

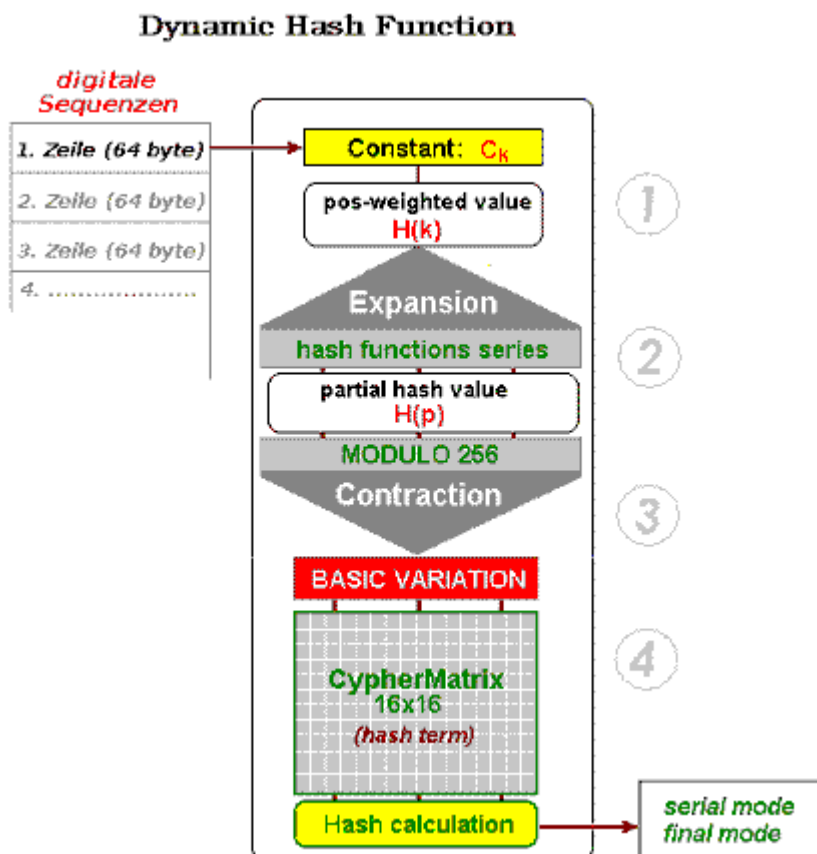
# >CypherMatrix< als dynamische Hash Funktion

(Ernst Erich Schnoor)

Die wesentliche Aufgabe einer Hashfunktion besteht in der Umformung digitaler Zeichen beliebiger Länge ( $n$ ) zu einer eindeutigen digitalen Ausgabe ( $H$ ) in festgelegter Form (term) [#2]. Das kann ein einzelner **Wert**, eine eindeutige **Zeichenfolge** (Vektor) oder eine bestimmte **Struktur** (z.B.: Matrix) sein. Mit der **Basisfunktion** werden dynamische Hashverfahren begründet, die in vielen Bereichen der Kryptographie eingesetzt werden können. Eine ausführliche Darstellung der Basisfunktion lesen Sie im Artikel: [Kryptographische Basisfunktion in Byte-Technik](#). Die folgende Abhandlung berücksichtigt die letzte Weiterentwicklung des Verfahrens.

## A. Der Hashwert Generator

Die **Basisfunktion** bildet den Kern der Hashverfahren [#1]. Digitale Zeichenfolgen (Dateien) werden in Klartextblöcke aufgeteilt (z.B. 64 Bytes), die in jeder Runde positionsgewichtet, mit der Hashkonstante  $C_k$  multipliziert, zu einer Hashfunktionsfolge erweitert, wieder zur BASIS VARIATION verdichtet und abschließend wahlweise einer dreifachen Permutation unterworfen werden. Die entstehende letzte CypherMatrix mit  $16 \times 16$  Elementen ergibt den **Hashwert (H)**. Nach den Gesetzen der Wahrscheinlichkeit entsteht eine Wiederholung identischer Strukturen erst in  $256!$  (Fakultät) =  $8E+506$  Fällen. Das folgende Schema erläutert die Zusammenhänge:

