4E	E6	19	D2	C9	3D	128
129	8A	AC	2C	39	D6	C8	28	A3	4F	AD	DB	F5	E9	71	B0	DD	144
145	6D	CC	8B	D8	AA	2A	95	FE	B8	10	6A	58	62	F4	0C	43	160

161	A6	A8	A7	0B	97	B3	44	9F	79	54	FD	9A	EE	53	B4	91	176
177	2B	F3	B2	A0	69	8C	0A	C5	87	AE	83	DF	08	24	B7	35	192
193	02	48	25	90	4D	E5	7C	05	60	3C	5A	CD	42	1E	64	2F	208
209	BF	88	84	31	23	8E	7E	9C	FF	04	F1	1C	96	89	DA	E1	224
225	55	51	34	85	9D	D1	22	F2	17	FA	1D	D9	D0	F0	09	E2	240
241	3B	E0	93	8F	33	B9	CA	2E	21	EF	E8	9B	BA	CF	5E	98	256

Die **CypherMatrix (CM3)** stellt eine eindeutige und kollisionsfreie Abbildung der Eingabe (Startsequenz) dar. Nach den Grundsätzen der Wahrscheinlichkeit wird eine gleiche Verteilung erst nach **256!** (Fakultät) = **8.578E+505** Fällen auftreten

7 Berechnung der Steuerungsparameter

Zur Steuerung der Verschlüsselung holt das Programm aus der laufenden CypherMatrix die folgenden Parameter:

Alpha: $((H(k) + H(p) \text{ MOD } 255) + 1) = 150$ legt den Beginn des **Chiffre-Alphabets** von 128 Elementen fest:

145	2A	95	FE	B8	..	6A	58	62	F4	..	43	160
161	A6	A8	A7	..	97	B3	44	9F	79	54	FD	9A	EE	53	B4	91	176
177	2B	F3	..	A0	69	8C	..	C5	87	AE	83	24	B7	35	192
193	..	48	25	90	4D	E5	7C	..	60	3C	5A	CD	42	..	64	2F	208
209	BF	88	84	31	23	8E	7E	9C	FF	..	F1	..	96	89	DA	E1	224
225	55	51	34	85	9D	D1	..	F2	..	FA	..	D9	D0	F0	..	E2	240
241	3B	E0	93	8F	33	B9	CA	2E	21	EF	E8	9B	BA	CF	5E	98	256
1	CB	6B	D7	AF	2D	5C	68	3F	92	82	61	41	A1	3E	16
17	9E	..	63	45	3A	73	6C	..	AB	EA	94	78	..	77	FB	74	32
33	..	E4	ED	..	E3	..	36	E7	..	59	F9	A4	C1	BE	48

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.

Beta: $(H(k) \text{ MOD } 169) + 1 = 105$ bestimmt den Offset, ab dem 63 Bytes für den laufenden **Block-Schlüssel** entnommen werden:

97	27	B1	46	DE	14	C6	C3	86	112
113	C2	DC	BD	A2	66	7F	F7	5B	FC	F6	4E	E6	19	D2	C9	3D	128
129	8A	AC	2C	39	D6	C8	28	A3	4F	AD	DB	F5	E9	71	B0	DD	144
145	6D	CC	8B	D8	AA	2A	95	FE	B8	10	6A	58	62	F4	0C	43	160
162	A6	A8	A7	0B	97	B3	44	176

Die Länge der Klartext-Blocks und ebenso die Länge der Block-Schlüssel sind grundsätzlich gleich. Sie können alternativ mit 35, 42, 49, 56, **63** oder 70 Bytes (Vielfaches von 7) gewählt werden. Block-Schlüssel sind nur in Verschlüsselungen mit XOR-Verknüpfungen erforderlich.

