

CypherMatrix as a cryptographic Data Generator

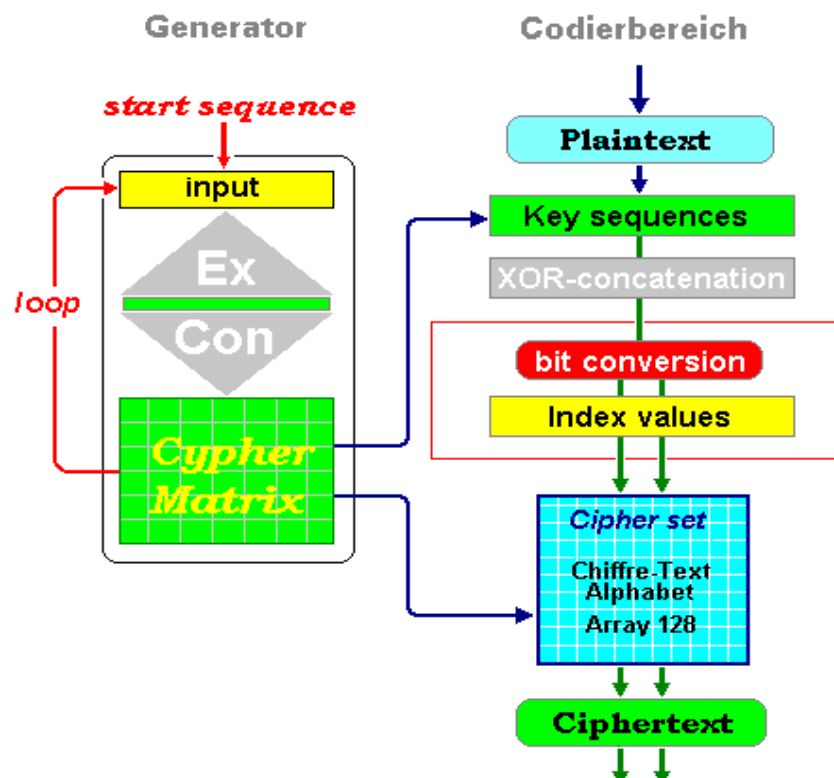
Ernst Erich Schnoor, Munich

Preliminary Remarks

The original intention was to search for a new encryption method – possibly in excess of today usually used bit manipulations – so no bits, but with whole characters and numbers (bytes) just as cryptographers have done since centuries – especially when using modern technology, so to say: **byte technology** when giving a distinguish name to this area.

The procedure found as a result – named by the Author: **CypherMatrix**[®] method (filed patent DPMA 19811593 as of 18.03.1998) – has little common actions with those methods mostly used today. The CypherMatrix procedure uses simple arithmetik, MODULO calculations and higher number systems (base 2 up to base 128). On principle it is very simple, even if it seems to be somewhat complicated first.

Divergent from actual methods CypherMatrix works with two different sectors: **Data generator** and **Coding area**. Both sectors are combined with each other but may be used even separately, different and each on its own.



The Data generator controls the whole procedure and results in several **control parameters** necessary for processing combined applications. The **CypherMatrix** especially is used for:

1. Creating unlimited character series (random number generator),
2. generating collision free dynamic hash values,
3. extended digital signatures and
4. all modes of encryptions.

Detailed description of all this applications would carry things too far and would exceed this talk. So, explanations are limited to construction and working method of the Data generator.

Start Sequence as Master Key

The generator starts with an input of optimal 42 characters (bytes) – here denoted as **Start sequence**. This input initializes the total course of the procedure, but is no initial vector (**IV**) in a conventional sense. Some examples:

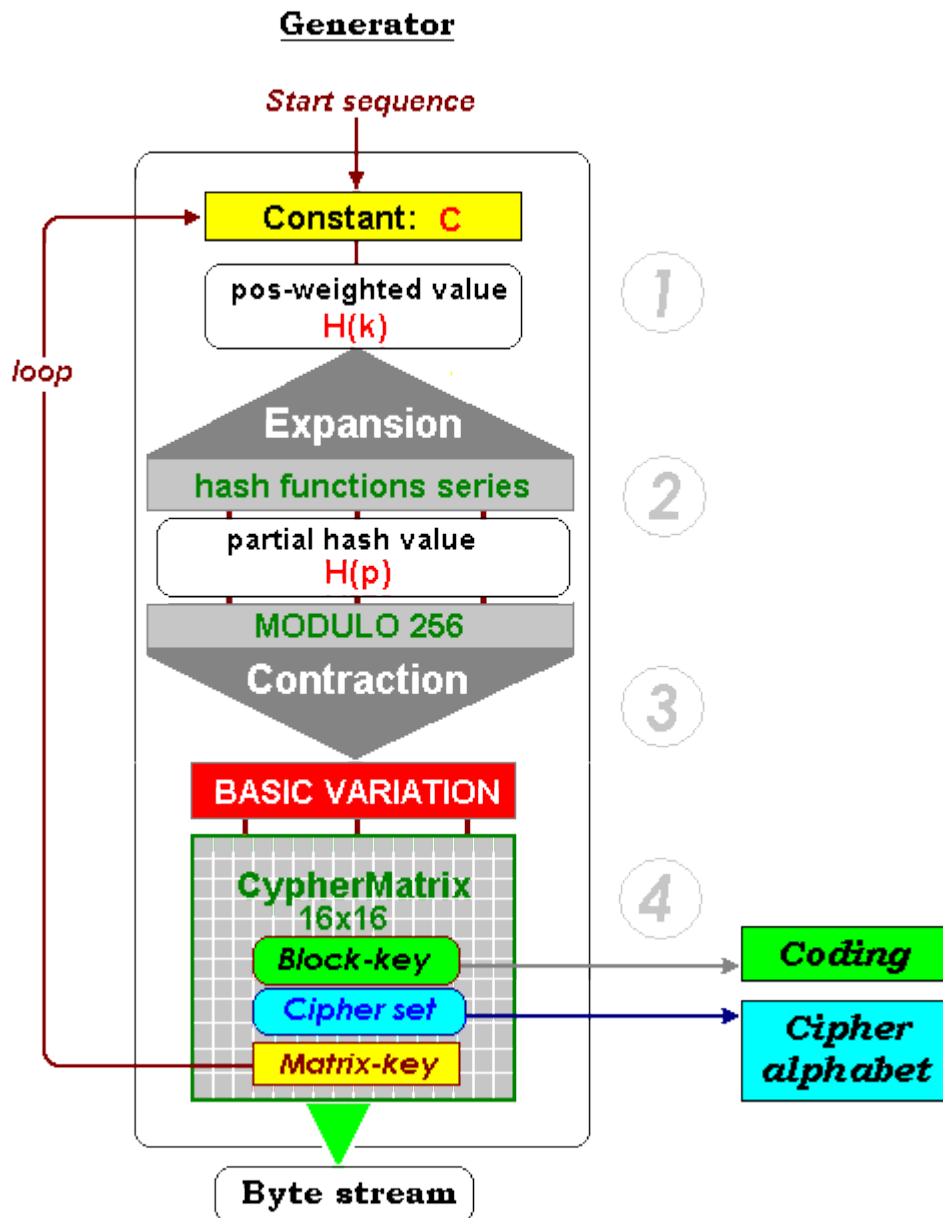
Blue heaven over Minnesota all the day long	[43 bytes]
Le petit prince sourit avec mélancolie	[38 bytes]
7 kangaroos jumping along the Times Square	[42 bytes]
Woody woodpecker diving in the Murray Mouth	[42 bytes]
The flying Dutchman was never in New Zealand	[44 bytes]

There is no border for your imagination. Start sequences (pass phrases) should be somewhat funny, but unusual. They have to be easy to remember and it's not necessary to write them down. Because of their length they will not be object of lexical attacks or iterative search techniques.

Collision free One-Way-Hashfunctions

In order to work with the **start sequence** - their optimal length of 42 bytes is still somewhat short – the sequence has to be widened and assigned to a univalent value in order to get a collision free control value (hash value). So done by the following steps:

1. Widening the start sequence to a destination factor which avoids collisions (hash constant **Ck**) and to a position weighted result **H(k)**,
2. expanding to a Hashfunktion Series (**HFR**) with 160 up to 2400 digits: **Expansion**,
3. contracting the Hashfunktion Series by MODULO 256 to an array with 16x16 elements (BASIC VARIATION): **Contraction** and
4. threefold permutation of the array to perform the **CypherMatrix** with 16x16 characters as final result.