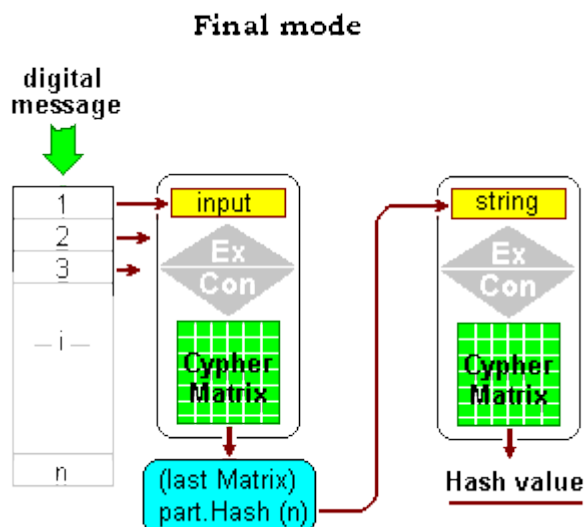
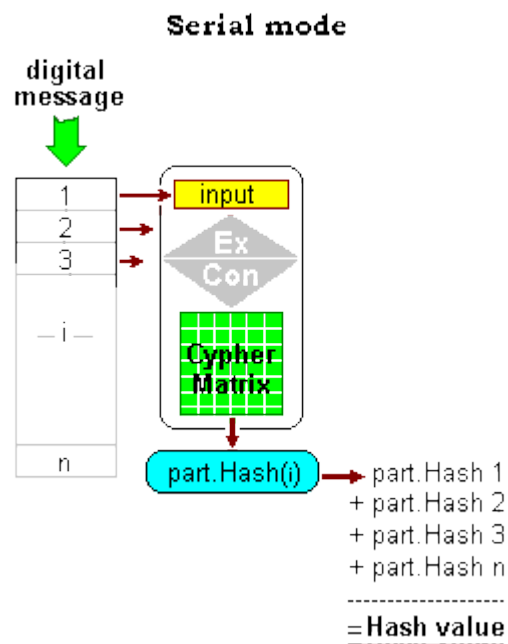


Die dynamische Hashfunktion arbeitet in vier Stufen:

1. Berechnung eines positionsgewichteten Werts **H(k)** der digitalen Eingangssequenz mit Einbindung der Hashkonstanten **C(k)** für kollisionsfreien Verlauf,
2. **Expansion**: Erweiterung der Eingabesequenz zu einer Hashfunktionsfolge von etwa 160 bis 2400 Ziffern im Zahlensystem zur **Basis 77** – erste Einwegfunktion - und Berechnung eines Zwischenwertes **H(p)**.
3. **Kontraktion**: Verdichtung der Hashfunktionsfolge mit Modulo 256 zu einem Array von 16x16 Elementen: **BASIS VARIATION** – zweite Einwegfunktion - und
4. wahlweise dreifache **Permutation** der BASIS VARIATION zur Generierung einer Matrix mit 16x16 Zeichen als endgültigen Hashwert: **CypherMatrix**.

Zwei Techniken sind möglich:

- a) Serielle Berechnung von partiellen Hashwerten der digitalen Sequenzen in jeder Runde und Addition aller Rundenwerte zum Hashwert (**serial mode**) und
- b) Berechnung des Hashwertes aus der letzten Cypher Matrix nach Addition und Berücksichtigung aller vorhergehenden partiellen Hashwerte (**final mode**).



## B. Die Arbeitsstufen zur CypherMatrix

Zur Erläuterung der Arbeitsschritte dient die Textdatei: HESSE.TXT (742 Bytes).

*Als Siddhartha den Hain verließ, in welchem der Buddha, der Vollendete, zurückblieb, in welchem Govinda zurückblieb, da fühlte er, dass in diesem Hain auch sein bisheriges Leben hinter ihm zurückblieb und sich von ihm trennte. Dieser Empfindung, die ihn ganz erfüllte, sann er im langsamen Dahingehen nach. Tief sann er nach, wie durch ein tiefes Wasser ließ er sich bis auf den Boden dieser Empfindung hinab, bis dahin, wo die Ursachen ruhen, denn Ursachen erkennen so schien ihm, das eben ist Denken, und dadurch allein werden Empfindungen zu Erkenntnissen und gehen nicht verloren, sondern werden wesenhaft und beginnen auszustrahlen, was in ihnen ist.*

Hermann Hesse, Siddhartha, Eine indische Dichtung, Montagnola 1953

Die Eingabe einer Schlüssel-Sequenz zur Initialisierung des Verfahrens ist nicht erforderlich. Die erste Zeile des zu hashenden Textes in der gewählten Länge (64 Zeichen) wird als erste Runde ( $\mathbf{r}_1$ ) der Hashfunktion unterworfen. Der Hashwert der Eingabesequenz  $\mathbf{H}(\mathbf{r}_1)$  wird berechnet.