Gamma: $((H(p) + \text{code}) \text{ MOD } 196) + 1 = 94$ bestimmt den Offset für die Entnahme des **Matrix-Schlüssels** mit 42 Bytes als Eingabe (Startsequenz) für die nächste Runde:

81	C4	5F	03	96
97	6E	4A	32	99	CE	67	4B	00	27	B1	46	DE	14	C6	C3	86	112	

```

113 C2 DC BD A2 66 7F F7 5B FC F6 4E E6 19 D2 C9 3D 128
129 8A AC 2C 39 D6 C8 28 .. .. .. .. .. .. .. .. 144

```

Zur Initialisierung der nächsten Runde wird der Matrix-Schlüssel auf den Beginn der Funktion zurückgeführt (**loop**). Die jeweiligen Matrix-Schlüssel steuern den gesamten Ablauf des Verfahrens, inhaltsgleich, sowohl beim Sender als auch beim Empfänger.

8 Erweiterung zur universellen Basisfunktion

Die vorhergehenden Erläuterungen beziehen sich nur auf einen Durchlauf (r_j). Um die Prozedur auf eine universell einsetzbare Basisfunktion (Generator) zu erweitern, wird jeweils eine neue Startsequenz mit 42 Bytes als **Matrix-Schlüssel** ab Position **Gamma** aus der laufenden CypherMatrix herausgezogen und auf den Beginn der nächsten Runde zurückgeführt ("**loop**"). Der Matrix Schlüssel als Eingabe initialisiert den nächsten Durchgang (r_{j+1}) und durchläuft die Basisfunktion erneut. So entsteht eine unbegrenzte Zahl von Runden bis ein Endeimpuls gesetzt wird.

B. Verschlüsselungen

Die Verschlüsselung - das Schreiben und Lesen von geheimen Informationen – findet ausschließlich im Codierbereich statt. Dazu ist allerdings eine strukturelle Behandlung der Bitfolgen erforderlich.

1 Ordnungssystem der Bitfolgen

Der Computer kann technisch nur "Bitfolgen" erzeugen. Sie sind in ihrer Menge unbegrenzt (1 bis ∞). Um mit Bitfolgen systematisch zu arbeiten, Zusammenhänge und Gesetzmäßigkeiten zu erkennen und Programme zu gestalten – insbesondere auch Verschlüsselungen - müssen die Bitfolgen **systematisiert** d.h. skaliert und in definierte Abschnitte (Einheiten) geteilt werden. Es liegt ein vergleichbares Phänomen vor, wie bei der Menge aller Zahlen. Daher bietet sich ein sinngemäßer **Strukturvergleich** mit der **Zahlentheorie** an. Das Bit entspricht der Ziffer, das Byte (bzw. Einheit) der Zahl und das Zeichenalphabet der geordneten Menge. Insoweit kann dann beispielsweise für 8-bit Folgen vom "**Bitsystem zur Basis 8**" gesprochen werden. Im Einzelnen ergeben sich:

Bitfolgen:	Bitsystem zur Basis	Zeichen	Alphabet
1-bit	1	2^1 Zeichen	2 Units
2-bit	2	2^2 Zeichen	4 Units
6-bit	6	2^6 Zeichen	64 Units
7-bit	7	2^7 Zeichen	128 Units
8-bit	8	2^8 Bytes	256 Bytes
11-bit	11	2^{11} Zeichen	2048 Units
16-bit	16	2^{16} Zeichen	65536 Units
32-bit	32	2^{32} Zeichen	4294967296 Units

Bisher sind in der Kryptographie nur folgende Bitfolgen als Einheiten definiert: Das Bitsystem zur Basis 1 umfasst die Zeichen: „0“ und „1“ oder historisch: „L“ und „H“. Das Bitsystem zur Basis 4 kennt die Einheit „nibble“ und 8-bit Sequenzen die Einheit: „**Byte**“. Im Bitsystem zur Basis 16 gibt es die Einheit „word“. Alle anderen Bitfolgen werden jeweils mit der enthaltenen Anzahl Bits gekennzeichnet.