All **control parameters** necessary to perform cryptographic applications are extracted out of the CypherMatrix. The most important parameter is the **Matrix key**, a series of 42 characters (bytes) which are to be led back (loop) to the beginning of the cycle in order to initialize the next round. The four steps outlined above generating a new CypherMatrix are repeated in each cycle, but by a different matrix key.

Single steps with the following start sequence read as follows:

Baron Münchhausen reitet über den Bodensee (n = 42)

The start sequence of length (n) comprises a series of definite signs **a(i)** (bytes):

a(1) a(2) a(3) a(i) a(n)

To assign the start sequence univalent with a value **H(k)** each byte **a(i)** is denoted with an index value and the single bytes have to be linked together in a qualified manner. Numbers of extended ASCII character set (zero – 255) serve as indexes. This values can be expressed in decimal numbers in order to calculate and convert them into other – especially higher - number system bases.

First assignment by addition:

$$H(k) = a(1) + a(2) + a(3) + \dots a(i) + \dots a(n)$$

(Single values for a(i) are increased by (+1) because otherwise ASCII-zero (0) would not be considered))

$$H(k) = \sum_{i=1}^n (a(i) + 1)$$

$$H(k) = 4074$$

But the thus calculated value is far away to serve as a **hash value**. In order to obtain a unmistakable association of all bytes some more features have to be added.

We may remember **Renè Descartes**. He would be more than 350 years old, but yet here of great interest. Every **fact** is exactly determined by coordinates for **subject**, **location** and **time** (cartesian coordinate system). While the subject **a(i)** is marked by ASCII-indexes only „location“ and „time“ are missed. Coordinate „time“ may be relevant if there is a functional connection between CPU rate and single bytes. That may be in „voice streaming“ but not here. Hence, we set coordinate „time“ indifferent with **t = 1**. As coordinate for „location“ we state position **p(i)** inside the sequence to be the location of the concerned character **a(i)**.

Each sequence character **a(i)** now will be **position weighted** by multiplying with its location **p(i)**:

$$H(k) = \sum_{i=1}^n (a(i) + 1) * p(i) * t(i)$$

$$t(i) = 1$$

But collisions are not yet excluded. A collision occurs when despite of changing characters inside the start sequence an equal hash value results. Thus, an additional factor must be provided which comprises all cases leading to collisions. Here serves the **hash constant Ck** which is dependent on length of the start sequence (**n**) and on an individual fixed **user code** (1 up to 99).

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

$$n = \text{sqrt} (C(k) + 1 - code) + 1$$

$$C(k) = 1681$$

A detailed explanation of hash constant **Ck** would carry things too far and would exceed this talk. You may read the pdf-file „**Determinants leading to collision free**“ in internet at:

<http://www.telecypher.net/Collfree.pdf>

Including hash constant **Ck** the procedure now calculates a position weighted partial hash value **H(k)** with no collisions:

$$H(k) = \sum_{i=1}^n (a(i) + 1) * (C(k) + p(i))$$

$$H(k) = 6935825$$

The calculated value **H(k)** excludes collisions but on the other side is still too narrow to establish a secure unchallengeable hash result, To obtain more security an **expansion function** is introduced which widens the determining factors to a voluminous scale without losing the quality of being collision free.

Expanding to Hashfunktion Series (HFR)

The function gets the task to widen the really short start sequence of 42 bytes to possibly utmost variables (160 up to 2400 digits) by way of a **Hashfunktion series**. This function converts values **a(i)** of the start sequence into digits of higher number systems – here to **base 85**. Converting factor may be determined individually from **base 36** up to **base 96**. In other applications with CypherMatrix method often base 77 is used alternatively.

The procedure generates the **Hashfunktion series** and a second intermediate result **H(p)** according to following pseudo-code:

```

FOR i = 1 TO n
  si = (( ai + 1 ) * pi * Hk ) + ( pi + c )
  CALL DezNachSystem (85, si, digit)
  Reihe = Reihe + digit
  Hp = Hp + si
NEXT i

```

- n = length of start sequence
- p_i = position in the start sequence
- a_i = element of the start sequence (extended ASCII-character set with 256 elements)
- s_i = single result of expansion calculation
- c = user code
- H_k = position weighted hash value
- H_p = intermediate result (hash value)

expansion base: 85
 contracting base: 86
 digit: number in number system to base 85
 Reihe: Hashfunktion series

The function „DezNachSystem (85, s(i), digit)“ converts decimal numbers to digits in number system on **base 85**. The assignment „Reihe = Reihe + digit“ expands characters of the start sequence to at most as possible digital elements: **expansion effect**.