*Als Siddhartha den Hain verließ, in welchem der Buddha, der Voll*

Das Verfahren durchläuft die folgenden Stufen:

### 1. Positionsgewichtung

Verknüpfung der Eingabe durch Addition der Zeichen (Bytes):

$$\mathbf{H}(\mathbf{k}) = \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3 + \dots + \mathbf{a}_i + \dots + \mathbf{a}_n$$

(der Wert für  $\mathbf{a}(i)$  erhöht sich um (+1), da sonst ASCII-null nicht berücksichtigt wird)

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

$$\mathbf{H}(\mathbf{k}) = 5930$$

Der für  $\mathbf{H}(\mathbf{k})$  errechnete Wert ist jedoch weit davon entfernt, als Hashwert zu dienen. Es müssen noch weitere Merkmale hinzu kommen, insbesondere **Positionsgewichtung** und **Kollisionsfreiheit**, um eindeutige Ergebnisse zu erzielen.

Jedes Zeichen der Sequenz  $\mathbf{a}(i)$  wird **positionsgewichtet**, indem sein Wert mit seiner Position  $\mathbf{p}(i)$  multipliziert wird [#3]. Die Zeit ist nicht relevant.

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

Kollisionen werden durch Einbindung der **Hashkonstante C(k)** verhindert.  
 Die Konstante **C(k)** bestimmt sich allein aus der Länge (**n**) der Eingabesequenz und einem individuellen Anwender Code (1 bis 99).

$$\begin{aligned} C(k) &= n * (n - 2) + \text{code} & \text{code} &= 1 \\ C(k) &= 64 * 62 + 1 = 3969 \end{aligned}$$

Die Ableitung der Hashkonstanten **C(k)** wird im Artikel ["Bestimmungsfaktoren für Kollisionsfreiheit"](#) dargelegt [#5].

Unter Einbindung der „Hashkonstanten“ **C(k)** wird der Zwischenwert **H(k)** wie folgt errechnet:

$$\begin{aligned} H(k) &= \sum_{i=1}^n (a_i + 1) * (p_i + C_k) \\ H_k &= 23727493 \end{aligned}$$

## 2. Expansion

Zur Erhöhung der Sicherheit wird die **Hashfunktionsfolge H(p)** eingeführt, die die Eingangssequenz zu einer umfangreichen Folge in einem höherwertigen Zahlensystem expandiert. Das Zahlensystem der Expansion ist wählbar zwischen 64 ... 77 ... 96. Hier wird **Basis 77** festgelegt. Für jedes Zeichen der Eingabe-Sequenz errechnet das Verfahren den dezimalen Wert (**s<sub>i</sub>**), der dann zu (**d<sub>i</sub>**) (Ziffern im Zahlensystem zur Basis 77) umgewandelt wird. Gleichzeitig ermittelt das Verfahren die Summe aller einzelnen Ergebnisse (**s<sub>i</sub>**) als zusätzlichen Wert **H(p)** zur Bestimmung verschiedener Steuerungsparameter.

$$\begin{aligned} s_i &= (a_i + 1) * p_i * H_k + p_i + \text{code} + \text{cycle} \\ s_i &\rightarrow d_i \text{ (base 77)} \end{aligned}$$

$$\begin{aligned} H_p &= d_1 + d_2 + d_3 + \dots + d_i + \dots + d_m \\ (m &= \text{Anzahl der Zahlen im System zur Basis 77}) \end{aligned}$$

$$\begin{aligned} H_p &= \sum_{i=1}^n S_i \\ H_p &= 4539615145447 \end{aligned}$$

Als Bestimmungsdaten ergeben sich die folgenden Werte:

	dezimal	Basis 62
Hashkonstante C(k):	<b>3969</b>	<b>121</b>
positionsgewichteter Wert (H <sub>k</sub> ):	<b>23727493</b>	<b>1bYbV</b>
partieller Zwischenwert (H <sub>p</sub> ):	<b>4539615145447</b>	<b>1HvByW87</b>
gesamter Hashwert (H <sub>p</sub> +H <sub>k</sub> ):	<b>4539638872940</b>	<b>1HvDa4jc</b>

Von den ermittelten Werten für Steuerungsparameter sind nur die Werte für **Variante**, **Alpha** und **Theta** erforderlich. Die weiteren Parameter werden nicht benötigt.

