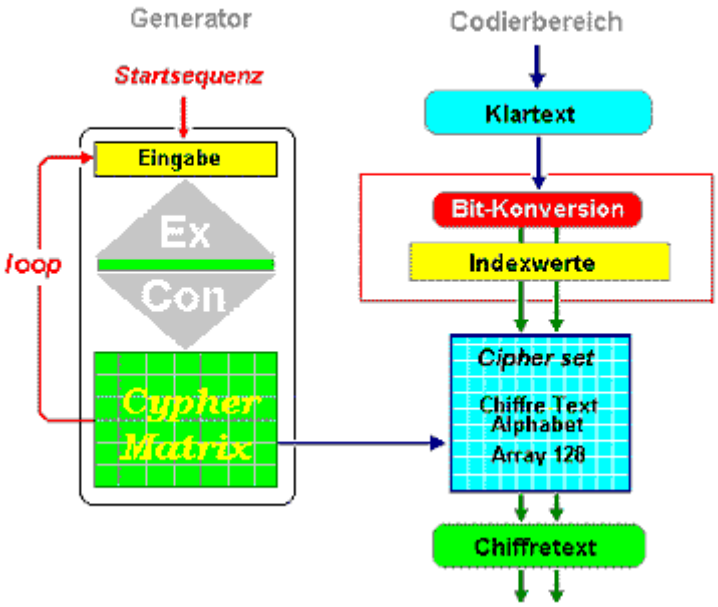
Data Generator

(Ernst Erich Schnoor)

In CypherMatrix procedure – explained in whitepaper: "Cryptographic Basic Function in Byte Techniques" – encrypting of an Information is comparatively simple [#1]:

> **Generator** creates the necessary **system alphabet**
> and cipher is performed in the **coding area**.



Both sectors are combined together, but can be used separate, as well. The essential task of the generator is to serve with all required parameters necessary for encryption. Actual writing of ciphertext takes place in the coding area.