Chosen number system on base 85 comprises the following digits:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
 abcdefghijklmnopqrstuvwxyz&#@âãäåæçèéêëìíîï{|}€

(determined by the Author, not standardized)

With the selected start sequence the procedure results to the following values as bases for control Data:

hash constant (Ck): 1680 + 1 = 1681
 position weighted value (Hk): 6935825
 intermediate hash value (Hp): 606406116520
 total hash value (Hk+Hp): 606413052345

The procedure derivates from this values following **control parameters**:

Variante : (Hk MOD 11)+1 = 7 begin of contraction (86->Dec)
 Alpha : (Hk+Hp MOD 255)+1 = 91 begin ciphertext alphabet
 Beta : (Hk MOD 169)+1 = 66 offset block key
 Gamma : ((Hp+Code) MOD 196)+1 = 150 offset matrix key
 Delta : ((Hk+Hp) MOD 155)+Code = 31 begin of dynamic bit series
 Theta : (Hk MOD 32)+1 = 18 dynamic number series

The chosen **user code** (1 up to 99) is included at calculation of **gamma** (offset for extraction matrix key) and **delta**. This enables user to initialize up to 99 different courses with the same start sequence. Because since 2nd round matrix keys (as start sequences) are generated anew in each cycle there will be different **control parameters** in each run.

Single digits of **Hashfunktion series** are calculated as follows:

char	a_{i+1}	p_i	$(a_{i+1}) * p_i$	H_k	$(a_{i+1}) * p_i * H_k$	s_i	base 85
.
M	78	7	546	6935825	3786960450	3786960458	çkaà}
ü	130	8	1040	6935825	7213258000	7213258009	1rFnăî
n	111	9	999	6935825	6928889175	6928889185	1l&jäU

c	100	10	1000	6935825	6935825000	6935825011	1lèááQ
h	105	11	1155	6935825	8010877875	8010877887	1ãdW2H
h	105	12	1260	6935825	8739139500	8739139513	1 ZJf3
a	98	13	1274	6935825	8836241050	8836241064	1€NTCJ
u	118	14	1652	6935825	11457982900	11457982915	2ngY€0
s	116	15	1740	6935825	12068335500	12068335516	2zGMçæ
e	102	16	1632	6935825	11319266400	11319266417	2kæjeH
n	111	17	1887	6935825	13087901775	13087901793	2ïzdPI
.
					sum =	606406116520	1puµ7aA

The function calculates 251 digits in number system to base 85:

1pWY58ëwW7Q3oLhjæVs@xisCyè#6laQQFHbçkaà}1rFnâî1l&jâU1lèááQ1ãdW21|
ZJf31€NTCJ2ngY€02zGMçæ2kæjeH2ïzdPIíjfqO3ZR26y3G4ZoL3eà67R420Rtè3F
O4X86€A1OqZ1á5O8DBg4FDZM84deFâT5I9íJF1kjë2B4ë0€EH58w5iX5zxQNZ3
un3Hf6Pz&TM5æj4fr64€ëAd6àG0Cà7LhWqk6jtäy6xHâKh

This expansion constitutes a first **one-way function** of the procedure. There is no way back to the initial start sequence. By the way the function demonstrates that the start sequence has to be searched for **in total** in a unique process, if at all.

Contracting to BASIC VARIATION

In order to reduce the variables of Hashfunktion Series to decimal data, three each digits of the function series – beginning at **variante** - are reconverted to decimal numbers (0 to 255) by **MODULO 256** (without repetition) and stored in an array of 16x16 elements: **BASIC VARIATION**. To achieve the reverse conversion the Hashfunktion Series is assumed to be digits in number system on **base 86** (expansion base 85 + 1). Compared with number system on base 85 in number system on base 86 there is only one more digit, but that remains unimportant to the contracting procedure.

