>CypherMatrix< as dynamic Hash Function

(Ernst Erich Schnoor)

The essential task of a hash function is to perform digital characters of arbitrary length (n) to a unique digital output (H) of specified term [#2]. This can be a single **value**, a unique **string** (vector) or a definite **structure** (e.g. matrix). By means of **Basic Function** a dynamic hash procedure is created, which can be used in various areas of cryptography. You may read detailed explanations in article: Cryptographic Basic Function in Byte Techniques. Following tract considers last developments of the procedure. Superseded solutions are not demonstrated any longer.

1 Hash Generator

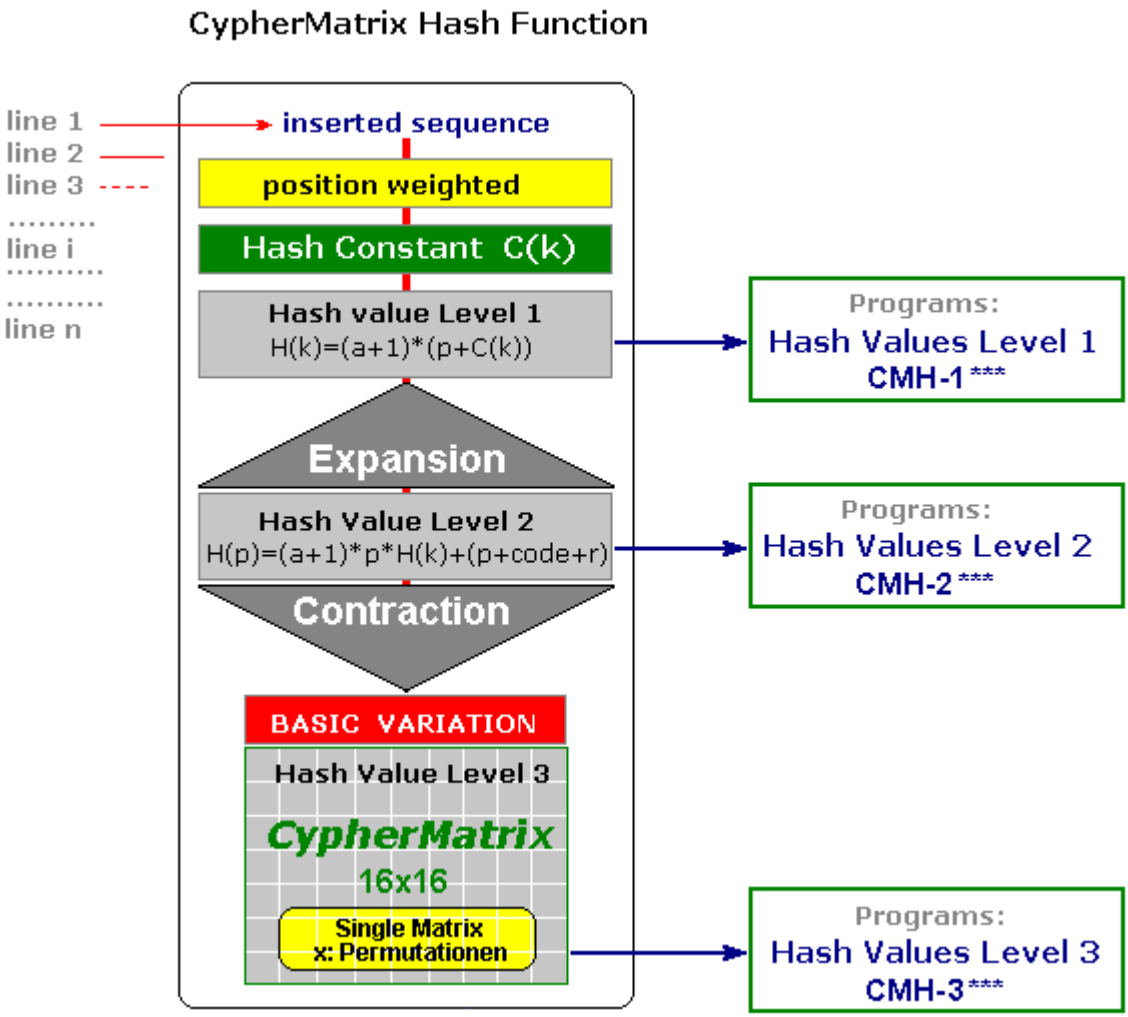
The **Basic Function** forms the core of the hash prodecure [#1]. Digital sequences (files) are devided into plaintext blocks (e.g. 64 bytes), which in each cycle will be sequentially position weighted, multiplied with hash constant Ck, expanded to hash function series, again contracted to BASIC VARIATION and finaly subjected to threefold permutation.
A key is'nt necessary. The first plaintext block is used as start sequence.

