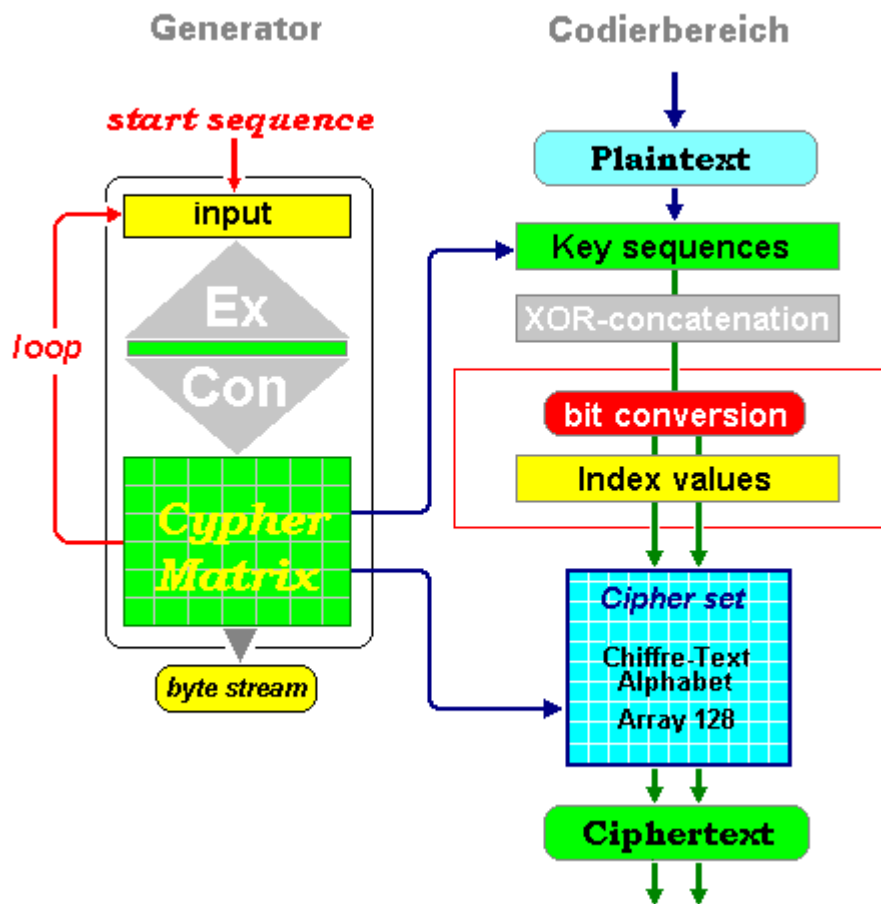


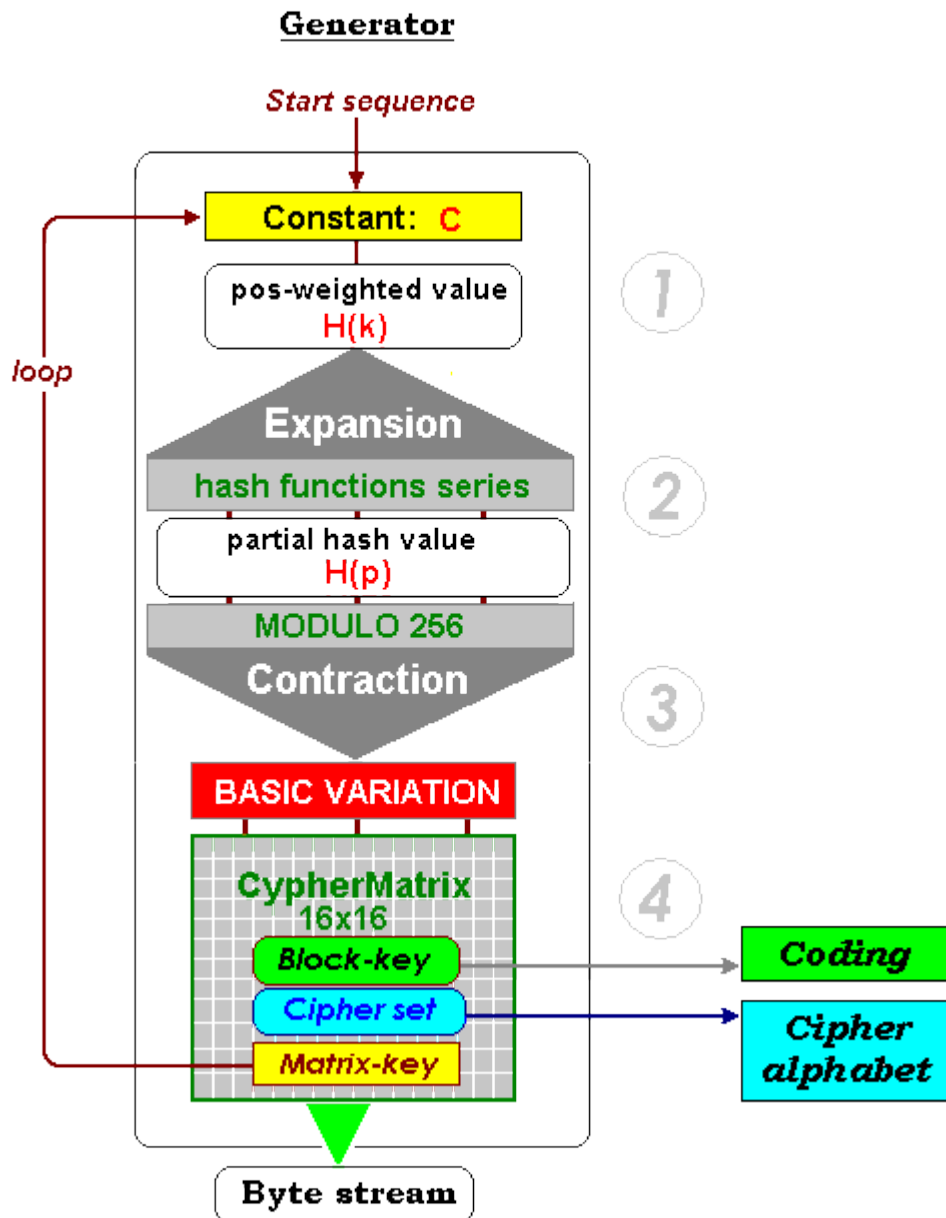
CypherMatrix

Construction of Cryptographic Basic Function

CypherMatrix operates in two areas:



Both areas will be combined together, but even can work separately from each other.



CypherMatrix as **Basic Function** serves for all control parameters which are necessary for processing cryptographic applications, especially for the **Matrix key** - a series of 42 bytes - which is led back to the begin as a new start sequence to initialize the next cycle („**loop**“).

The Generator operates in four steps:

1. Position weighted value **H(k)** of the start sequence and embedding hash constant **C(k)** for collision free performances,
2. **Expansion**: extrapolating to hash-functions-series using number system on **base 77** and calculating an interim value **H(p)**,
3. **Contraction**: condensing the hash-functions-series by Modulo 256 to **BASIC-VARIATION** and
4. threefold permutation **CypherMatrix** (final **hash value H**).

1

Erweiterung der **Start-Sequenz** zum positionsgewichteten **Zwischen-Wert H_k**

All digital Data are series of digits „0“ and „1“ (bits). Divided into 8-bit sequences (bytes) they can be denoted in decimal numbers (0 – 255) which may be calculated, converted into alternative number systems (base 3 up to base 256) and may be reassigned to index values of expanded ASCII-character set (zero to 255).

The start sequence is a series of defined bytes (a_i) with length (n).

$$\text{Sequence} = a_1 a_2 a_3 \dots a_i \dots a_n$$

To demonstrate the following explanations we choose the start sequence:

Bruno der Braunbär aus Bregenz im Breisgau

In order to assign the start sequence ($n = 42$) with a univalent value $H(k)$ each byte (a_i) has to be denoted with an index value and the single values have to be linked together in a qualified manner. As values we choose the indexes of extended ASCII-character set (zero to 255).

First assignment by addition:

$$H(k) = a_1 + a_2 + a_3 + \dots a_i + \dots a_n$$

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