This **contraction** is the second **one-way-function** of the procedure. The function is not reversible and retrograde destination of the Hashfunktion Series is not achievable. The contraction process may be performed alternatively with other MODULO factors (2 up to 256, especially with 8, 16, 24, 32 and 64)

In single steps the procedure works as follows::

FOR k=1 TO 256

digit = MID\$(Reihe, k, 3)	three digits number base 86
CALL SystemNachDez (86, digit, decimal)	reconversion
element = (decimal MOD 256)	limitation on 0 to 255

variation(k)=element
IF k>1 THEN

BASIC VARIATION: 256 elements

```

n = 0
DO
  INCR n
  IF variation(n) = element THEN
    INCR element
    variation(k) = element
    n = 0
  END IF
LOOP UNTIL n = k-1
END IF
NEXT k

```

By this we get the following abridged derivation:

1pu|WY58ëwW7Q3oLhjæVs@xisCyè#6laQQFHbKb.....

With parameter **variante = 7** (begin of reconversion) the process begins at position seven of the hashfunktion series:

3 digits base 86	decimal	modulo 256
58ë	37744	112
8ëw	65762	226
ëwW	567116	76
wW7	431727	111
W7Q	237300	244
7Q3	54011	251
Q3o	192604	92
3oL	26509	141
oLh	371649	193
...

In first round the following array **variation (k)** is generated with 256 elements (if an element appears double the following element is increased by +1)

BASIC VARIATION (256 elements)
Distribution of the elements

112 226 076 111 244 251 092 141 193 083 177 013 220 000 211 254
138 096 093 033 053 052 079 070 014 230 243 067 055 218 082 184
081 057 199 119 194 113 022 203 027 225 236 221 154 010 165 061
247 168 202 158 133 195 227 204 126 062 127 205 147 222 183 181
135 086 051 167 071 239 020 201 104 228 040 128 210 233 087 206
237 015 252 023 131 255 130 149 088 032 229 231 136 054 159 139
217 207 164 099 134 137 064 190 008 132 140 016 187 142 017 153
223 094 124 240 208 084 102 060 232 073 209 143 063 185 036 234
018 114 219 174 026 120 155 045 089 041 253 019 196 021 144 145

029 146 115 238 212 169 224 108 152 077 160 213 059 241 001 028
 148 150 095 161 214 248 235 151 098 162 103 129 116 024 004 072
 156 056 242 097 163 166 117 091 080 046 065 100 049 007 245 106
 066 197 107 030 105 173 090 186 157 246 170 002 003 069 012 068
 171 123 101 172 175 109 074 005 025 110 058 009 006 075 078 085
 118 121 200 122 125 176 038 215 042 178 216 031 249 179 180 182
 188 189 191 250 043 011 034 035 037 039 044 047 048 050 192 198

Distribution of elements in the array depends on **length** of start sequence. In case of length less than 36 bytes the order of numbers in the array may be insufficient mixed. That might harm the intenseness of permutation.

Threefold Permutation to CypherMatrix

With **start sequence** in the first round, respectively with the **matrix keys** beginning at second and further rounds and two **one-way-functions** (expansion and contraction) the procedure creates an array **variation(k)** comprising 16x16 elements, which is named: **BASIC VARIATION**.

f: Start sequence ----> BASIC VARIATION

BASIC VARIATION is a **definite** and irreversible mapping of start sequence and of respective matrix keys.

Elements in the CypherMatrix are all characters of extended ASCII-character set (ASCII-00 up to 255). Thus all numbers of BASIC VARIATION can be related to values of ASCII-characters. In each round the procedure generates the **CypherMatrix** with 16x16 elements from all characters of the array **variation(k)** in three loops (permutations).

k = Alpha

Initialisation

FOR i = 1 TO 16

FOR j = 1 TO 16

Matrix\$(1,i,j) = CHR\$(Variation(k))

BASIC VARIATION

INCR k

256 elements

IF k > 256 THEN k = 1

NEXT j

NEXT i

CypherSet\$ = ""

CypherMatrix in three loops

FOR s = 1 TO 3

(permutations)

FOR i = 1 TO 16

FOR j = 1 TO 16

a = i - j

IF a <= 0 THEN a = 16 + a

```

SELECT CASE s
  CASE 1
    Matrix$(2,a,j) = Matrix$(1,i,j)
  CASE 2
    Matrix$(3,a,j) = Matrix$(2,i,j)
  CASE 3
    Char$          = Matrix$(3,i,j)
    CypherSet$    = CypherSet$+Char$    SERIES01.RND
END SELECT
NEXT j
NEXT i
NEXT s

```

With this three loops a total **permutation** of all elements is achieved. All elements of a definite CypherMatrix are aggregated in a string **CypherSet** and stored in the file **SERIES01.RND** to result in a unlimited file of byte series.