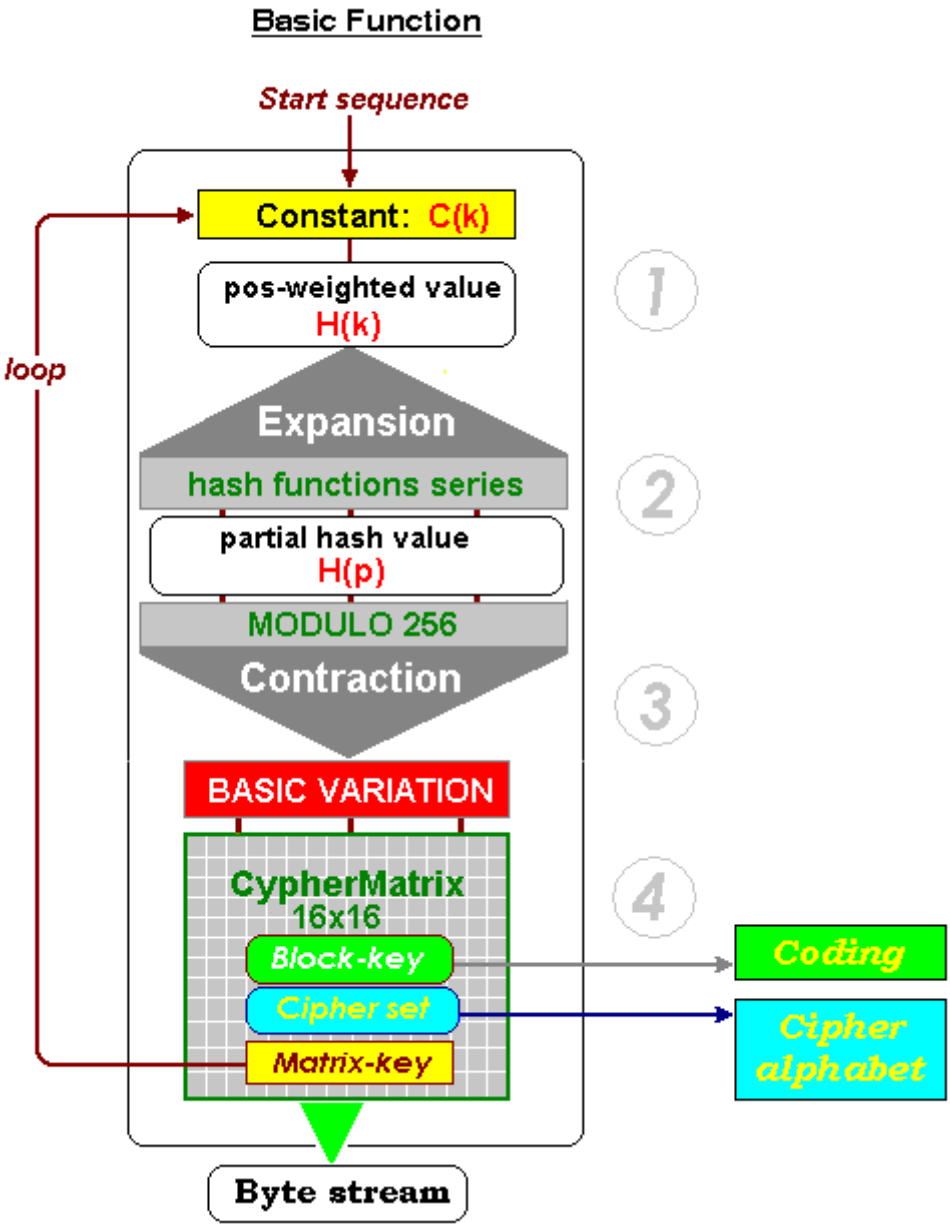
Generator

Any **start sequence** (pass phrase) with at least 36 and optimal 42 characters drives and controls the whole process. Some examples:

7 kangaroos jumping along the Times Square	[42 bytes]
Red eagles resting on the rocks of Amarillo	[43 bytes]
3 koala bears are diving at Murray's Mouth	[42 bytes]
Blue flamingos flying to Northern Sutherland	[44 bytes]

Start sequence should be easy to remember and possibly somewhat catchy and of funny words. The chosen phrase cannot be guessed, can easyly be kept in mind and may not be written down, anywhere. Because of their length lexical attacks and iterative searching should be impossible. An attacker even isn't able to analyse parts of the start sequence neither apart nor successive because the passphrase could be found in total only, if at all.

Each start sequence at sender and recipient, as well, generates identical control parameters and course of the procedures.



The generator delivers for each round all control parameters necessary for cipher processing:

1. Cipher alphabet (system alphabet) for the actual round,
2. block key for XOR-concatenation and
3. matrix key as start sequence for the next round.

At end of each round a **CypherMatrix** with 16x16 elements is created, which serves for all control parameters to achieve the encryption. The matrix key (42 signs) is to be led back to the beginning (**loop**) in order to initialize the next round, till an end mark is designated. Due to probability laws a repetition of identical Data will occur first in **256!** (faculty) = **8E+506** cases .

Start Sequence

For analysing Data the following start sequence (input) is chosen as an example:

The Hudson River flows into California Bay [42 bytes]

54 68 65 20 48 75 64 73 6F 6E 20 52 69 76 65 72 20 66 6C 6F 77
73 20 69 6E 74 6F 20 43 61 6C 69 66 6F 72 6E 69 61 20 42 61 79

The goal is to find an evident determination base for analysing the start sequence.
Input **m** is a series of definite bytes **a(i)** with lengths **n.** In order to analyse the series as state of facts the single characters have to be systematized (scaled). For this each byte **a(i)** gets an index and all bytes will be appropriate associated with each other (addition).

$$m = a_1 + a_2 + a_3 + \ldots a_i + \ldots a_n$$

(single value of „a(i)“ has to increase by (+1) because otherwise ASCII-zero (0) would not be considered)

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

$$m = 3965$$

In order to differentiate single bytes **a(i)** inside the series additional criterions have to be added, because otherwise no definite results will resume, especial: position weighting and collisionfree.

Position Weighting

With reference to **Renè Descartes** (1596 - 1650) we know that every object (fact) – which is scalable in its dimensions – can be exactly determined by their coordinates for **subject, location** and **time** (cartesian coordinate system).

In order to distinguish single characters we multiply each byte **a**(**i**) with its location **p**(**i**) and time **t**(**i**) and summarize all results to a destination value (**H**(**k**). But time is not relevant, we set: **t = 1.**

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

$$H(k) = 85864$$

Excluding Collisions

Collisions in consequence of exchanging bytes inside series are not excluded, yet. In order to achieve this we shift position weighting by a distance **C** to an area above length **n,** that means: p_i will be extended by distance **C.** Creating of deviding constant **C**(**k**) is detailed explained in whitepaper **"Determinants leading to collision free"** .

$$C(k) = n * (n - 2) + code$$
$$C(k) = 1681$$

With code – a shosen number between 1 and 99 – the function will be individualized.
We set: **code = 1**

By including separating constant **C**(**k**) the destination value **H**(**k**) results as follows (r = round):

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

$$H(k) = 6754994$$

The calculated destination value **H**(**k**) with a range of about 10^7 = 10000000 facilities avoids collisions but appears being too low for a definite mapping.