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

$$H(k) = 3993$$

But the thus calculated value $H(k)$ is far away to serve as a univalent mapping of the start sequence. In order to obtain a unmistakable association of all bytes some more features have to be added

With respect to **Renè Descartes** we know that every fact – which is scaleable in its dimensions - is exactly determined by coordinates for **subject**, **location** and **time** (cartesian coordinate system) By this reason we have to consider an additional attribut for the **location**. Subject is the single byte (a_i). **Time** is relevant only if there is a functionally connection between CPU frequency and single bytes. But those connection is constant here, thus coordinate „time“ is fixed with $t = 1$.

As „location“ we state the position (**p_i**) inside the sequence of the concerned character series.

Each sequence character **a(i)** is **position weighted** by multiplying its value with its „location“ **p(i)**. Other linkages are cogitable.

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

But occurrences of collisions are not yet excluded. An additional factor has to be integrated which excludes collisions on principle. This task is done by **hash constant C(k)** (denomination by the author) which depends on length (**n**) of the start sequence and an individual user code (1 to 99), only.

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

$$C(k) = 1680 + 1 = 1681$$

In our example **code** is fixed with 1. Derivation of „hashconstant“ **C(k)** you will find in internet at:

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

With inclusion of „hash constant“ **C(k)** the interim value **H(k)** is calculated as follows:

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

$$H_k = 6798793$$

The result **H(k)** avoids collisions but is still too narrow to establish a secure unchallengeable hash value. To obtain more security an **expansion function** is introduced which widens the determining factors to a voluminous scale of digits without additional inputs and without losing the quality of being collision free.