<b>Variante</b>	$(H_k \text{ MOD } 11) + 1$	=	<b>10</b>	Beginn Rückumwandlung
<b>Alpha</b>	$((H_k + H_p) \text{ MOD } 255) + 1$	=	<b>21</b>	Offset BASIS VARIATION
<b>Theta</b>	$(H_k \text{ MOD } 32) + 1$	=	<b>6</b>	Variation Rückumwandlung

Bei der Generierung der Hashfunktionsfolge ergibt sich beispielsweise für das Zeichen “**S**” an der Position **5** der Eingabesequenz folgende Berechnung:

$$s_5 = (83+1) * 5 * 23727493 + 5 + 1 + 1 = 9965547067$$

$$d_5 = \mathbf{3qbwBE} \text{ (base 77)}$$

Die **Hashfunktionsfolge H(p)** umfasst 383 Ziffern im Zahlensystem zur Basis 77:

igHçE1âBCiq**33ã**qiZ1C7ZâS**3qbwBE**5iLéàN6FG5aw76TSK08Lâr#k8jawpsB6âA9vCNpVh  
 áBéQYévC25Kfi4Q8vezECwudêFFViJnHdkd0H5cGGVACzZâ9dl37ãNàKY3hY2MTGn@Y6çi  
 zJ4Q64Qá6NJ2ëvFRGzääâQw2M99QçârWàQ#X5mHzVãWsDCléYRa9g3@éOVjldAiY4Lê@iA  
 VâEâ5cää9pwXêGHULbKPENrZ4ääwdbuxuêvbgkNLNfZnR#CCu59lvd@xOGAf9ëTCPITJMK  
 fDâCjc6Sxè7âUptS2zBjB&6âlk2êTqámyJ0WDkTêOHLRrhunkGFRelwoZ@ák7páCk&Vxat  
 QOAHRYë97ke7âNKyâ36eayF5BKbzBnHg&

### 3. Kontraktion

Als nächste Stufe wird die Hashfunktionsfolge H(p) mit **Modulo 256** zum Array **BASIS VARIATION** in 16x16 dezimale Ziffern verdichtet. Dabei wird für die Hashfunktionsfolge das Zahlensystem zur Basis **78** unterstellt (Expansion 77 + 1).

Die ersten drei Rückrechnungen ab **Variante = 10** zeigen sich wie folgt:

3 Ziffern Basis 78	dezimal	Modulo 256 - Theta			Element
<b>iq3</b>	271755	139	6	133	<b>133</b>
<b>q33</b>	316605	189	6	183	<b>183</b>
<b>33ã</b>	18554	122	6	116	<b>116</b>

Die **BASIS VARIATION** ergibt die folgende Struktur:

#### **BASIS VARIATION (256 Elemente)**

**133 183 116** 050 014 085 086 032 109 111 067 076 209 136 068 074  
 005 202 175 100 075 090 219 135 198 121 052 091 228 000 061 138  
 071 132 203 130 082 127 193 017 035 238 066 207 244 160 105 034  
 134 178 123 106 211 153 072 093 095 063 163 018 013 038 012 176  
 137 199 015 073 204 028 016 245 205 252 254 139 161 011 062 255  
 148 019 154 007 191 008 039 020 049 165 140 212 021 001 022 231  
 045 170 114 002 057 182 206 141 227 158 036 119 210 087 179 173  
 115 023 120 248 169 024 131 069 180 083 184 064 104 033 092 030

```

174 192 084 124 216 253 186 229 056 004 110 167 098 112 230 151
077 194 195 025 026 147 070 108 144 027 065 155 185 029 031 187
166 125 078 233 053 079 246 128 037 181 101 142 003 251 088 197
200 164 040 096 168 117 177 080 041 159 042 249 156 232 226 043
044 010 081 188 046 047 113 213 089 048 208 051 242 171 172 162
196 189 214 190 054 094 143 055 126 107 129 247 122 234 223 060
097 099 058 201 118 102 157 059 145 237 235 103 146 149 150 152
215 217 218 220 250 236 239 221 222 224 225 240 241 243 006 009

```

Die Zahlen der **BASIS VARIATION** (16x16) bilden die Grundlage für die folgenden Berechnungen.

#### 4. Generierung der CypherMatrix (CM)

Die dezimalen Zahlen der BASIS VARIATION (16x16) können direkt auf die 256 Werte des erweiterten ASCII-Zeichensatzes bezogen werden und ergeben so die erste **CypherMatrix** mit 16x16 Elementen (**CM1**). Zur Erhöhung der Sicherheit werden die Elemente der ersten CypherMatrix einer **dreifachen Permutation** unterworfen und damit zur dritten **CypherMatrix** (**CM3**) erweitert.