Expansion

To extend the destination base an **expansion** function is introduced which widens the sequence to an extensive series in a superior number system. Number system of expansion is fixed with **base 77.** For each value of the inserted sequence the function calculates its decimal value s_i which is changed to d_i (digits in number system on base 77). The function simultaneous calculates the sum of all single results s_i as an additional destination value **H**(**p**) for creating several control parameters and accumulates serial the results d_i as hash function series (HF).

$$s_i = (a_i + 1) * p_i * H_k + p_i + code + round$$

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

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

(m = number of digits in system on base 77)

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

$$H_p = 580010805803$$

The chosen number system on base 77 comprises the following digits:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz&#@àáâãäåæçèéêë
(determinated by the author, not standardized)

By generating hash function series the section "**River**" at position 9 of the inserted pass phrase results to the following calculations:

char	pi		Hk	(ai+1)*pi*Hk		Si	Basis 77
	(ai+1)	(ai+1)*pi			pi+code+r		
R	83	12	996	6754994	6727974024	14	6727974038	**2bU7éG**
i	106	13	1378	6754994	9308381732	15	9308381747	**3XzLën**
v	119	14	1666	6754994	11253820004	16	11253820020	**4CAlbw**
e	102	15	1530	6754994	10335140820	17	10335140837	**3#0OwL**
r	115	16	1840	6754994	12429188960	18	12429188978	**4jiDCq**
						sum:	580010805803	2yLjeFi

The hash function series comprises 246 digits in number system on base 77:

GPqs**keRGêé**wzpBqPS8azåAnrF1x3xa21wàêBà2OOèjå2droud2xMâyUäw44ã**2bU7éG3XzL**
ën4CAlbw3#0OwL4jiDCq1UztLf4mKOhX5Cé8vt5jXqào6MIn@G6SU9ãZ1ãàTcE6QàwSæ6æ
Irèç7jgNàé7g6êFa2NgxQU4åçLnh7Pè6&B8XNovq8Z&0@H8bBj2k9cvKgMA3Y4ui9éáélç
9yo3Sf9MkTBê3GN#VJ6që48mA27bfMCymE4w

The variables are digits (not characters). There is no way back to the start sequence (first **one-way-function**). The function simultaneous calculates the following destination Data:

Destination factors

Separating constant C(k):	1681
Position weighting value (H_k):	6754994
Destination value (H_p):	580010805803
Total destination value (H_p+H_k):	580017560797

From destination factors the following control parameters are derived:

Variante	(H_k MOD 11) +1	=	5	begin of contraction
Alpha	((H_k + H_p) MOD 255) +1	=	203	offset cipher alphabet
Beta	(H_k MOD 169) +1	=	65	offset block key
Gamma	((H_p + code) MOD 196) +1	=	25	offset matrix key
Delta	((H_k + H_p) MOD 155) +code	=	53	dynamic bit series
Theta	(H_k MOD 32) +1	=	19	offset back calculating
Omega	(H_k MOD 95) +1	=	20	begin double signs
Kappa	(H_k MOD 16484) +1	=	13039	length double signs

Control parameters serve for solution of several cryptographic tasks.

Contraction to BASIC VARIATION

In order to reduce the variables back to decimal values a contract function is introduced. The digits of hash function series are assumed to be digits in number system on **base 78** (expansion base +1). Each three digits of the function series are reconverted in serial manner by **MODULO 256** to decimal numbers **0** to **255** (without repetition). The parameter **Theta** is deducted. Results are stored in BASIC VARIATION, an array of 16x16 elements. A backwards searching of foregoing Data is not possible (second **one-way-function**).

The first four reconversions beginning at variante = **5** shows as follows:

keRGêé

3 digits base 78	decimal	Modulo 256	- Theta	element
keR	283011	131	19	**112**
eRG	245482	234	19	**215**
RGê	165591	215	19	**196**
Gêé	103268	100	19	**81**