2

Hochrechnung der **Start-Sequenz** zur Hash-Funktions-Reihe im Zahlensystem zur **Basis 77 (Expansion)** und Hash-Wert **Hp**

Expansion has the task to extrapolate the current start sequence of 42 bytes onto utmost variables (about 160 up to 2400). To achieve this the decimal values are changed into a higher number system – here into **base 77** - and the respective result **Si** will be stored in a **hash function series (r: 1 ... m)**. Simultaneous the procedure calculates the sum of all resulting values **Si** as an additional interim value **Hp**. Instead of number on base 77 other bases may be used, as well (e.g. from base 36 to base 256).

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

$$S_i \text{ ---> } d_i \quad (\text{Basis 77})$$

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

Chosen number system on **base 77** comprises the following digits:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz&#@àáâãäåæçèéë

(determined by the author, not standardized)

EXPANSION

The **hash funktion series** contains 501 digits in number system on base 77:

CèxëäibGQ2ãZàOo18âBY@1VNn#RcMoãp1xuzçM23#4Ut2kDZ1i##XbX1âfhêu3Zäl273FU
rpu4Bcè#o4E1gç@3êRM9z5qMP1k5FQâ3A1iKW#V4æ5wâu6HJçla6VhéBY1ázFWD42ëréG7
H3Btq6oâEMz746St@7DSFUa86i62F9KpOWP2hàvks8e2XTb994yPj2#06425âfKçæAUr#6
J9aâKQdA92ãd#BRéXYLAYhp8hA77écuCYdâaæ2#WT3âYæaâdYCucé77Ah8phYALYXéRB#d
ã29AdQKâa9J6#rUAæçKfã52460#2jPy499bTX2e8skvâh2PWOpK9F26i68aUFSD7@tS647
zMEâo6qtB3H7Gérë24DWFzä1YBéhV6alçJH6uâw5æ4V#WKi1A3âQF5k1PMq5z9MRê3@çg1
E4o#ècB4uprUF372LäZ3uêhfà1XbX##i1ZDk2tU4#32Mçzux1pãoMcR#nNV1@YBâ81oOàZ
ã2QGbiäèxèC

This expansion is the first **one way function** of the procedure. There is no way back to the start sequence. Besides the function brings about that the start sequence may be found in a whole effort only, if at all. Performing expansion of our chosen start sequence (42 bytes) results into the following Data:

	dezimal	Basis 77
hash constant C(k):	1681	L@
position weighted value (H _k):	6798793	EπS1
partial interim value (H _p):	588503523025	2#WECDX
total hash value (H _p +H _k):	588510321818	SIFxbqz

Control parameters derived from destination Data show as follows:

Variante	$(H_k \text{ MOD } 11) + 1$	=	2	begin of reconversion
Alpha	$((H_k + H_p) \text{ MOD } 255) + 1$	=	249	offset cipher alphabet
Beta	$(H_k \text{ MOD } 169) + 1$	=	93	offset block keys
Gamma	$((H_p + \text{code}) \text{ MOD } 196) + 1$	=	7	offset matrix key
Delta	$((H_k + H_p) \text{ MOD } 155) + \text{code}$	=	144	dynamic bit series
Theta	$(H_k \text{ MOD } 32) + 1$	=	10	dynamic number series

In defining of **Gamma** (offset of matrix key) and **Delta** the chosen personal **code** is involved. Thus, inspite of equal start sequences up to 99 different processes are possible. Because since 2nd round start sequences (matrix keys) are generated anew in each round there will be different control parameters in each cycle, as well.

Single digits of **hash function series** are calculated in extracts as follows:

char	a _{i+1}	p _i	(a _{i+1})*p _i	H _k	S _i	base 77
B	67	11	737	6798793	5010710453	1àfhêu
r	115	12	1380	6798793	9382334353	3ZäL27
a	98	13	1274	6798793	8661662296	3FU rpu
u	118	14	1652	6798793	11231606051	4Bcè#o
n	111	15	1665	6798793	11319990361	4E1gç@
b	99	16	1584	6798793	10769288129	3êRM9z
ä	133	17	2261	6798793	15372070991	5qMP1k
r	115	18	2070	6798793	14073501529	5FQâ3A
...
				Summe H _p	588503523025	2#WECDX

With intend to get better manageable Data and regain decimal values the procedure performs as a third step a **contraction** of the digits into number system on base 10 (decimal numbers).

3

Verdichtung der Hash-Funktions-Reihe durch Modulo 256 zum Array BASIC-VARIATION (Contraction)

To reduce variables of **hash function series** to decimal data, each three digits of the function series – beginning at parameter **variante** – are reconverted by **MODULO 256** to decimal numbers **0** to **255** (without repetition). From the result variable **Theta** is subtracted and the remaining value is stored as **BASIC VARIATION** in an array of **16x16** elements. To achieve the reverse conversion the hash function series is assumed to be a series of digits in number system on **base 78** (expansion base **77 + 1**). Compared with number on base 77 in number base 78 there is one more digit but that does not harm the procedure.

This contraction is the second **one way function** of the procedure. The function is not revisible and retrograde destination of hash function series is not possible. The contraction process can be performed alternatively with other MODULO factors (2 to 256, especially with 8, 16, 24, 32 and 64). Even customary dynamic **S-boxes** may be generated with this function.