In the course of the function there are three sections which generate a definite and collisionfree mapping of the inserted plaintext series.and therefore meet all preconditions as a **hash value**:

1. Position weighted and free of collisions (level 1),
2. Extrapolating to hash function series (level 2) and
3. BASIC VARIATION with generating CypherMatrix (level 3).

The last CypherMatrix with 16x16 elements results in its structure to an definite and collisionfree expression. Transformation to a concise term produces the searched **hash value.** Due to probability law a repetition of identic structures will occur first in **256!** (faculty) = **8E+506** cases.

The following scheme demonstrates the connections:

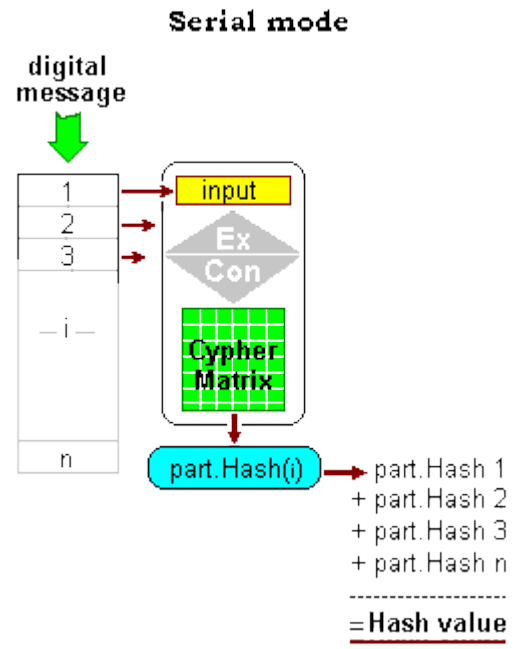


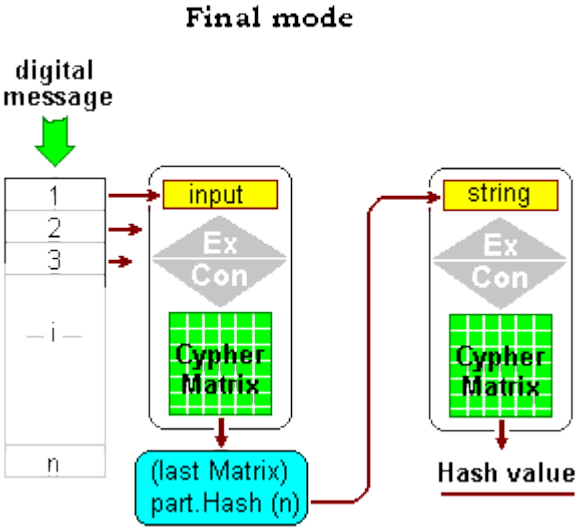
The dynamic hash function works in four steps:

1. Position weighted value **H(k)** of inserted digital hash sequences with including constant **C(k)** (collision free) to generate **Hash Values Level 1,**
2. **Expansion**: extrapolating to a hash function series of about 160 to 2400 digits in number system on **base 77** – first one way function - and calculating an intermediate value **H(p)** to generate **Hash Values Level 2,**
3. **Contraction**: condensing the hash functions series by Modulo 256 to an array of 16x16 elements: **BASIC-VARIATION** –second one way function - and
4. alternative threefold permutation of BASIC-VARIATION to generate the **CypherMatrix** (16x16 characters) as final **Hash Value Level 3.**

Two techniques of hash processing are possible:

a) Serial calculation of partial hash values from digital sequences in each round and accumulate to a hash value (**serial mode**) and
b) final calculation of the hash value from the last CypherMatrix beyond all foregoing partial hash values are added and passed (**final mode**).





1.1 Determination Base for Hash Values

The text file CANNERY.TXT (401 bytes) serves to explain the working steps:

> ***The WORD is a symbol and a delight which sucks up men and scenes, trees, plants, factories, and Pekinese. Then the Thing becomes the Word and back to Thing again, but warped and woven into a fantastic pattern. The word sucks up Cannery Row, digests it and spews it out, and the Row has taken the shimmer of the green world and the sky-reflecting seas.***
>
> John Steinbeck, Cannery Row, New York 1945

Inserting of a separate key sequence to initialize the procedure is not neccessary. The first line of the text to be hashed in chosen length (64 characters) will be subjected to the procedure as first round ($\mathbf{r}_1$). Then the hash value $\mathbf{H}(\mathbf{r}_1)$ of the inserted sequence will be calculated.

The procedure runs the following steps:

1.2 Position weighting

A definite mapping of the inserted sequence as determination base for calculating hash values has to be searched for. As an entry into searching an addition of characters (bytes) is performed.