2 Umwandlung der Bitfolgen

Umwandlungen von Bitfolgen werden bisher nur im Verfahren „**Coding Base64**“ [Mo01] vorgenommen. 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 wird statisch vorgegeben.

Im „CypherMatrix“ Verfahren wird die Verschlüsselung prinzipiell durch die Umwandlung von einem Bitsystem in ein anderes Bitsystem erreicht. Die allgemein für Verschlüsselungen verwendeten Schritte, wie beispielsweise: Feistel-Netzwerke, Spaltenversion, S-Boxen, Betriebsarten ECB, CBC, CFB und OFB, Konfusion und Diffusion werden in CypherMatrix Verschlüsselungen nicht benötigt. Nur eine Blockverschlüsselung und die XOR-Verknüpfung finden Anwendung.

3 Bit-Konversion

Der Klartext ist grundsätzlich in **Bytes** strukturiert, d.h. er besteht aus Zeichen im Bitsystem zur Basis 8. Zur Verschlüsselung wandelt das Verfahren die Bitfolgen der Bytes anderweitig um in Abschnitte (Einheiten) des angestrebten Bitsystems.

Ein Beispiel für Umwandlungen (Bit-Konversionen): Das Wort „Buddha“ wird vom Bitsystem zur **Basis 8** in Zeichen des Bitsystems zur **Basis 11** umgewandelt.

Alphabet Bitsystem zur Basis 8 (ASCII: 256 Bytes)

```
Byte$(66) = "B"
Byte$(97) = "a"
Byte$(100) = "d"
Byte$(104) = "h"
Byte$(117) = "u"
...      B      u      d      d      h      a      ...
Index    66     117    100    100    104    97
... 01000010 01110101 01100100 01100100 01101000 01100001 ...
```

Umwandlung zur Basis 11

```
... 01000010011 10101011001 00011001000 11010000110 0001 ...
Index          531          1369          200          1670
```

Alphabet Bitsystem zur Basis 11 (2048 Zeichen)

(die Zeichen sind aus zwei ASCII-Zeichen zusammengesetzt)

```
Alphabet$(531) = „cŸ“
Alphabet$(1369) = „g#“
Alphabet$(200) = „fY“
Alphabet$(1670) = „hZ“
```