In pseudo code demonstrating as follows:

```

p = Variante
FOR k=1 TO 256
  Zahl = MID$(Reihe, p, 3)
  CALL SystemNachDez (78, Zahl, Dezimal)
  Element = (Dezimal MOD 256)
  INCR p
  Variation(k)=Element
  IF k>1 THEN
    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
Index = Variation(k) - Theta
IF Index<0 THEN Index = 256 + Index
Variation(k) = Index

```

triple-digit number base 78
reconversion
limitation on 0 to 255

BASIC-VARIATION: 256 elements

With parameter **variante = 2** the reversion begins at second position of hash function series.

In our example the process develops in extracts as follows:

CèxëäibGQ2äZäOo18âBY@1VNn#RcMoãp1xuzçM23#4Ut2kDZ1i##XbX

3 digits base 78	decimal	modulo 256	- Theta	element
èxë	448810	42	10	32
xëä	364953	153	10	143
ëäi	467810	98	10	88
äib	423265	97	10	87
ibG	270598	6	10 (256-4)	252
bGQ	226382	78	10	68
GQ2	99374	46	10	36
Q2ä	158408	200	10	190

BASIC-VARIATION (256 elements)
Distribution of elements

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



dreifache Permutation der BASIC-VARIATION zur CypherMatrix als finaler Hash-Wert H

BASIC VARIATION is a unique and not reversible mapping of the start sequence resp. of the adjusted matrix key. Values of BASIC VARIATION can be related directly to indexes of ASCII-character set because all its elements comprises signs of extended ASCII-values, as well. During each round the procedure generates from all elements of BASIC VARIATION in three loops the **CypherMatrix** with 16x16 elements.

Pseudo code demonstrates as follows:

```

k = Alpha
FOR s = 1 TO 3
  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$(1,i,j) = CHR$(Variation(k))
          INCR k
          IF k>256 THEN k = 1
        CASE 2
          Matrix$(2,a,j) = Matrix$(1,i,j)
        CASE 3
          Matrix$(3,a,j) = Matrix$(2,i,j)
          CypherSet$ = CypherSet$ + Matrix$(3,i,j)
      END SELECT
    NEXT j
  NEXT i
NEXT s

```

Three loops achieve a complete **permutation** of all elements. All characters are aggregated in string **CypherSet** and stored in file **SERIES01.RND**.

CypherMatrix (16x16) derivated from BASIC VARIATION (256 elements)

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

Each round generates a new CypherMatrix. A repetition of identical structures due to probability law first occurs in **256!** (faculty) = **8E+506** cases.

CypherMatrix is a unique mapping of the start sequence. It is free of collisions, as well. The result implements all features of a hash function and CypherMatrix in its structure represents a final **hash value H**.

If the structure of the matrix represents a unique hash value then all single **rows** and **columns** corresponds to properties of a hash value, as well.

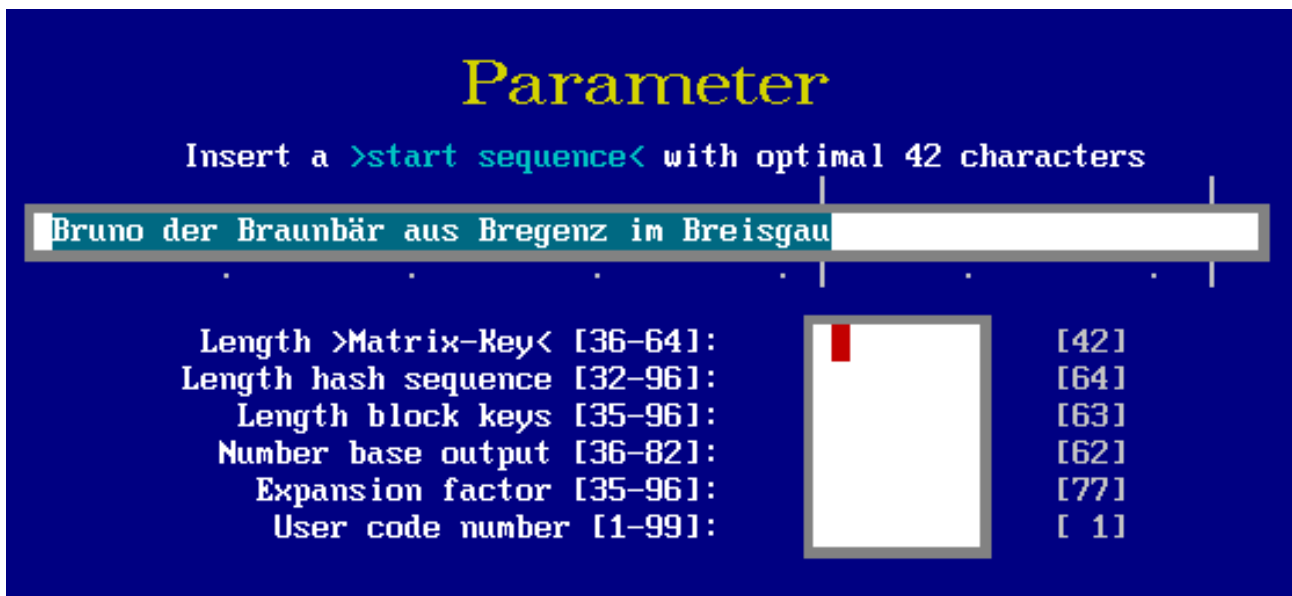
Hash-value: line 7 (CM-final hash)
2C 68 9D 2D 0D 91 32 C4 5C FF D1 3B FC A7 16 DE