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

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

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

$$H(k) = 5786$$

But the thus calculated value **H(k)** is far away to serve as a determination base for calculating hash values. To achieve definite results some more features have to be added, especially **position weighting** and **collision free** device**.**

Each character **a(i)** is **position weighted** by multiplying its value with its location **p(i)** [#3]. Time is not relevant.

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

1.3 Collision free

Collisions are avoided by including the **hash constant C(k)**. The constant is defined by the length (**n**) of the input sequence and an individual code (1 to 99).

$$C(k) = n * (n - 2) + code \qquad \text{code} = 1$$

$$C(k) = 64*62 + 1 = 3969$$

Derivation of **„hash constant" C(k)** you will find in internet at:**"Determinats leading to Collisionfree"** [#5]

With inclusion of „hash constant" **C(k)** the interim value **H(k)** will be calculated as follows:

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

$$H_k = 23158479$$

2 Hash Values Level 1

Determination value **H(k)** in fact is still somehow small comparatively but includes all features of a hash value: the result is definite and collisionfree. The determination formula establishes the hash calculation of the first level: Program: **CMH-1 fmx.exe**

CMH-1 fmx / 64 / 62 / 77 / 1
Serial Weighted Values in cannery.txt

decimal	base 62	cycle
23164265	1ZC5B	1
22832653	1Xnob	2
22809255	1XhjD	3
23390333	1a8tR	4
23158585	1ZAbZ	5
22567499	1Wgpv	6
357357	1Uxp	7

decimal: 138279947
base 16: 83DFC0B
base 62: **9MCvj**

The sums of all serial results in all three terms (decimal, hex and base 62) und the number of rounds form a new sequence (**R**) to be inserted as an additional round to eastablish the **Hash value Level 1:**

Input string (**R**) = *base 62 + decimal + base 16 + rounds*
Input string **(R)** = *9MCvj13827994783DFC0B7*

The calculation leads to the following hash value:

Matrix value last cycle:
decimal: 9828697491
base 16: 249D60593
base 62: AjADCt

Final Hash value for: CANNERY.TXT
decimal: 9966977438
base 16: 25214019E
base 62: **AsWQ8c**

3 Hash Value Level 2

To enlarge the determiniation base **Hp** a major formula including the hash constant **H(k)** is introduced, which widens the determination factors to a higher combination of digits.

$$\mathbf{Hp} = \sum_{i=1}^{n} (a_i + 1) * p_i * H_k + p_i + \mathbf{code} + r_j$$

$$\mathbf{Hp} = 4490276951133$$

Results of **Hp** cover a range which seems to be qualified for calculating hash values (level 2). Results of the determination formula are definite and collision free. The program **CMH-2 fmx.exe** generates hash values of level 2 as follows:

```
CMH-2 fmx / 64 / 62 / 77 / 1
Serial Weighted Values in Cannery.txt
-------------------------------------------------
     decimal            base 62  cycle

  4490276951133     1H3KyTwT    1
  4338455231455     1ENcK1CJ    2
  4304448559661     1DmUtUNx    3
  4103903899388     1AFasX3E    4
  4373846158489     1F0FQk25    5
  3795656615396     14p80LGG    6
  3925924291        4HgmFX      7
-------------------------------------------------------
decimal: 25410513339813
base 16:   171C589F05A5
base 62:         7DMiCO7p
---------------------------------
```

The sums of all serial results in all three terms (decimal, hex and base 62) und the number of rounds form a new sequence (**R**) to be inserted as an additional round to eastablish the **Hash Value Level 1:**

Input string (**R**) = *base 62 + decimal + base 16 + rounds*
Input string **(R)** = *7DMiCO7p25410513339813171C589F05A57*

The calculation leads to the following hash value:

```
Matrix value last cycle:
decimal:      88764387705
base 16:      14AAC51179
base 62:            1YtCK8f
---------------------------------
Final Hash value for: CANNERY.TXT
decimal: 25499277727518
base 16:   17310364171E
base 62:         7EvbOiGU
====================
```

4 Hash value Level 3

To condense the determination base an **expansion** (**hash function series**) is introduced which widens the determining factors to a voluminous scale of digits without additional inputs and without loosing the quality of being collision free.

4.1 Expansion

The number system of expansion can be chosen between 64 up to 96. Here base 77 is fixed. The procedure expands each sequence character to decimal value (si) which is changed to (di) a digit in number system on base 77 and combined it to hash function series (i: 1 ... m).

Simultaneously the procedure calculates the sum of all single results (si) as an additional value H(p) in order to fix various destination Data.

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

$$s_i \longrightarrow d_i \quad (base\ 77)$$

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

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

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

$$Hp = 4490276951133$$

Number system on base 77 comprises the following digits:

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

Performing expansion of chosen input sequence (64 bytes) results to the following determination Data:

	decimal	base 77
hash constant C(k):	**3969**	**pg**
position weighted value (H_k):	**23164265**	**outl**
partial serial value (H_p):	**4490276951133**	**Lfä7Apo**

Only **Variante**, **Alpha** and **Theta** are necessary for further researching. The other parameters are not needed.