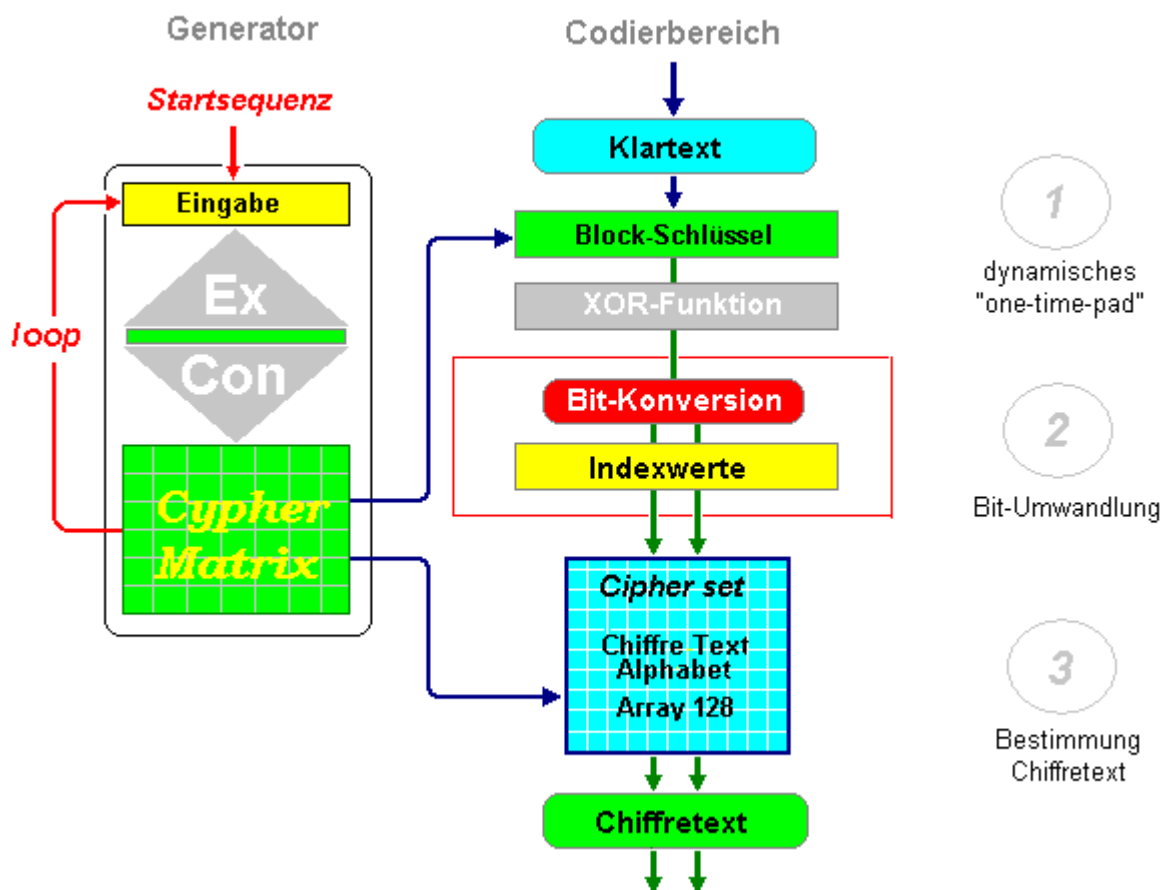
Chiffretext Basis 11: „...cŸg#fYhZ...“

Die Folge der 8-bit Sequenzen teilt das Verfahren in 11-bit Abschnitte. Dabei bleiben die Anzahl der Bits und ihre Reihenfolge gleich. Kein Bit wird hinzugefügt und kein Bit wird weggelassen. Die dezimalen Werte im Bitsystem zur Basis 11 sind Indexwerte für die Zeichen im Array Chiffre-Alphabet. Die Umwandlung kann auf alle Bitsysteme von zur Basis 1 bis zur Basis 12 und höher angewendet werden.

Als Folge der Umwandlung von einem Bitsystem in ein anderes zeigt sich eine grundsätzliche Wirkung des Verfahrens: Der Chiffretext wird länger im Verhältnis der Länge der Zeichen im Chiffretext zur Länge der Zeichen im Klartext. Die Forderung, Klartext und Chiffretext sollten gleiche Länge haben, wird mithin nicht erfüllt.

4 Durchführung der Verschlüsselung

Mit Eingabe der gleichen Startsequenz, sowohl beim Sender als auch beim Empfänger, werden sowohl ein identischer Verlauf als auch identische Steuerungsparameter erzeugt. Die Verschlüsselung vollzieht sich im Codierbereich in drei Schritten:



Die Schritte werden nacheinander durchlaufen.

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 Folgenden wird der Ablauf der Verschlüsselung mit dem Programm `>dynacode.exe<` anhand einer Textdatei demonstriert. Als Startsequenz dient die im vorhergehenden Abschnitt 1.1 gewählte Passphrase:

"7 Nordlichter wandern über den Großen Belt" (n = 42).