BASIC VARIATION (256 elements)
distribution of elements

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

Generating Indexes

All index values (16x16) to perform the CypherMatrix are new generated from elements of BASIC VARIATION in a special two dimensional array **IndexFolge(2,16)** (another application of CypherMatrix function).
In parts of source code:

```
SHARED IndexFolge(2,16), Delta, Omega

FOR B = 1 TO 2
   IF B=1 THEN X = Delta
   IF B=2 THEN X = Omega
   FOR C = 1 TO 16
      INCR X
      IF X > 256 THEN X = 1
      A = VARIATION(X) MOD 16              elements of BASIC VARIATION
      IndexFolge(B,C) = A
      IF C>1 THEN
         L = 0
         DO
           INCR L
           IF IndexFolge(B,L) = A THEN
              INCR A
              A = (A MOD 16)
              IndexFolge(B,C) = A
              L = 0
           END IF
         LOOP UNTIL L = C-1
      END IF
      IndexFolge(B,C)  =  IndexFolge(B,C)+1    array IndexFolge (2,16)
   NEXT C
NEXT B
```

```
N = Alpha
FOR I = 1 TO 16
   FOR J = 1 TO 16
      X = IndexFolge(1,I)
      Y = IndexFolge(2,J)
      Matrix$(X,Y) = VARIATION$(N)            generating of CypherMatrix
      INCR N
      IF N > 256 THEN N = 1
    NEXT J
 NEXT I
```

IndexFolge (2,16) to establish (permutation) the first CypherMatrix:

index series (1,16): 10 11 15 12 14 3 6 9 13 5 16 4 1 2 7 8
index series (2,16): 3 15 16 8 4 11 14 1 2 7 5 10 6 9 12 13

CypherMatrix

With **CypherMatrix** a definite destination base is attained for analyzing the start sequence. According to theory of probabilities an identical CypherMatrix will be created first in **256!** (faculty) = **8E+506** cases.

Final CypherMatrix (variant D)

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

In each round the control parameters necessary to perform the cipher are extracted from the current CypherMatrix at defined positions.

Control parameters

As cipher alphabet (system alphabet of bitsystem on base 7) 128 characters are taken from the actual CypherMatrix at position **Alpha = 203.** Certain signs (hex: 00 bis 20, 22, 2C, B0, B1, B2, D5, DB, DC, DD, DE, DF and others) are ignored because they do still their original task (e.g. **1A =** ASCII-26) and disturb the proper realization of the program.

Cipher alphabet (hex)

```
E1 .. C8 5D 73 63 34 44 30 .. A5 A7 90 45 5C ..
.. 92 2D 9A 31 E4 .. 67 74 55 8B C7 EA A9 23 FE
.. E6 FD F4 CD 48 B9 96 8C B7 29 21 F5 65 9E 6E
E0 7C AA 4F .. 39 24 C5 FA 64 93 26 .. C6 E8 AD
C3 B4 83 BC 3F 5B B6 .. C0 A3 88 99 A4 EC BB B3
46 25 6C .. 91 BA 71 85 BE 79 6B D9 6D 9F .. A0
9D AB 32 75 .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. .. .. .. 66 7B 98 .. 41 7A
B8 A6 77 59 F2 3E .. 42 57 AE FF .. .. C2 A1 F1
43 .. .. .. 61 8F 8E 8D .. 56 .. .. 9C .. .. 95
.. D3 4B 4C 2B E7 .. .. AC 8A 72 A8 4D E3 C1 F3
```

Cipher alphabet (base 7)

```
  1   f { ˜ A z ¸ ¦ w Y ò > B W ® □ Â    16
 17   ¡ ñ C a □ Ž □ V œ • Ó K L + ç ¬   32
 33   Š r ¨ M ã Á ó á È ] s c 4 D 0 ¥    48
 49   § □ E \ ’ - š 1 ä g t U ‹ Ç ê ©   64
 65   # þ æ ý ô Í H ¹ – Œ · ) ! õ e ž    80
 81   n à | ª O 9 $ Å ú d “ & Æ è  Ã     96
 97   ´ ƒ ¼ ? [ ¶ À £ ˆ ™ ¤ ì » ³ F %   112
113   l ‘ ° q … ¾ y k Ù m Ÿ   □ « 2 u  128
```

Block key

The block key extracted at position **Beta = 65** of the current CypherMatrix reads as follows:

Ã´ƒ¼?[¶ÞÀ£ˆ™¤ì»³F%l#‘ºq…¾ykÙmŸ± □«2u'^€ÉÊ†

```
 49   .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..  64
 65   C3 B4 83 BC 3F 5B B6 DE C0 A3 88 99 A4 EC BB B3  80
 81   46 25 6C 10 91 BA 71 85 BE 79 6B D9 6D 9F B1 A0  96
 97   9D AB 32 75 27 5E 80 C9 CA 86 .. .. .. .. .. .. 112
113   .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. 128
```

Matrix key

To get the start sequenz (pass phrase) for the next round a series of **42** characters is extracted at position **Gamma = 25** of the current CypherMatrix:

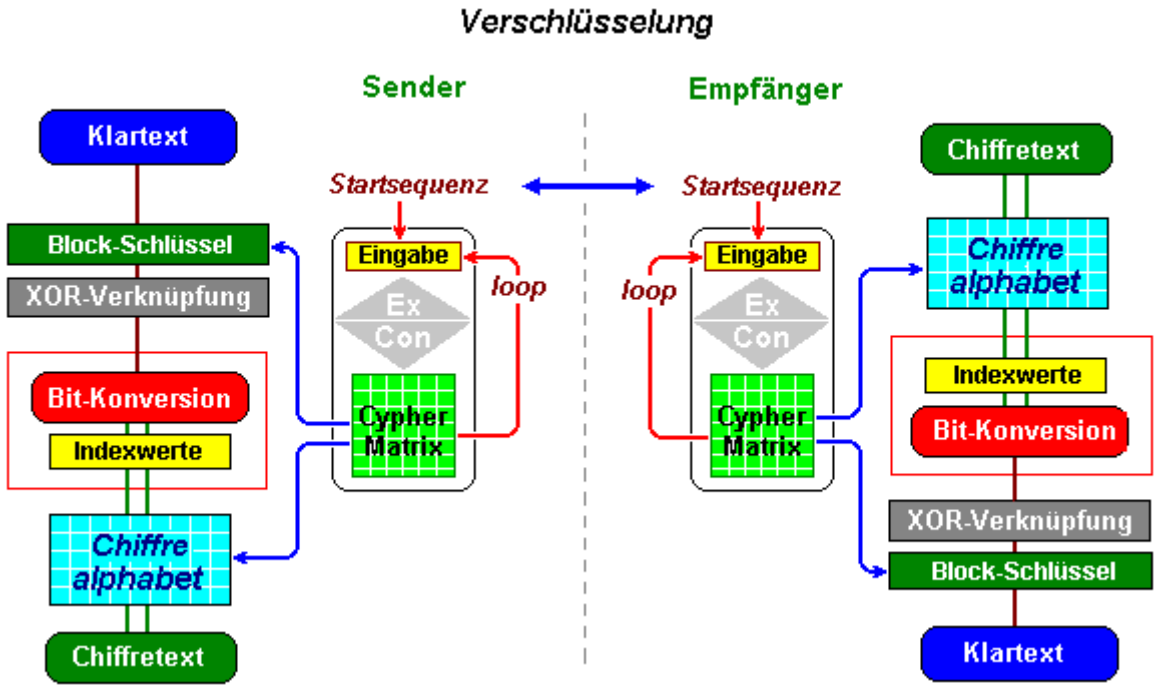
tU‹Çê©#þÝæýôÍH¹–Œ·)!õežnà|ªO#9$Åúd“&#ÆèÃ´

```
  1   .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..  16
 17   .. .. .. .. .. .. .. .. 74 55 8B C7 EA A9 23 FE  32
 33   DD E6 FD F4 CD 48 B9 96 8C B7 29 21 F5 65 9E 6E  48
 49   E0 7C AA 4F 05 39 24 C5 FA 64 93 26 07 C6 E8 AD  64
 65   C3 B4 .. .. .. .. .. .. .. .. .. .. .. .. .. ..  80
```

The matrix key is led back to the begin of the function (**loop**). The respective matrix key controls the complete course of the procedure on equal content at sender and recipient, as well.

Coding area

Encryption – writing and reading of secret informations – is exclusively performed in the coding area. By inserting an identical start sequence at sender and recipient, as well an equal course and identic control parameters are generated. The following schema shows the connections:



Ciphering is performed by following alternatives:

1. **Basic Coding:** bit conversion without further operations or
2. **Compound Coding:** bit conversion with additional operations ,

   a) with XOR-concatenation (foregoing or succeding),
   b) connected with further operations.

According to foregoing prinziples the Author has developed a range of programs which are listened in the following survey:

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).

Programs, single or in groups with or without sourcecode, may requested per e-mail at the author and tested or further development within the scope of the **CMLizenz.** All programs are Dos-programs and will run under WindowsXP only. They have to be converted to **C#,** as already done under direction of Prof. **Bernhard Esslinger** (University of Singen) and his CrypTool-team (especial: **Michael Schäfer**) with the programs **DataCode** and **DynaCode**. All further programs still have to be converted to C#.

You also may download the analysing program **"StepConD.exe"** and test the above mentioned programs.
E-mail adress: eschnoor@multi-matrix.de

Munich, in april 2013

Notes

[#1] All statements depend on scientific research of the author during the last 10 years. They are targeted for expanding sciences of digital cryptography. More details in whitepaper: "New Techniques in digital Cryptography"