Variante	(H_k MOD 11) +1	=	**4**	begin of reconversion
Alpha	((H_k + H_p) MOD 255) +1	=	**163**	offset initializing
Theta	(Hk MOD 32) +1	=	**16**	dynamic number series

By generating hash function series the sequence " **symbol** **"** at position 15 of the input sequence results to the following calculation:

char	pi		Hk	(ai+1)*pi*Hk		Si	base 77
	(ai+1)	(ai+1)*pi			pi+code+r		
s	116	15 1740	23164265	40305821100	17	40305821117	Eãiw&N
y	122	16 1952	23164265	45216645280	18	45216645298	GsLftt
m	110	17 1870	23164265	43317175550	19	43317175569	G0IäUq
b	99	18 1782	23164265	41278720230	20	41278720250	FJJ#Vê
o	112	19 2128	23164265	49293555920	21	49293555941	IGJrrE
l	109	20 2180	23164265	50498097700	22	50498097722	IoeBtç
					Summe:	4490276951133	Lfä7Apo

The hash function series **H(p)** results in 383 digits in number system 77 as follows:

u0áBæ1zTN2I2lnMgà19êlao3wçTåp48Mén74éàoâw4tvGf52fsn5X95bãiDAå#mK23TçàW
äAädBS@3èXopGEãiw&NGsLfttG0läUqFJJ#VêlGJrrEloeBtç5æobcNIYsV7OLàNëZoKvN
j&æ74n7éæL&1Mvn7m9åGIOFeRyiPOEcSçRêyHjUS9P4qDSb09L#ToLXaZY3O&Hk9ã7BCba
éqtuTXJ32eâYaKyTDXSæ9q4ZçkêiEBihhâWfrWçKUhWfOvsboUãDOfje4è8jpE5&pDL2ëj
RmaOn8älTmtRUE9LFç#m0usEêjU7VnGoQmjmnFJJ#WYk9x0uvrF4ànçnKkV&yGTIXä5whã
KàNpQu8éBrJ1MNpwãæSqfsëYBqG#f6EyU

4.2 Contraction

In order to lead back the determination base to decimal dimensions next step the hash function series **H(p)** will be contracted by **modulo 256** to array **BASIC VARIATION** with 16x16 decimal digits. By this the hash function series is assumed to be a series of digits in number system on **base 78** (expansion base 77+1).

First reconversions beginning at **variante** = **4** show as follows:

...Bæ1zTN2 …....

3 digits base 78	decimal	modulo 256	- Theta	element
Bæ1	72463	15	10	05
æ1z	432103	231	10	221
1zT	10871	119	10	109
zTN	373409	161	10	151
TN2	178232	56	10	46

BASIC VARIATION results to the following structure:

BASIC-VARIATION (256 elements)

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

Numbers of BASIC VARIATION (16x16) form the base for further hash calculation.

4.3 Generating CypherMatrix (CM)

Decimal values are related directly to indexes of ASCII-character set and by this to the first **CypherMatrix** (**CM1**) with 16x16 elements.

Derivation first CypherMatrix (CM1) from BASIC VARIATION

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

The elements of the first CypherMatrix (**CM1**) can be used directly as determination base for calculating hash values. But to increase security an additional threefold permutation is introduced which results in **CypherMatrix** (**CM3**). To perform the permutation three variants (A, B and C) are possible.

Pseudo code demonstrates as follows:

4.3.1 Simply replacing index: i (variant A)

```
CM1Set$ = ""                    elements of first CypherMatrix (CM1)
CM3Set$ = ""                    elements of third CypherMatrix (CM3)


FOR s = 1 TO 3                                          three loops
    FOR i = 1 TO 16                                    (permutation)
        FOR j = 1 TO 16
           a = i - j
           IF a <= 0 THEN a = 16 + a
           SELECT CASE s
              CASE 1
                 CM1Set$ = CM1Set$ + Matrix$(1,i,j)  CypherString
                 Matrix$(2,a,j)  = Matrix$(1,i,j)
              CASE 2
                 Matrix$(3,a,j)  = Matrix$(2,i,j)
              CASE 3
                 CM3Set$ = CM3Set$ + Matrix$(3,i,j)  CypherString
           END SELECT
        NEXT j
    NEXT i
NEXT s
```

4.3.2 Displaced changing of indexes: i,j (variant B)

```
a = i - j
IF a <= 0 THEN a = 16 + a
SELECT CASE s
   CASE 1
      CM1Set$ = CM1Set$ + Matrix$(1,i,j)  CypherString
      Matrix$(2,a,j)  = Matrix$(1,i,j)
   CASE 2
      Matrix$(3,i,a)  = Matrix$(2,i,j)
   CASE 3
      CM3Set$ = CM3Set$ + Matrix$(3,i,j)  CypherString
END SELECT
```