Hash-value: column 12 (CM-final hash)
25 9F 01 E6 D5 F8 3B 06 57 63 CC A1 CB 2B 82 0C

During permutations generating the CypherMatrix each element has the same probability to be regrouped to an alternative position of the next matrix. Because **matrix key** is derivated from the foregoing round and is used as a **start sequence** for the following cycle all permutations make an impact on all following rounds and by this on the total procedure. This feature is basis for generating a unlimited **byte stream**.

Key Parameter

Start sequence as a master key controls the whole procedure. In order to combine further applications with the Basic Function additional **key parameters** are necessary. The following screen shot shows the resp. parameter being available at combined applications.



Predefined parameters in any program are listed at the right column. They may be replaced with own Data.

Following wingspans are defined for the respective parameters:

length of >Matrix key< :	36 – 64 bytes
length of hash sequence :	32 – 96 bytes
length of block key :	35 - 96 bytes
Number system for Data output :	36 – 82 basic digits
Number system for expansion function :	35 – 96 basis digits
individual user code :	1 – 99 numbers

Parameters actually being available will be shown at the following declarations of single applications.

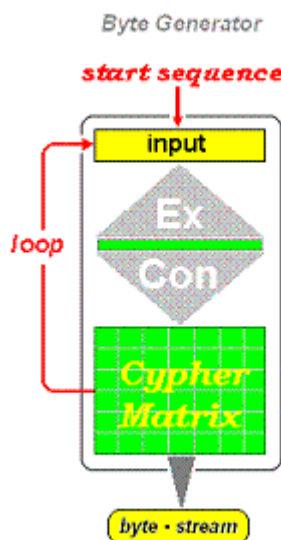
Applications using the Basic Function

Combined with **Basic Function** in order to solve cryptographic problems especially the following applications have been developed:

Generating of unlimited dynamic byte series (byte generator),
calculating of serial and final hash values (CypherMatrix Hash),
simple and extended signature (telePortal) and
all kinds of encryptions with varied operations.

Byte Generator

The **byte generator** produces unlimited **bit-** and **byte series** (byt stream).



Byte series can be performed optionally in 16-bit, 32-bit, 64-bit or alternatively bit series in variable length.

Variante, gamma, delta and **theta** serve as control parameters, only.
Alpha and **beta** are not needed. Following **key parameters** are embedded:

Length of >Matrix Key< : 36-64 bytes
Number system for expansion function : 35-96 basis digits
Individual user code : 1-99 numbers

All elements of the respective CypherMatrix with 256 bytes in order of its structures will be written line by line into a file (SERIESxx.RND). Length of the series depends on number of rounds to be inserted.

If otherwise **contraction** is performed with **MODULO 8** (alternatively with **16, 32, 64** or others) instead of MODULO 256 (contraction to BASIC VARIATION) any number series may be created, for instance for encryption operations. Further details you will find in internet at:

telecypher.net/CORECYPH.HTM#Z15

Dynamic Hash Function

Next a **dynamic hash function** to designate digital informations in a unique value is introduced.

Digital strings (files) are divided into plaintext blocks (e.g. 64 bytes), which in each cycle will be sequentially position weighted, multiplied with the hash constant **Ck**, expanded to hash function series, again contracted to BASIC VARIATION and finally subjected to a threefold permutation. The resulting last CypherMatrix with 256 elements represents the **Hash Value (H)**. An identical hash value will occur first in **256!** (faculty) = **8E+506** cases.

Compared with current hash functions the procedure has some advantages:

1. Including of a **compression** is not necessary,
2. only bytes and simple mathematics are implemented,
3. with the results of the function (hash values) can be **calculated** (e.g.. addition, subtraktion, multiplikation, division und modulo calculation) and
4. the results are remarkable **shorter** than 128 bit resp.160 bit (SHA, MD5, RIPE-MD).

Applied to the identification Data of the Authors the following results come up:

CypherMatrix Hash base 62: **dnjpkSi7e**
 base 16: **1EE033B3567631**
 base 10: **8 690 761 958 061 617**

SHA 1:
47 19 58 FF 6D 10 F5 CD 85 A3 0F 56 61 E9 E2 BE 0C 77 82 C4

MD 5:
D9 B3 04 D2 E1 14 95 20 47 89 69 C1 8B DA FE 50

In a **RFID chip** stored the hash value can be [selected and adjusted](#) mobilely. Even if the Data are changed only in **one bit** (at last position number 9 is changed to number 8)

9 = 111 001
8 = 111 000

then the following results show up:

CypherMatrix Hash base 62: **gLjIviYBm**
 base 16: **20D9ED1E651012**
 base 10: **9 246 811 695 157 266**

SHA 1:
95 99 E1 4E 95 DA D9 DD E9 BA F6 92 BB 5F BC 99 29 49 CE 3E
MD 5:
AB DA 29 40 D2 63 F8 D0 04 F0 82 42 14 04 06 AE

If the CM-Hash value is to be seen as **finger print**, then SHA and MD5 are a „**Tatzelwurm**“ !