Zu verschlüsselnder Text der Datei „Fontane.txt“ (432 Bytes):

*Herr von Ribbeck auf Ribbeck im Havelland,
ein Birnbaum in seinem Garten stand.
Und kam die goldene Herbsteszeit
und die Birnen leuchteten weit und breit,
da stopfte, wenn's Mittag vom Turme scholl,
der von Ribbeck sich beide Taschen voll.
Und kam in Pantinen ein Junge daher,
so rief er: "Junge, willst 'ne Beer?"
Und kam ein Mädél, so rief er: "Lütt Dirn,
kumm man röwer, ick hebb 'ne Birn".*

Theodor Fontane, 1819-1898

4.1 Dynamisches „one-time-pad“

Für den ersten Verschlüsselungsschritt in jeder Runde holt das Verfahren ab Position **Beta = 105** der aktuellen CypherMatrix einen internen Block-Schlüssel in der gleichen Länge wie der eingelesene Klartextblock (63 Bytes) und verknüpft beide mit der XOR-Funktion. Auf diese Weise entsteht ein partielles „one-time-pad“ [SB96a]. Klartext und Schlüssel sind stets gleich lang. Der Schlüssel wird auch nicht wiederholt, da in jeder Runde ein anderer Schlüssel aus der aktuellen CypherMatrix entnommen wird.

Die Bitfolgen gestalten sich im Ablauf wie folgt:

Als erste Runde werden 63 Bytes des Klartextes eingelesen
Klartext (Basis 8):

Herr von Ribbeck auf Ribbeck im Havelland, ein Birnbaum in sei
48 65 72 72 20 76 6F 6E 20 52 69 62 62 65 63 6B 20 61 75 ...
01001000 01100101 01110010 01110010 00100000 01110110 01101111 01101110

Block-Schlüssel (Basis 8):

27 B1 46 DE 14 C6 C3 86 C2 DC BD A2 66 7F F7 5B FC F6 4E ...
00100111 10110001 01000110 11011110 00010100 11000110 11000011 10000110

XOR-verknüpft (Basis 8):

6F D4 34 AC 34 B0 AC E8 E2 8E D4 C0 04 1A 94 30 DC 97 3B ...
01101111 11010100 00110100 10101100 00110100 10110000 10101100 11101000

4.2 Umwandlung von Basis 8 nach Basis 7

Die Folge der 8-bit Abschnitte aus der XOR-Verknüpfung teilt das Verfahren in 7-bit Abschnitte, deren dezimale Werte dienen als 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. Das zugehörige Alphabet von 128 Zeichen kommt ab Position **Alpha = 150** aus der aktuellen CypherMatrix und liefert die Chiffre-Zeichen für die Werte des Bitssystems zur Basis 7:

1	*	•	þ	¸	j	X	b	ô	C	!	¨	§	–	³	D	ÿ	16
17	y	T	ý	š	î	S	´	`	+	ó		i	€	Å	‡	®	32
33	f	§	•	5	H	%	□	M	å		`	<	Z	Í	B	d	48
49	/	¿	^	„	1	#	Ž	~	œ	□	ñ	–	%	Ú	á	U	64
65	Q	4	...	□	Ñ	ò	ú	Û	Đ	ð	â	;	à	¨	□	3	80
81	¹	Ê	.	!	ï	è	>	°	Ï	^	~	Ë	k	x	–	–	96
97	\	h	?	'	,	a	A	j	>	ž	c	E	:	s	l	«	112
113	ê	¨	x	w	û	t	ä	í	ã	6	ç	Y	ù	¤	Á	¼	128

1	2A	95	FE	B8	6A	58	62	F4	43	A6	A8	A7	97	B3	44	9F	16
17	79	54	FD	9A	EE	53	B4	91	2B	F3	A0	69	8C	C5	87	AE	32
33	83	24	B7	35	48	25	90	4D	E5	7C	60	3C	5A	CD	42	64	48
49	2F	BF	88	84	31	23	8E	7E	9C	FF	F1	96	89	DA	E1	55	64
65	51	34	85	9D	D1	F2	FA	D9	D0	F0	E2	3B	E0	93	8F	33	80
81	B9	CA	2E	21	EF	E8	9B	BA	CF	5E	98	CB	6B	D7	AF	2D	96
97	5C	68	3F	92	82	61	41	A1	3E	9E	63	45	3A	73	6C	AB	112
113	EA	94	78	77	FB	74	E4	ED	E3	36	E7	59	F9	A4	C1	BE	128