4.3.3 Dynamic generating of indexes: i,j (variant C)

All index values (16x16) are generated anew in a separate index-array **Index$(16,16)** (application of CypherMatrix function):

```
a = i - j
IF a <= 0 THEN a = 16 + a
SELECT CASE s
   CASE 1
      CM1Set$ = CM1Set$ + Matrix$(1,i,j)  CypherString
      Matrix$(2,a,j)  = Matrix$(1,i,j)
   CASE 2
      x = VAL(Index$(i,j))
      Matrix$(3,i,x)  = Matrix$(2,i,j)
   CASE 3
      CM3Set$ = CM3Set$ + Matrix$(3,i,j)  CypherString
END SELECT
```

The third (final) CypherMatrix (**CM3**) is deduced from first CypherMatrix (**CM1**) as follows:.

Generating third CypherMatrix (CM3) by threefold permutation (variant B)

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

The **structure** of CypherMatrix represents a definite mapping of the input sequence and by this is qualified as determination base to calculate hash values. A repetition of identic structures due to probability law first occurs in **256!** (faculty) = **8E+506** cases. But the structure of CypherMatrix as "**Hash Value"** (term) seems to be still very unmanageable. Therefore content of the CypherMatrix has to be transformed to a concise term.

4.4 Calculating Hash Value H(r)

Best solution is subjecting the total content of CypherMatrix to the dynamic **hash function.** All elements (e_i) of the CypherMatrix (256 ASCII-characters) fill a **CypherString(CM)** in series with length **n** = **256** which is inserted to the hash function (section B) once more.

$$\textbf{CypherString(CM)} = e_1 + e_2 + e_3 + \ldots + e_i + \ldots + e_{256}$$

In our case the following CypherString is subjected to the hash function:

nÕ¾`□‹Ûwa(4Æ§##ë#'"ñ†3#XvQ##˜zÒ^ùøôyèÙå□Œëue#ÚK#g#Uþ#¸©ÐäB#tNoM
#ÜŠ| %º\I#æ¹#‰□üsJL&=Ä÷ìàó¡ð£±€2@~¶#DR□#ûÇÂç)1é<#k.œG;#W¬#¼'ßS5Á
*¡Í6—¤r-®?ÏÊxP#0²ÞH{òbmÝj,F×C,Î>•$îÀ·/âÅ##pEŽ}š:È½ˆõŸªËT9žØµi+h¯
8Z›É«ï□#…ýÓã´Ã°ƒqÌ»³ÖY‡¿áÔ! #¥d#7Ñú–"¢□□¨#'[A"í„c#™V_]#êOl

The hash constant **C**(**k**) is calculated as follows:

$$\textbf{C(k)} = \textbf{n} * (\textbf{n} - 2) + \textbf{code} \qquad \text{code} = 1$$
$$\textbf{C(k)} = 256*254 + 1 = 65025$$

In order to avoid collisions the new CypherString is **position weighted**.
The cypherString (CM) of the last CypherMatrix will be subjected to **position weighted** and **expansion** as well (step 1 and step 2 of section B):

$$\textbf{CM(k)} = \sum_{i=1}^{256} (e_i + 1) * (p_i + C_k)$$

$$\textbf{CM(p)} = \sum_{i=1}^{256} (e_i + 1) * p_i * \textbf{CM(k)} + p_i + \textbf{code}$$

$$\textbf{H(r)} = \textbf{CM(p)}$$

For better handling the hash value is shown in digits of number system on bas 62 which comprises the folowing numbers:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
(defined by the author, not standardized)

The final hash value $\mathbf{H(r_1)}$ of first start sequence results as follows:

		decimal	base 62
CM(k)	=	**2143377944**	**2L3OW0**
$H(r_1)$	=	**9179333265149120**	**g2ZJCcxeq**

In further hash calculations **variant B** (displaced changing of indexes: i,j) will be used. Some more alternative solutions for hashing digital series are possible as well.