With the CM-Hash value of the average 9 to 11 indications in the number system on **base 62** altogether different values without collisions can be represented up to **$62^{11} = 5,2E+19$** .

In case SHA 1 it will be **$2^{160} = 1,5E+48$** and at MD5 **$2^{128} = 3,4E+38$** values..

On principle two techniques are possible:

- a) Serial calculation of partial hash values from the CypherMatrix in each round and accumulating to a hash value ad end (**serial mode**) and
- b) final calculation of the hash value from the last CypherMatrix beyond all foregoing cycles passed (**final mode**)

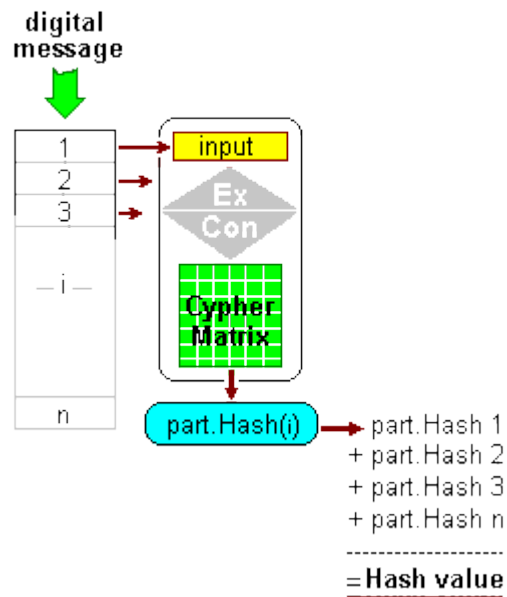
As key parameters are necessary:

Length of >Hash Sequence< : 32-96 bytes
Number system for Data output : 36-82 basis digits
Number system for expansion function : 35-96 basis digits
Individual user code : 1-99 numbers

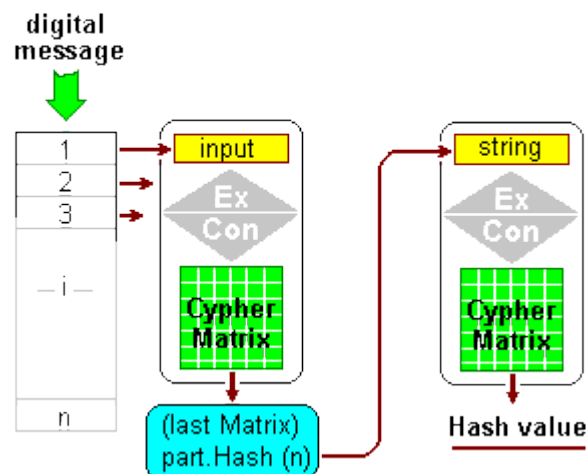
>Hash Sequence< takes the place of >Matrix Key<.

Connections are shown by the following schemes:

Serial mode



Final mode



Development and working features of **CypherMatrix Hash Function** are detailed demonstrated in the following pdf-files:

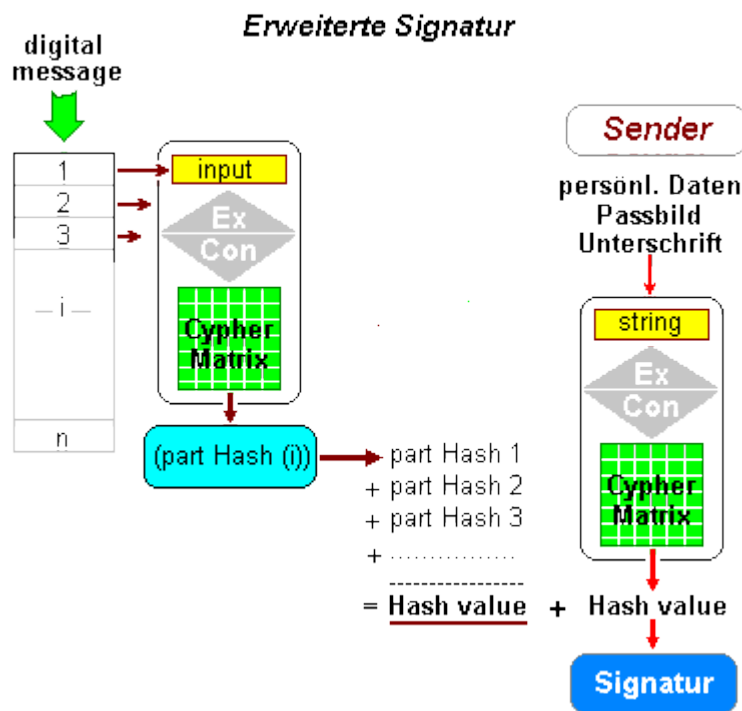
[Dynamische Hash Funktion](#)

Extended digital Signature

If the hash function is applied to both the **message** to be **signed** and the **identification data (ID)** of the signatory (including personal photo and signature) an **„extended digital signature“** can be created.