XOR-verknüpft (Basis 8):

6F D4 34 AC 34 B0 AC E8 E2 8E D4 C0 04 1A 94 30 DC 97 3B ...
01101111 11010100 00110100 10101100 00110100 10110000 10101100 11101000

Bit Konversion: Basis 8 → Basis 7

0110111 1110101 0000110 1001010 1100001 1010010 1100001 0101100 1110100

(dez) 55 117 6 74 97 82 97 44 116

(+1) 56 118 7 75 98 83 98 45 117

Chiffre-Alphabet Basis 7:

(Text) ~ t b â h . h z û

(hex) 7E 74 62 E2 68 2E 68 5A FB

Chiffre-Text: ~tbâh.hZûœÊs□*C â—ið□l*□Qâ§Áú%‡.Yâãĭ+®óÍĐ\$5×³âš\□1^û®«Úç. ...

Die Länge des Chiffretextes verlängert sich im Verhältnis 8:7 d.h. Ein Chiffretextzeichen wird zu 1,143 Klartextzeichen. Die Forderung, Klartext und Chiffretext sollten gleiche Länge haben, wird mithin nicht erfüllt.

4.3 Verschlüsselungsprogramme

Das CypherMatrix Verfahren bewirkt die Verschlüsselung mit der Kombination geeigneter Operationen: XOR-Verknüpfung, Bit-Konversion, Substitution „dyn24“, Byte-Transposition und weitere. Für jedes Programm sind die Schlüssel-Parameter in folgender Variation wählbar:

- Länge des >Matrix-Schlüssels< : 36 – 64 Bytes
- Länge des Block-Schlüssels : 35 – 96 Bytes
- Zahlensystem für Expansionsfunktion: 35 – 96 Ziffern
- Individueller Anwender-Code: 1 - 99 Zahl

Die Einzelheiten sind im Artikel [>Der Kern des CypherMatrix Verfahrens<](#) zusammengestellt. [Sch07b].

Zur Bit-Konversion sind vom Autor die folgenden Verschlüsselungsprogramme entwickelt worden:

Bit-Umwandlungen

Bitsystem Basis	Chiffre-Alphabet	Programm (ohne Permut.)	Aufruf	Programm (3-fachPermut.)	Aufruf	Längenverhältnis
1	2	MonoCode	moc	MiniCode StepMini	mic stm	1:8 1:8
2	4	ZweiCode	zwc	DualCode	duc	1:4
3	8	DreiCode	drc	TriaCode	trc	1:2,66
4	16	VierCode	vrc	FourCode	foc	1:2
5	32	QuinCode	qnc	FiveCode StepFive	fic stf	1:1,6 1:1,6
6	64	CM64Code Coding64	cm64 c64	Neu6Code	n64	1:1,33
7	128	DataCode DataCryp + div. Prog	dac dtc	DynaCode DynaCryp CymaCode. CypherXT + div. Prog.	dyc dnc cmc ext	1:1,143 1:1,143 1:1,143 1:1,143
8	256	ByteCode	byc	CyphCode StepCyph	cyc stc	1:1 1:1
9	512	NeunCode	nec	Cypher89	c89	1:1,79
10	1024	ZehnCode	zec	DecaCode	dec	1:1,6
11	2048	ElvaCode	elc	CM11code StepCM11	cm11 sc11	1:1,46 1:1,46
12	4096	MegaCode	mec	MaxiCode	max	1:1,33

Die Programme sind nach ihrer Bitsystematik geordnet (Bitsystem zur Basis 1 bis Bitsystem zur Basis 12). Außerdem werden die Programme nach der Generierung der CypherMatrix unterschieden, entweder **ohne** oder mit **dreifacher** Permutation. Die Programme können einzeln oder alle per e-mail beim Autor angefordert und unter der **CMLizenz** analysiert und weiter bearbeitet werden.