5 Alternative Hash calculations

If digital character series are longer than chosen input block (e.g. 64 bytes), for instance: longer sequences, text files (messages), drawings, digital images, digital music, programs, measuring results and further digital stored informations, then the hash function procedure has to pass more than one cycle. Basicly two techniques are possible: ***serial mode*** and ***final mode***. The alternatives lead to following variantes:

A. Program in „**serial mode**“
   1. **without** permutation of BASIC VARIATION
   2. **threefold** permutation

B. Program in „**final mode**“
   1. on base of **last** CypherMatrix (version: ***last cycle LC***)
      a) **withput** permutation
      b) **threefold** permutation
   2. on base of **all** cycles (version: ***all cycles AC***)
      a) **without** permutation
      b) **threefold** permutation

The variant cases are tabulated in following overview:

Programm	Basis	mode serial	mode final	Permutation ja	Permutation nein	Hashwert
CMH-1 fmx	FM					2Yzp50P
CMH-2 fmx	FM					1VOP3iSmaq
CMH-3 sm	SM					T9LFZ3QPcoi
CMH-3 smx	SM			B		TD5tFSnNydk
CMH-3 fm	FM					UWQOGEc70L0
CMH-3 fmx	FM			B		TQj5XYvrQBQ
CMH-3 lc	LC					UVu8dtuLu0S
CMH-3 lcx	LC			B		TPn0JL9X5Ku
CMH-3 ac	AC					VZrygbm0GHg
CMH-3 acx	AC			B		UVZHaSs1nui

meanings: CMH = CypherMatrix Hash SM = „serial mode“
FM = „final mode“ LC = „last cycle“
AC = „all cycles“ **x = 3-fach Permutation

5.1 Hash calculation in „serial mode“

In „**serial mode“** the program adds all single hash values **H(r)** (rounds: **r → 1...m)** to a resulting hash value **H(s).** This term can be established either by **threefold** permutation of BASIC VARIATION or **without** permutation.

$$H(s) = \sum_{r=1}^{m} H(r)$$

The file **Cannery.txt** has 7 digital input sequences (**m = 7).** Hence, the serial hash value in **serial mode H(s)** comprises the sum of all 7 current single cycle hash values. Dependent on using **without** or **with threefold permutation** there are different courses and results.

Program **SEhashLT** calculates in serial mode without permutation the following hash value for the file „Cannery.txt“:

CMH-3 sm / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery.txt

decimal	base 62	cycle	CM(k)
9177625875610505	g25FViSPR	1	2L3OJB
9047399075016964	fR6X7yZng	2	2L3H4g
9004700087069993	fEynGqbUH	3	2L3KRX
9248631284063376	gMFL5oAMq	4	2L3wWy
9017297474745232	fIYZspUPY	5	2L3d3E
8930014319381028	etluPabae	6	2L3b1G
8252507714858112	bnO1cchsm	7	2L2PTs

Serial Hash value for: CANNERY.TXT
decimal: 62678175830745210
base 16: DEAD78058FA07A
base 62: **4d48QvObxC** **H(s)**

In „serial mode“ with threefold permutation the definite hash value **H(s)** results as follows:

CMH-3 smx / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery.txt

decimal	base 62	cycle	CM(k)
9179333265149120	g2ZJCcxeq	1	2L3OW0
8875591246015012	eeJlBt11A	2	2L2wGI
8876597508913628	eebTZVad2	3	2Jm2hN
8633026986588816	dXRHRUBEW	4	2L2jxU
9149020339451776	ftxdHrvO4	5	2L3t0S
9292917235850244	gYp1BwgSK	6	2L4IyY
9215450950314884	gCpBGzAG4	7	2L4I8M

Serial Hash value for: CANNERY.TXT
decimal: 63221937532283480
base 16: E09C04689FAA58
base 62: **4fYXfAJWe0** **H(s)**

5.2 Hash calculation in „final mode“

To calculate the hash value in „final mode“ the serial hash value **H(s)** is supplemented by a further cycle (round). To achieve this there are three variants to perform the final hash value:

a) a new input sequence (**R)** is compiled out of all serial results (all three notations) and number of cycles to pass the dynamic hash function (section 1,2,3,4) another time,
b) a new input sequence (**R**) is created out of all elements of the last CypherMatrix and subjected to the dynamic hash function (section 1,2,3,4)) (version: **last cycle: *LC***) or

c) all serial hash values **H(s)** of single rounds (in **total** or in **parts**) are combined to a CypherString and as an additional round are subjected to the dynamic hash function (section B). The result added to the serial hash value **H(s)** represents the final hash value (version: **all cycles** ***AC***).

5.2.1 Serial results of all cycles (version: final mode)

Sum of the foregoing rounds in all three notations and number of the rounds form a new input string (**R**) to perform an additional pass. Cycles $\mathbf{r_j}$ are not necessary bacause the number of rounds is already included in the input string.