In Pseudo-Code Darstellung geschieht folgendes:

```

k = Alpha                                     Initialisierung
FOR i = 1 TO 16
  FOR j = 1 TO 16
    Matrix$(1,i,j) = CHR$(Variation(k))      BASIS-VARIATION
    INCR k                                     256 Elemente
    IF k > 256 THEN k = 1
  NEXT j
NEXT i

CM1Set$ = ""                                 Elemente der ersten CypherMatrix (CM1)
CM3Set$ = ""                                 Elemente der dritten CypherMatrix (CM3)

FOR s = 1 TO 3                                drei Schleifen
  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
          Matrix$(2,a,j) = Matrix$(1,i,j)
          CM1Set$ = CM1Set$ + Matrix$(1,i,j)  CypherString
        CASE 2
          Matrix$(3,i,a) = Matrix$(2,i,j)
        CASE 3
          CM3Set$ = CM3Set$ + Matrix$(3,i,j)  CypherString
      END SELECT
    NEXT j
  NEXT i
NEXT s

```

## Ableitung der ersten CypherMatrix (CM1) aus der BASIS VARIATION

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

Aus dieser ersten CypherMatrix (CM1) mit 16x16 Elementen wird durch Permutation die dritte CypherMatrix (CM3) gebildet.

## Generierung der dritten CypherMatrix (CM3) durch dreifache Permutation

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

Die **Struktur** der Cypher Matrix stellt eine eindeutige Abbildung der Eingangssequenz dar. Nach den Gesetzen der Wahrscheinlichkeit tritt eine Wiederholung der Struktur erst in **256!**(Fakultät) = **8E+506** Fällen ein. Die Struktur der CypherMatrix als **Hashwert** (Term) ist in der vorliegenden Form allerdings recht unhandlich. Sie muss daher in prägnante Größen umgeformt werden.

## 5. Endstufe: Hashwert

Die Umformung geschieht am besten in der Form, dass der Inhalt der Cypher Matrix in seiner Gesamtheit der dynamischen **Hash Funktion** unterworfen wird. Die Elemente ( $e_i$ )

der CypherMatrix (256 ASCII-Zeichen) werden seriell in einem **CypherString** mit der Länge **n = 256** geordnet und der Funktion als weitere Eingabesequenz übergeben.

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

Die Hashkonstante **C(k)** wird wie folgt errechnet

$$\begin{aligned} C(k) &= n * (n - 2) + \text{code} & \text{code} &= 1 \\ C(k) &= 256 * 254 + 1 = 65025 \end{aligned}$$

Zur Vermeidung von Kollisionen wird der CypherString zusätzlich **positionsgewichtet**. Außerdem muss die laufende Runde (**r<sub>i</sub>**) berücksichtigt werden. Zur Bestimmung des Hashwertes **H(r)** bestehen zwei unterschiedliche Verfahren (**Endstufe a** und **Endstufe b**):

#### a) Verfahren mit „Endstufe a“

Dem Ergebnis der Positionsgewichtung des **CypherStrings(CM)** (nur Stufe 1 des Abschnitts B) wird der partielle Wert **H(p)** der vorhergehenden Eingangssequenz hinzu gezählt

$$\begin{aligned} H(\text{CM}) &= \sum_{i=1}^{256} (e_i + 1) * (p_i + C_k + r_j) \\ H(r) &= H(\text{CM}) + H(p) \end{aligned}$$

Als Ergebnis der Berechnung mit der Endstufe a) ergibt sich folgender Hashwert **H(r)**:

	dezimal	Basis 62
<b>H(CM3)</b> =	2143289981	2L31dF
<b>H(p)</b> =	4539615145447	1HvByW87
-----		
<b>H(r)</b> =	4541758435428	1HxX1XIM
=====		

#### b) Verfahren mit “Endstufe b”

Der **CypherString(CM)** der letzten CypherMatrix wird sowohl **positionsgewichtet** als auch der **Expansion** unterworfen (Stufen 1 und 2 des Abschnitts B):

$$\begin{aligned} H(\text{CM}) &= \sum_{i=1}^{256} (e_i + 1) * (p_i + C_k j) \\ H(\text{CM3}) &= \sum_{i=1}^{256} (e_i + 1) * p_i * H(\text{CM}) + p_i + \text{code} + r_j \\ H(r) &= H(\text{CM3}) \end{aligned}$$

Die Berechnung mit der Endstufe b) ergibt als Ergebnis den endgültigen Hashwert **H(r)**:

	dezimal	Basis 62
<b>H(CM)</b>	<b>2143289981</b>	<b>2L31dF</b>
<b>H(r)</b>	<b>9060932001199369</b>	<b>fUwmvFhJh</b>

---

Für die weiteren Hashwertberechnungen wird das Verfahren mit der **Endstufe b)** angewendet. Anstelle des partiellen Wertes **H(p)** der vorhergehenden Eingangssequenz wird der neue Expansionswert **H(CM)** des CypherStrings(CM) in die Berechnung einbezogen. Damit entsteht eine größere Sicherheit für die errechneten Werte. Für Hashwertberechnungen digitaler Folgen gibt es weitere Lösungsmöglichkeiten.