Cypher-Matrix derivated from BASIC VARIATION (16x16 elements)

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

Blue signed characters show the **ciphertext alphabet** extracted from the CypherMatrix beginning at offset **alpha** (position 91+3=94).

Each element of the matrix has the same **probability** to be transferred to any position in the next matrix. Due to extracting matrix key from the foregoing CypherMatrix to be the start sequence in the next cycle all permutations take effect directly in the next round, and by this for all further rounds, as well.

According to the fact that all characters are passed on with the same probability very strong random effects are working on all characters distributed in the next CypherMatrix. This feature is used to generate unlimited bit and byte series.

A new CypherMatrix is created in each round. Due to principles of probability a repetition of an identical distribution of matrix elements will occur once in **256!**

(faculty) = 8E+506 cases. That should be sufficient for random performing.

Dynamic Processing Factors

In order to show the connection between **generator** and **coding sector** the generated **control parameters** are explained as follows:

Variante	: (Hk MOD 11)+1	= 7	begin of contraction (86->Dec)
Alpha	: (Hk+Hp MOD 255)+1	= 91	begin ciphertext alphabet
Beta	: (Hk MOD 169)+1	= 66	offset block key
Gamma	: ((Hp+Code) MOD 196)+1	= 150	offset matrix key
Delta	: ((Hk+Hp) MOD 155)+Code	= 31	begin of dynamic bit series
Theta	: (Hk MOD 32)+1	= 18	dynamic number series

Control parameters (alpha, beta, gamma, delta and theta) fix the positions (offsets) in the CypherMatrix from where partial sequences of elements have to be extracted necessary to perform combined applications.. For random and hash purposes only **gamma** is necessary, for encryptions in addition **alpha** and **beta** are required.

Generating Ciphertext Alphabet (128 characters)

Fixing the partial amount „**alphabet**“ (128 characters) is performed by parameter **alpha**. Certain characters (ASCII-00 to ASCII-31, ASCII-34,44 and others) are excluded because in some cases they do still their original tasks (e.g. ASCII-26) and will disturb proper performance.

n = Alpha - 1

```
FOR i = 1 TO 128
  INCR n
  IF n > 256 THEN n = 1
  Zeichen$ = MID$(CypherSet,n,1)           [Cypher-string]

  SELECT CASE ASC(Zeichen$)
    CASE <32                               [without control characters]
      DECR i
    CASE 34,44,219,xxx                       [masking out of characters]
      DECR i
    CASE ELSE                                 [ciphertext array]
      ChiffreArray$(i) = Zeichen$
  END SELECT
NEXT i
```

Ciphertext alphabet (array 128 characters): hexadecimal
Offset: Alpha: 91+3=94

```
95 59 2E D8 43 D2 B9 55 8A 56 7C A1 AF FB E3 BE
98 F6 88 F5 B6 51 61 7D 34 3C 62 6E CD BB F1 C6
F7 CF 73 2B 71 82 2D 50 F3 80 3F 4E FE 87 5E 5F
AC F4 C3 40 6C 9D 27 EC E7 C4 B4 B8 ED 72 F2 7A
35 EF 66 97 53 7F 3B 45 C0 3D D9 92 6B FA C2 9B
5B 2A E6 28 8F 74 4B D3 B5 96 65 6F 85 89 E0 BA
25 E1 E5 31 B3 52 CE 38 C8 21 47 54 EB C1 3E 8C
D5 32 A5 8B C5 BF 77 83 78 75 D7 E4 D1 81 B7 99
```

The ciphertext alphabet is needed in all encryptions procedures of **CypherMatrix** method.

Matrix Key

In first round the procedure takes the following character sequence (42 bytes) from the CypherMatrix at parameter **gamma**.

Matrix key at offset: gamma = 150 --> 42 Bytes

```
FE 87 5E 5F AC F4 C3 40 6C 9D 27 EC E7 C4 07 B4 B8 ED 72 F2 7A
35 EF 66 97 19 53 7F 10 3B 45 C0 3D D9 92 6B FA C2 FF 9B 5B 2A
```

The matrix key has to be led back to cycle's beginning in order to initialize the next run.