Input string **(R)** = *Base 62 + decimal + base 16 + rounds*
Calculation leads to following hash value:

Serial Hash value for: CANNERY.TXT
decimal: 63221937532283480
base 16: E09C04689FAA58
base 62: 4fYXfAJWe0

Input string **(R)** = *4fYXfAJWe063221937532283480E09C04689FAA587*
Calculation leads to following hash value:

CMH-3fmx-B / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery.txt

decimal	base 62	cycle	CM(k)
9179333265149120	g2ZJCcxeq	1	2L3OW0
8875591246015012	eeJlBt11A	2	2L2wGI
8876597508913628	eebTZVad2	3	2Jm2hN
8633026986588816	dXRHRUBEW	4	2L2jxU
9149020339451776	ftxdHrvO4	5	2L3t0S
9292917235850244	gYp1BwgSK	6	2L4IyY
9215450950314884	gCpBGzAG4	7	2L4I8M

decimal: 63221937532283480
base 16: E09C04689FAA58
base 62: 4fYXfAJWe0 **H(s)**

Matrix value last cycle:
decimal: 325135213628
base 16: 4BB391A43C
base 62: 5itmDkq **H(l)**

Final Hash value for: CANNERY.TXT
decimal: 63222262667497108
base 16: E09C501C314E94
base 62: **4fYdO45kOq** **H**

Integrating of all previous sums and number of rounds acquire each divergence in analysing the digital data.

5.2.2 Hash value of last CypherMatrix (version: last cycle)

Despite of chosen block length (e.g. 64 bytes) all 256 elements ($\mathbf{e_i}$) of last Cypher Matrix are aggregated sequently in a new CypherString which forms an additional input sequence subjecting it to the dynamic hash function (section B) and generating a further matrix value **H(l)**.

$$\mathbf{sequence} = \mathbf{e_1} + \mathbf{e_2} + \mathbf{e_3} + \ldots + \mathbf{e_i} + \ldots + \mathbf{e_{256}}$$

The thus generated additional **Matrix value H(l)** and the sum of all foregoing rounds **H(s)** are summerized. The result forms the **definite Hash Value H**

The following list (program: **CmhashLC**) demonstrates the single rounds of file CANNERY.TXT with an additional pass of all elements from the last CypherMatrix:

```
CMH-3 lcx / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery.txt
-------------------------------------------------------------------------
      decimal              base 62       cycle   CM(k)

9179333265149120     g2ZJCcxeq        1     2L3OW0
8875591246015012     eeJlBt11A        2     2L2wGl
8876597508913628     eebTZVad2        3     2Jm2hN
8633026986588816     dXRHRUBEW        4     2L2jxU
9149020339451776     ftxdHrvO4        5     2L3t0S
9292917235850244     gYp1BwgSK        6     2L4IyY
9215450950314884     gCpBGzAG4        7     2L4I8M
-------------------------------------------------------------------------------
decimal:  63221937532283480
base 16:     E09C04689FAA58
base 62:            4fYXfAJWe0                    H(s)
-------------------------------------------------------------------------
Matrix value last cycle:
decimal:    9214460994866052
base 16:     20BC80E2654B84
base 62:            gCXkgyzHo                     H(l)
-------------------------------------------------------------------------
Final Hash value for: CANNERY.TXT
decimal:  72436398527149532
base 16:    10158854B04F5DC
base 62:            5LI5PrlVvo                    H
=========================================
```

5.2.3 Aggregating all serial hash values (version: all cycles)

The sum of all hash values **H(s)** is supplemented by an additional round. The program takes the respective last three digits (number system on base 62) from each serial hash value **H(r)** and combines them to a new sequence to be inserted. The extracts are limited to three digits because otherwise in case of voluminous files the sequences and thus the hash values as well would become too long. By this all results of the foregoing rounds are included into the final hash calculating. The result of the additional round **H(l)** and the sum **H(s)** of serial hash values are added to form the **final hash value H.**

The program **CMH-3 acx.exe** calculates the following hash values:

CMH-3 acx / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery.txt

decimal	base 62	cycle	CM(k)
9179333265149120	g2ZJCc**xeq**	1	2L3OW0
8875591246015012	eeJlBt**11A**	2	2L2wGl
8876597508913628	eebTZV**ad2**	3	2Jm2hN
8633026986588816	dXRHRU**BEW**	4	2L2jxU
9149020339451776	ftxdHr**vO4**	5	2L3t0S
9292917235850244	gYp1Bw**gSK**	6	2L4IyY
9215450950314884	gCpBGz**AG4**	7	2L4I8M

decimal: 63221937532283480
base 16: E09C04689FAA58
base 62: 4fYXfAJWe0 **H(s)**

The sequence to be inserted to the last processing cycle reads as follows:

Seqence = **xeq11Aad2BEWvO4gSKAG4**

The program calculates this sequence as last cycle to interim matrix value **H(l)** which is added to the sum **H(s)** in order to form the **final hash value H**.