As key parameters are necessary:

- Length of >Hash Sequence< : 32-96 bytes
- Number system for Data output : 36-82 basis digits
- Number system for expansion function : 35-96 basis digits
- Individual user code : 1-99 numbers



In addition to the personal Data of the signatory his digitally photo and his digitally signing can be included into calculation of the hash value. A recipient of the signature will be able to verify the integrity of the message and the identity of the sender (program „**telePort**“). Further details you may find in internet in the file: [Eine einfache digitale Signatur](#)

Digital Encryptions

The CypherMatrix procedure performs encryptions with a multiple number of operations and combinations:

- XOR-concatenation
- substitution „dyn24“
- bit conversion
- bit exchange
- structure changing
- number system base 4
- bit crossing
- block transposition
- number system base 256

In each program the following key parameters are necessary:

- Length of >Matrix Keys< : 36-64 bytes
- Length of Block Keys : 35-96 bytes
- Number system for expansion function : 35-96 basis digits
- Individual user code : 1-99 numbers

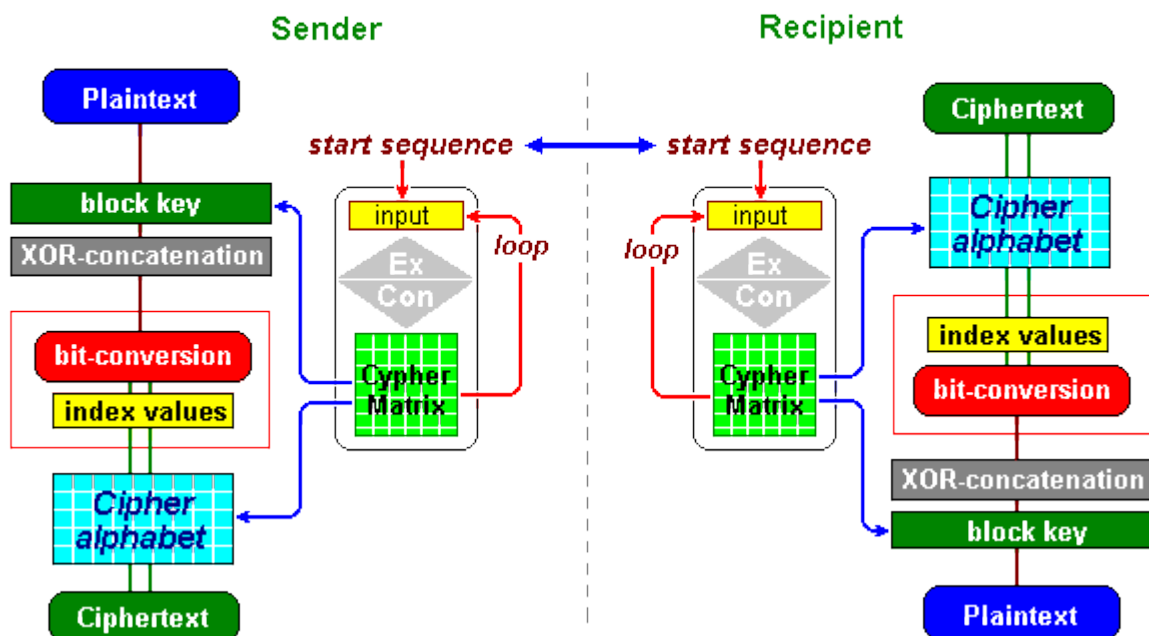
Following scheme shows the combined working of **generator** and **coding area** in performing several operations

As in a symmetric procedure both **sender** and **recipient** insert an identical **start sequence** which constructs and controls the whole process. This passphrase has to be easy to remember and possibly should be catchy and even of funny words which one can easily keep in mind and which may not be written down anywhere. Here are some examples:

Sven Hedin is sailing around the North pole
Kangaroos jumping in the Hills of Amarillo
Blue flamingos flying to Northern Sutherland
Im Steinhuder Meer wurden 70132 Heringe gezählt

Start sequences should be about **42 bytes** long hence 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 can be found in total only, if at all.

Encryption / Decryption scheme



Changing of **one bit** only blocks out the whole function and leads to quite a different result.

Performing of Encryption

Encryption in each round takes place by the following four steps:

1. Generating the actual CypherMatrix (round matrix) either
 - a) by inserted start sequenz (first round) or
 - b) by matrix key derivated from the foregoing round matrix (begin at second round)
2. derivating cipher alphabet (128 characters) from the current round matrix beginning at control parameter alpha
3. derivating block key (63 characters) from the current round matrix beginning at control parameter beta and
4. performing extracted Data according to the chosen operation or combination, resp. bit conversion.