Block Key

For XOR-concatenations with alternatively 35 to 70 **plaintext blocks** (especially 63 bytes) the parameter **beta** fixes the position in the CypherMatrix from where a sequence in length of the **block key** is to be extracted. In first round the function generates the following block key:

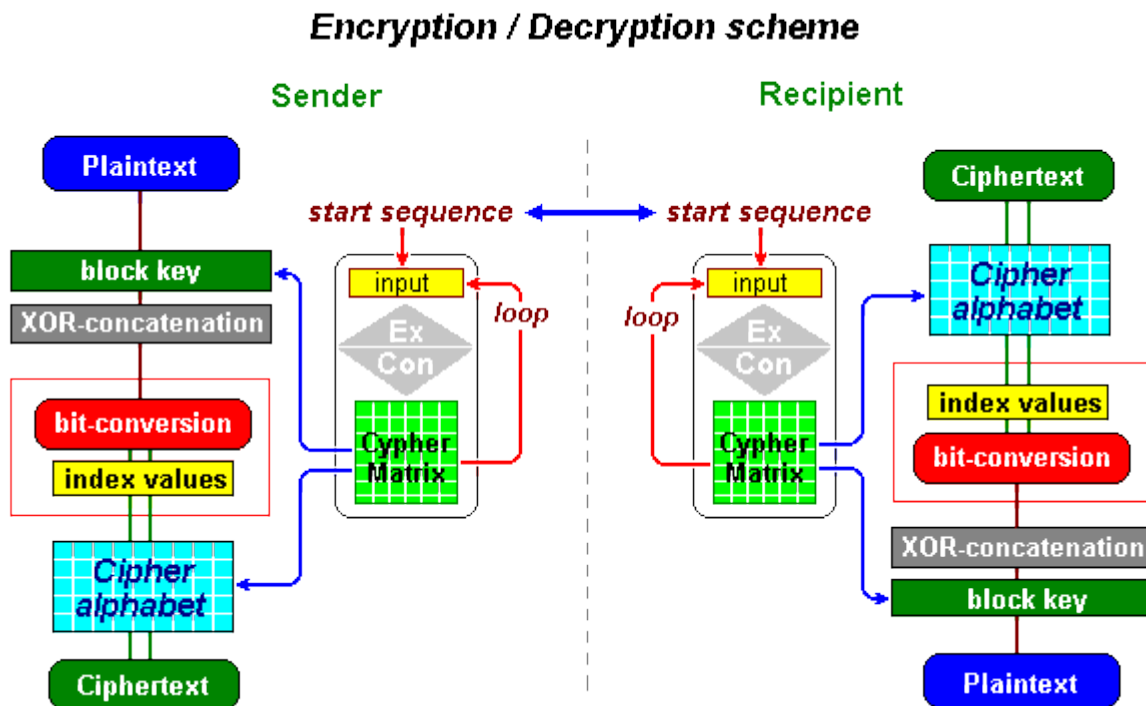
Block key (at offset: beta = 66 --> 63 bytes)

```
2F 9A 36 90 6A BC 39 FC AE A3 B0 4F C9 E8 A2 3A 2E 93 8E 01 44
70 A8 A4 EE 69 0B 16 95 59 2E D8 43 D2 B9 04 55 8A 56 7C A1 AF
FB E3 BE 98 F6 DD 88 15 F5 B6 51 0F DB 61 7D 34 14 3C 62 6E B1
```

Length of plaintext blocks – and consequently the length of block keys, as well – may be fixed alternatively with 35, 42, 49, 56, **63** or 70 bytes for each session. Block keys are necessary for encryption purposes, only.

Outlook

Working together of **generator** and **coding area** in order to achieve encryption and decryption are demonstrated by the following scheme:



In symmetric mode **sender** and **recipient** insert an identical start sequence which controls the total procedure on both sites. In each round identical **matrixes** and **control parameters** are generated. An attacker would be able to enter the procedure only if he finds the start sequence (42 bytes and not written down anywhere).

In each round **plaintext block** and **block key** extracted from the respective CypherMatrix are of same length (here chosen with 63 bytes). Plaintext and block key are XOR-concatenated. In principle that means a partial dynamic „**one-time-pad**“ even for each cycle. The block key never will be repeated because a different key is generated in each round.

By the way besides of XOR-concatenations there are a lot of further operations and applications which in combination with **CypherMatrix** as cryptographic **Data generator** resume in secret results and solutions. In case of interest you may read all articles concerned at:

<http://www.telecypher.net>

Munich, in June 2008