Matrix value last cycle:
decimal: 9214460994866052
base 16: 20BC80E2654B84
base 62: gCXkgyzHo **H(l)**

Final Hash value for: CANNERY.TXT
decimal: 72436398527149532
base 16: 10158854B04F5DC
base 62: **5LI5PrIVvo** **H**

From all here demonstrated examples the program **„CMH-3 fmx“** seems to be best qualified for practical using. Lengths of hash values are from **9** to **12** digits in number system on base 62. Thus the range of the hash values streches from **62^9** to **62^12**, rsp. in decimal numbers:

1.35370865462636E+16 up to **3.2262667623979E+21**

6 Sensitivity analysis

In order to demonstrate the sensitivity of the hash function the last character in the last line of file **Cannary.txt** is changed from ..4**5** to ..4**4**:

John Steinbeck, Cannery Row, New York 1945
. . . . . 01110010 01101011 00100000 00110001 00111001 00110100 00110101
John Steinbeck, Cannery Row, New York 1944
. . . . . 01110010 01101011 00100000 00110001 00111001 00110100 00110100

char		bit
5	=	00110101
4	=	00110100

The last bit „1“ is set to „0“. The entire rest remains unchanged. Calculation of the respective hash value by the program **CMhashLC.exe** leads to following Data:

Primary text file

CMH-3fmx-B / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery.txt
---
decimal: 63221937532283480
base 16: E09C04689FAA58
base 62: 4fYXfAJWe0

Matrix value last cycle:
decimal: 325135213628
base 16: 4BB391A43C
base 62: 5itmDkq

Final Hash value for: CANNERY.TXT
decimal: 63222262667497108
base 16: E09C501C314E94
base 62: **4fYdO45kOq**
==========

Changed last „bit“ (file: Cannery.chd)

CMH-3fmx-B / 64 / 62 / 77 / 1
Serial Matrix Values in Cannery1.txt
---
decimal: 63042067281734244
base 16: DFF86D1A3BEA64
base 62: 4ejSwjanFE

Matrix value last cycle:
decimal: 351794513180
base 16: 51E896451C
base 62: 6BzxvPI

Final Hash value for: CANNERY1.TXT
decimal: 63042419076247424
base 16: DFF8BF02D22F80
base 62: **4ejZ8jYieW**
==========

The change caused by only **one bit** results in a hash value difference as follows:

	original	changed	difference
base 62	**4fYdO45kOq**	**4ejZ8jYieW**	**p4FKX1kK**
decimal	63222262667497108	63042419076247424	179843591249684

7 CypherMatrix compared with actual processes

In researching process of digital facts many hash functions have been developed, which fulfill their task more or less well. As problem remains however that procedures must be safe enough (clear and undisputable). The so far used procedures of the SHA family are lately endangered by special attacks. Thus NIST [#9] at present accomplishes an international competition, in which at the end the best suitable procedure is to be elected.

Those candidates already became known [#10] are essentially arranged according to a pattern, which corresponds to the structure of previous hash functions to a large extent. In order to compare with CypherMatrix hash values a confrontation of the two ranges is shown in the following tabula.

Accordingly – strictly speaking – CypherMatrix is a pure **mathematical** procedure. There are only few connections to current hash algorithms.

Hash Functions (general)	CypherMatrix Hash Function
Fundamental Elements	
bits	**bytes**
Additional Functions	
IVs, SALT, padding	none
Working Steps, Sequences (message digest)	
224, 256, 384, 512, 1024 bits (in part: variable)	continuous, unlimited (optimal: 16 up to 256 bytes)
Internal Consistence (internal states)	
Feistel network keys, S-boxes, constants, shifting, rotation, mixing, XOR swapping, permutations	position weighted hash constant C expansion, contraction (one way functions)
Compression Function	
„Merkle-Damgård“ block ciphers, AES-based Threefish based	none
Output Function	
output function truncated to output fixed length	CypherMatrix: GF(16^2) threefold permutation number system to basis 62 (9 up to 11 digits)
Applications (Anwendungen)	
hashing, MAC, randomizing, PRNG, digital signatures, authentification, encryptions	hash funktion, randomizing digital signatures, RNG, authentification, encryptions

8 Concluding remarks

Detailed explanations and further examples to CypherMatrix procedure are presented in internet at: http://www.telecypher.net/.

In section C. Dynamic Hash Functions shown programs may be downloaded from http://www.telecypher.net/DELIVERY.HTM and tested in detail.

9 References

[#1] Author's patent, DPMA 19811593 from 18.03.1998

[#2] Knudsen, Lars R., Lai, Xuejia and Preneel, Bart, Attacks on Fast Double Block Length Hash Functions, Journal of Cryptology, Vol.11#1, New York, 1998, p.59

[#3] analogous to: Descartes, René, (without more indication) Cartesian coordinate system: object = subject * location * time

[#5] Collisionfree: telecypher.net/Collfree.pdf

[#6] Article at: telecypher.net/CORECYPH.HTM#Z15

[#9] NIST: National Institute of Standards and Technology, USA

[#10] Grøstl, LANE, SHAvite-3, Skein, TIB3, ... siehe: //ehash.iak.tugraz.qt/wiki/The_SHA-3_Zoo

Munich, in January 2012