Generating Matrix Key

In the first round of our chosen example the procedure derivates from round matrix beginning at control parameter **Gamma** the following character sequence:

Matrix Key (at offset: Gamma = 7)

ë#oK5%|Vâ¾a?©ï#^0<#PÏÿJ`#ÇÁfz(jÚ#i£¤s#¬!\$^

EB 15 6F 4B 35 25 7C 56 E2 BE 61 3F A9 EF 1F 5E 30 8B 17 50 CF
9F 4A 60 07 C7 C1 66 7A 28 6A DA 14 EC A3 A4 73 01 AC A6 24 88

The matrix key will be led back to start of the procedure (**loop**) and serves to initialize the next round.

Creating Cipher Alphabet

Destination of subset „**alphabet**“ (128 characters) takes place at control parameter **alpha**, here at offset = 249:

Ciphertext alphabet: ASCII-character set

Offset: alpha = 249

Index 1 - 16: ¿ ... ™ k S × ' õ , g μ ë o K 5 %
Index 17 - 32: | V â ¾ a ? © ï # ^ 0 < # P Ï ÿ J `# Ç Á f z (j Ú # i £ ¤ s # ¬ ! \$ ^ †
Index 33 - 48: É f / ÷ û m b Î » è æ Ö B À d v
Index 49 - 64: œ T Œ w @ Ö . D x E i t ~ ” =
Index 65 - 96: & [ª ø 4 å Ê , h □ - ' 2 Ä \
Index 97 - 112: Ñ ; ü § % º é Å ú “ • O Õ " C É _
Index 113 - 128: y R Ž) ã q ó : ¥ W n M 6 ' * }

Ciphertext alphabet: hexadecimal

BF 85 99 6B 53 D7 92 F5 B8 67 B5 EB 6F 4B 35 25
7C 56 E2 BE 61 3F A9 EF 5E 30 8B 50 CF 9F 4A 60
C7 C1 66 7A 28 6A DA EC A3 A4 73 AC A6 24 88 86
C8 83 2F F7 FB 6D 62 CE BB E8 E6 D6 42 C0 64 76
9C 54 8C AD 77 40 D5 2E 44 78 45 69 74 7E 94 3D
26 5B A0 AA F8 34 E5 CA 2C 68 9D 2D 91 32 C4 5C
D1 3B FC A7 89 E9 C5 FA 93 95 4F D4 22 43 C9 5F
79 52 8E 29 C3 71 F3 3A A5 57 6E 4D 36 27 2A 7D

During extraction of characters destined elements (ASCII-00 to ASCII-31, ASCII-34, 44 and others) are excluded because in certain circumstances they do their primarily tasks (e.g. ASCII-26) and produce somewhat confusion.

Generating Block Key

To achieve XOR-concatenations with plaintext blocks of chosen 63 characters the control parameter **beta** fixes a position in the round matrix from where a sequence of chosen length as block key is to be extracted.

Block key (at offset: Beta = 93 --> 63 bytes)

4â±Ê,h□-.‘2Ä\□Ñ;üş#P%œÁú“•O#Ô"#CÉ_yR#Ž#)ÝÃqó:¥WnM6"*}i<G,,1Í€#

34 E5 B1 CA 2C 68 9D 2D 0D 91 32 C4 5C FF D1 3B FC A7 16 DE 89
E9 C5 FA 93 95 4F 00 10 D4 22 06 43 C9 5F 79 52 11 8E 0B 29 DD
C3 71 F3 3A A5 57 6E 4D 36 27 2A 7D ED 3C 47 84 B9 CD 49 80 05

Block keys are necessary for encryptions only. Because plaintext blocks and block keys always are of equal length (e.g. 63 bytes) they form an partial „one time pad“

Bit Conversion

Bit conversion as a new and important encryption step comprises data technically a change in the number of bits in that respective sign. The series of 8-bit sequences – for example as result of XOR-concatenations - are regrouped into 7-bit segments (0-127), which serve as indexes (+1) for 128 characters in the cipher alphabet. No bit is added and not bit is removed. The rows of digits „0“ and „1“ remain identical. A new grouping (8-bit → 7-bit) is arranged, only.

As an example:

8-bit XOR-sequences rearranged into 7-bit sequences as "index data"

8-bit: 11101111100001111001101010101111101010011101111101000100
7-bit: 11101111100001111001101010101111101010011101111101000100

A detailed explanation of bit conversion you will find in pdf files

XOR und Bit Conversion Ergänzung zur Bit Conversion

Due to bit conversion ciphertext compared with plaintext lengthens at ratio **7:8**. The functional connection between plaintext and ciphertext is disconnected. This is of fundamental influence on **cryptanalyse** of the procedure. All present methods of attacks – except „brute force“ - presuppose equal length of plaintext and ciphertext. Each plaintext character has to be related to a definite ciphertext character. But CypherMatrix does not include the usually **congruence of length**. One plaintext character is related to **8/7** ciphertext characters.

Thus, all todays customary attacks will not work on CypheMatrix bit conversion encryptions. Even „brute force“ will have no success. You may read the proof in:

Cryptanlys of the procedure

Munich, in Septembre 2008
Ernst Erich Schnoor