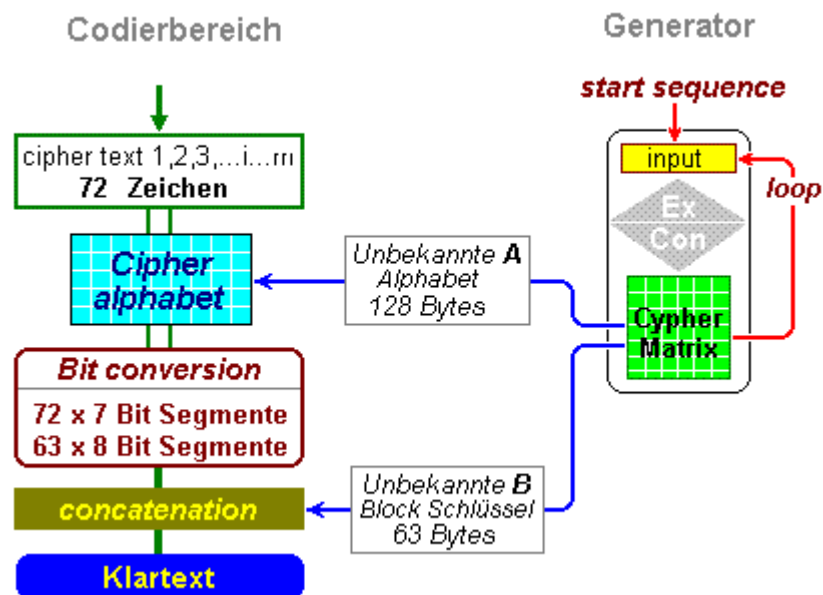
C. Entschlüsselung

Für die Entschlüsselung erzeugt die Basisfunktion 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 **72 Bytes** Chiffretext sucht das Verfahren im identisch erzeugten Chiffre-Alphabet die dezimalen Index-Werte (1 bis 128 (-1)) der einzelnen Zeichen und verbindet deren binäre Zahlen zu einer Bitfolge von **504** Bits. Diese Bitfolge wird wiederum in **63** 8-bit Sequenzen im Bitsystem zur Basis 8 aufgeteilt und mit dem entsprechenden **Block-Schlüssel** XOR-verknüpft. Als Ergebnis erscheint der ursprüngliche Klartext.

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:

$$\begin{aligned}
 cm &= f [f_1 (an, k_1), f_2 (b_1, b_2), f_3 (b_2, k_2)] \\
 an &= f [f_3 (cm, k_2), f_2 (b_2, b_1), f_1 (b_1, k_1)]
 \end{aligned}$$

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 unbekanntem 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 = k_1 und dem in derselben Runde generierten Chiffre-Alphabet $(128) = k_2$ gibt es keine Verbindung. Beide sind zwar aus der aktuellen CypherMatrix entnommen, haben aber keine funktionale Beziehung: $(k_1 \rightarrow (Hk \text{ MOD } 169)+1)$ und $k_2 \rightarrow (Hk +Hp) \text{ MOD } 255)+1$. Die aktuelle CypherMatrix selbst ist aus der ursprünglichen Startsequenz 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.. Somit kann auch „brute force“ keinen Erfolg haben.

D. Sicherheit des Verfahrens

Die im binären Bereich angesiedelten Verschlüsselungsverfahren – außer „brute force“, ElGamal und Encoding Base64 – erfordern fast alle gleiche Klartext- und Geheimtextlängen: ["Strukturvergleich zwischen Klartext und Geheimtext"](#). Viele Kryptographen gehen implizit davon aus, dass Klartext und Geheimtext auch gleich lang sein müssen, da sie ja an der gleichen Stelle wieder abgelegt werden sollen [SK98][BS96b]. Bei gleicher Länge muss daher für jeden Klartextbuchstaben auch ein bestimmtes Geheimzeichen in irgend einer Form vorhanden sein oder jedenfalls durch die funktionale Verbindung zugeordnet werden können. Im Ergebnis wird somit nur die **Struktur** der Bits verändert. Die **Anzahl** der Bits im Klartextzeichen (Input) und im zugehörigen Chiffretextzeichen (Output) bleibt grundsätzlich gleich (8-bit). Da Klartext und Chiffretext im selben Bitssystem zur Basis 8 arbeiten, ist ein einheitliches **Ordnungssystem** vorhanden. Insoweit hat der verschlüsselte Text auch gewisse Informationen über den zugrunde liegenden Klartext [SB96].

In Verschlüsselungen nach dem CypherMatrix Verfahren dagegen findet ein Wechsel im Bitssystem statt. Durch die Umwandlung der Klartextzeichen vom Bitssystem zur **Basis 8** in Zeichen des Bitsystems zur **Basis 7** entsteht ein neues Ordnungssystem. Für jedes Klartextzeichen wird ein Chiffrezeichen generiert, das um den **Faktor 1,143** länger ist, als das ursächliche Klartextzeichen.

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.

Eine Analyse aller dieser Merkmale und Versuche, aus dem Chiffretext Rückschlüsse auf den Klartext zu erhalten, setzen voraus, dass beide Bereiche sich auch **vergleichen lassen**. Bei den herkömmlichen Verschlüsselungsverfahren ist das auch der Fall. Aufgrund des gleichen Bitsystems zur Basis 8 besteht ein einheitliches Ordnungssystem, das sowohl im Klartext als auch im Chiffretext wirksam ist.

In CypherMatrix Verschlüsselungen gibt es kein einheitliches Ordnungssystem. In beiden Bereichen werden verschiedene Bitssysteme verwendet. Somit besteht keine Möglichkeit, sie einheitlich zu analysieren und zu vergleichen. Es fehlt die Ausgangsbasis für alle genannten Angriffsszenarien. Sie sind **wirkungslos** und wir können sie vergessen.

E. Abschließende Bemerkungen

Auf der Grundlage einer sinnvollen Systematisierung der Bitfolgen, angelehnt an die Systematisierung in der Zahlentheorie, reichen Umwandlungen vom Bitsystem zur Basis 8 in ein anderes Bitsystem (Basis 1 bis Basis 12) für eine einfache und sichere Verschlüsselung aus.

Die zugeordneten Chiffretext-Alphabete sind jeweils neu zu definieren. Ab Basis 9 werden zweckmäßigerweise zwei Zeichen des ASCII-Zeichensatzes kombiniert oder UNICODE verwendet..

Infolge Fehlens eines einheitlichen Ordnungssystems sind Verschlüsselungen durch Wechsel der Bitsysteme mit herkömmlichen Angriffen nicht zu brechen. Allerdings muss ein etwas längerer Chiffretext in Kauf genommen werden.

Hinweise

- [Mo01] Morin, R.C., „How To Base 64 Encoding“, www.kbcafe.com/articles/HowToBase64.pdf
- [SB96a] Schneier, Bruce, Angewandte Kryptographie (deutsche Ausgabe), Bonn 1996, S.17
- [SB96b] Schneier, Bruce, a.a.O., S. 229
- [Sch04] Schnoor, E.E., „Strukturvergleich zwischen Klartext und Geheimtext verschlüsselter Dateien“, www.telecypher.net/EQUILANG.HTM
- [Sch07a] Schnoor, E.E., „Bestimmungsfaktoren für Kollisionsfreiheit“, www.telecypher.net/Kollfrei.pdf
- [Sch07b] Schnoor, E.E., „Der Kern des CypherMatrix Verfahrens“, Abschn. D.4, www.telecypher.net/CYPHKERN.HTM
- [SK98] Schmeh, Klaus, Safer Net, Kryptographie im Internet und Intranet, Heidelberg 1998, S.61

München, im Dezember 2011