### C. Alternative Hashwerte

Sind die digitalen Zeichenfolgen länger als der gewählte Eingangsblock (z.B. 64 Bytes) wie zum Beispiel: eine längere Eingangssequenz, Texte (Nachrichten), Zeichnungen, digitale Bilder, digitale Musik, Programme, Messergebnisse und weitere digital gespeicherte Informationen, dann muss die Hashfunktion mehr als eine Runde durchlaufen. Dabei sind grundsätzlich zwei Techniken möglich: *serial mode* und *final mode*, die zu folgenden Gestaltungen führen:

- A. Berechnung im „**serial mode**“
  - 1. **ohne** Permutation der BASIS VARIATION
  - 2. **dreifache** Permutation
- B. Berechnung im „**final mode**“
  - 1. auf Basis der **letzten** CypherMatrix (Version: *last cycle LC*)
    - a) **ohne** Permutation
    - b) **dreifache** Permutation
  - 2. auf Basis **aller** Runden (Version: *all cycles AC*)
    - a) **ohne** Permutation
    - b) **dreifache** Permutation

Die einzelnen Fälle sind in der folgenden Übersicht zusammengestellt:

**Dynamische Hashfunktionen**

Programm	Basis	mode		Permutation		Hashwert
		serial	final	ja	nein	
CMhashAC.exe	AC					VGH3II5eCoC
AChashLT.exe	AC					VSXQJ1EMIoC
CMhashLC.exe	LC					V5IJUJFLLcG
LChashLT.exe	LC					VFpXbQFN0ee
CMhashSE.exe						UFTCGGCX7z6
SEhashLT.exe						UWiAaNQTzfc

Es bedeuten: CM = CypherMatrix SE = „serial mode“  
 LC = „last cycle“ AC = „all cycles“  
 LT = „light“ (ohne Permutationen)

## 1. Hashwerte im „serial mode“

Im „serial mode“ addiert das Verfahren alle Hashwerte  $\mathbf{H}(r)$  der einzelnen Runden (wobei:  $r \rightarrow 1 \dots m$ ) zum seriellen Hashwert  $\mathbf{H}(s)$ . Dabei kann der Rundenwert entweder mit **dreifacher** Permutation der BASIS VARIATION oder **ohne** Permutation ermittelt werden.

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

Die Textdatei **Hesse.txt** hat 12 digitale Eingabesequenzen (Runden:  $m = 12$ ). Der serielle Hashwert im „serial mode“  $\mathbf{H}(s)$  umfasst somit die Summe aller 12 vorhergehenden Rundenwerte. Abhängig von Anwendung der **Endstufe a** oder der **Endstufe b** ergeben sich unterschiedliche Verläufe und Hashwerte.

### a) Verfahren mit „Endstufe a“

Für die Datei **Hesse.txt** wird mit der Endstufe a) folgender Hashwert errechnet:

**SMhashSE** / 64 / 62 / 77 / 1  
 Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde	Matrixwert
4541758435428	1HxX1XIM	1	2L31dF
4459228180103	1GVRj0Ex	2	2L2sQr
4487270822186	1H03X4Ti	3	2L44xV
3796561251149	14q7E6LV	4	2L32Pa
4295590162031	1DcpOYUp	5	2L3Wwh
4353281000363	1EdnfQxH	6	2L38cJ
4518558715913	1HYCxsuv	7	2L3A40
3981759111337	186GdRub	8	2L3M49
4965657479359	1PQEijT5	9	2L2YMc
5071811921350	1RI6nx8I	10	2L2ruY
3970626778204	17u7FJWO	11	2L366W
194678614005	3QV1cjF	12	2L3DH6

Serieller Hashwert für HESSE.TXT

dezimal: 48636782471428

Basis 16: 2C3C22246904

Basis 62: **DoHDyfy8** **H(s)**

=====

### b) Verfahren mit „Endstufe b“

Mit der Endstufe b) errechnet das Programm „**SEhashLT**“ im „serial mode“ ohne Permutation für die Datei **Hesse.txt** den folgenden Hashwert:

**SEhashLT** / 64 / 62 / 77 / 1  
 Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde
9419200118399817	h8glbyNPI	1
9123597415131185	fmk31kKCv	2
9449674032440145	hHKoFk5th	3
9219245987514180	gDtj4V64	4
9323223236784353	ghQZY47TV	5
9045405501500289	fQXR3LGWP	6
9283380451364544	gW77MesKG	7
8330384976512225	c9V68i8bh	8
8888735440412292	ei3AfRBLA	9
9131734594658212	fp3J7gMQ0	10
9106921548394384	fi0SZWvzs	11
9267556069916288	gRcWLMMyrQ	12

dezimal: 109589059373027914  
 Basis 16: 18556AB3B32D24A  
 Basis 62: **85uyqRLAAk** **H(s)**  
 =====

Mit der **dreifachen** Permutation in „serial mode“ ergibt sich der folgende Hashwert **H (s)**:

**CMhashSE** / 64 / 62 / 77 / 1  
 Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde	Matrixwert
9060932001199369	fUwmvFhJh	1	2L31dF
8984988440749553	f9NI7tSML	2	2L2sQr
9600271080544529	hy69fmbCb	3	2L44xV
9067368120315460	fWm6EbbVs	4	2L32Pa
9319443179075233	ggM1RjCJI	5	2L3Wwh
9118593568662849	flJx6YUAj	6	2L38cJ
9130536243375808	foiD4KVHc	7	2L3A40
9229621380449441	gGqevptuz	8	2L3M49
8819333900678532	eOLJmPAU0	9	2L2YMc
8980687108755108	f8A21ukxl	10	2L2ruY
9097850271140752	ffQkro22C	11	2L366W
9157046152699520	fwEvp0Oy8	12	2L3DH6

dezimal: 109566671447646154  
 Basis 16: 185424EA2DBDFCA  
 Basis 62: **85och17oXC** **H(s)**  
 =====

## 2. Hashwerte im „final mode“

Zur Berechnung von Hashwerten im „**final mode**“ wird den Ergebnissen der seriellen Hashwert **H(s)** ein abschließender (finaler) Durchlauf hinzugefügt. Dabei kann der endgültige Hashwert **H(f)** in zwei Varianten berechnet werden:

- a) alle **Elemente** der letzten CypherMatrix bilden als **CypherString (CM)** eine neue

Eingabesequenz und werden als abschließende Runde der Hashfunktion (Abschnitt **B**) unterworfen (**Version: letzte Runde**) oder

b) die seriellen **Hashwerte H(s)** der einzelnen Runden (**ganz oder teilweise**) werden zusammengefasst und als abschließende Runde der Hashfunktion (Abschnitt **B**) unterworfen (**Version: alle Runden**).

**a) Hashwert der letzten CypherMatrix (Version: letzte Runde)**

Ungeachtet der gewählten Blocklänge (z.B. 64 Bytes) wird aus allen 256 Elementen (**e<sub>i</sub>**) der letzten laufenden CypherMatrix eine neue Eingangssequenz zur Berechnung eines zusätzlichen **Matrixwert H(I)** generiert.

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

Der **CypherString** von 256 Bytes (in Textzeichen) wird der dynamischen Hashfunktion (Abschnitt **B**) unterworfen. Der auf diese Weise entstehende letzte **Matrixwert H(I)** schließt die Ergebnisse aller vorhergehenden Runden ein. Als letzter Schritt werden die Summe aller Rundenwerte **H(s)** und der zusätzliche Matrixwert **H(I)** addiert, die zusammen den **finalen Hashwert H** ergeben.

Die folgende Aufstellung (Programm: **CMhashLC.exe**) zeigt die Reihenfolge der einzelnen Vorgänge für die Datei HESSE.TXT:

**CMhashLC / 64 / 62 / 77 / 1**

Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde
9060932001199369	fUwmvFhJh	1
8984988440749553	f9NI7tSML	2
9600271080544529	hy69fmbCb	3
9067368120315460	fWm6EbbVs	4
9319443179075233	ggM1RjCJl	5
9118593568662849	flJx6YUAj	6
9130536243375808	foiD4KVHc	7
9229621380449441	gGqevptuz	8
8819333900678532	eOLJmPAU0	9
8980687108755108	f8A21ukxl	10
9097850271140752	ffQkro22C	11
9157046152699520	fwEvp0Oy8	12

dezimal: 109566671447646154

Basis 16: 185424EA2DBDFCA

Basis 62: 85och17oXC

Matrixwert letzte Runde:

dezimal: 9195241660778193

Basis 16: 20AB0608D086D1

Basis 62: g75NvdnSD

**Finaler Hashwert** für: HESSE.TXT

dezimal: 118761913108424347

Basis 16: 1A5ED54ABAC669B

Basis 62: **8lvi4wlbzP**

=====

## b) Zusammenfassung aller Hashwerte (Version: alle Runden)

Die Summe aller Runden Hashwerte **H(s)** wird um eine zusätzliche Runde ergänzt. Dazu entnimmt das Programm von allen seriellen Hashwerten **H(r<sub>i</sub>)** die jeweils letzten drei Ziffern (Zahlen im System zur Basis 62) und verbindet sie zu einer neuen Eingabesequenz. Die Entnahmen werden auf drei Ziffern begrenzt, da sonst bei umfangreichen Dateien die Eingabesequenz und demzufolge auch die Hashwerte zu lang werden. Das Ergebnis des erneuten Durchgangs Rundenwert **H(l)** wird der Summe der seriellen Hashwerte **H(s)** hinzugerechnet.

Für die Datei **Hesse.txt** errechnet das Programm **CMhashAC.exe** folgende Hashwerte:

**CMhashAC / 64 / 62 / 77 / 1**

Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde
9060932001199369	fUwmvF <b>hJh</b>	1
8984988440749553	f9NI7t <b>SML</b>	2
9600271080544529	hy69fm <b>bCb</b>	3
9067368120315460	fWm6E <b>bbVs</b>	4
9319443179075233	ggM1Rj <b>CJl</b>	5
9118593568662849	flJx6Y <b>UAj</b>	6
9130536243375808	foiD4K <b>VHc</b>	7
9229621380449441	gGqevpt <b>uz</b>	8
8819333900678532	eOLJmP <b>AU0</b>	9
8980687108755108	f8A21u <b>kxl</b>	10
9097850271140752	ffQkro <b>22C</b>	11
9157046152699520	fwEvp0 <b>Oy8</b>	12

dezimal: 109566671447646154

Basis 16: 185424EA2DBDFCA

Basis 62: **85och17oXC** **H(s)**

Die Eingabesequenz für den letzten Durchgang ergibt sich wie folgt:

Hashsequenz = **85och17oXChJhSMLbCbCJiUAjVHctuzAU0kxl22COy8**

Mit dieser Sequenz errechnet das Programm im letzten Durchgang folgenden Matrixwert **H(l)** und anschließend aus der Summe mit **H(s)** den **finalen Hashwert H**

Matrixwert letzte Runde:

dezimal: 8956464080891108

Basis 16: 1FD1DB4BFC2CE4

Basis 62: **f1HZViPlq** **H(l)**

=====

**Finaler Hashwert** für: HESSE.TXT  
 dezimal: 118523135528537262  
 Basis 16: 1A51429EED80CAE  
 Basis 62: **8kpuGWqDq2** **H**  
 =====

Von den vorstehend erläuterten Verfahren erscheint das Programm „**CmhashLC**“ für den praktischen Einsatz am besten geeignet zu sein. Die Länge der Hashwerte bewegt sich zwischen **9** bis **12** Ziffern im Zahlensystem zur Basis 62. Die Spanne der Hashwerte reicht somit von **62<sup>9</sup>** bis **62<sup>12</sup>**, bzw. in dezimalen Werten:

**1.35370865462636E+16** bis **3.2262667623979E+21**

#### D. Sensibilität der Hashfunktion

Zur Darstellung der Sensibilität der Hashfunktionen wird in der Datei **Hesse.txt** die letzte Ziffer der letzten Zeile von **..53** auf **..52** geändert.

```

    Hermann Hesse, Siddhartha, Eine indische Dichtung, Montagnola 1953    (Hesse.txt)
    .... 01100001 00100000 00110001 00111001 00110101 00110011
    Hermann Hesse, Siddhartha, Eine indische Dichtung, Montagnola 1952    (Hesse1.txt)
    .... 01100001 00100000 00110001 00111001 00110101 00110010
           char      bit
           3      =   11
           2      =   10
  
```

Das letzte Bit „**1**“ wird auf „**0**“ gesetzt. Im Übrigen bleibt alles unverändert.

Die Berechnung des jeweiligen Hashwerts mit dem Programm **CmhashLC.exe** führt zu folgenden Ergebnissen:

#### Unveränderte Textdatei

```

CMhashLC / 64 / 62 / 77 / 1
Serielle Matrixwerte in Hesse.txt
-----
dezimal: 109566671447646154
Basis 16: 185424EA2DBDFCA
Basis 62:      85och17oXC
-----
  
```

```

Matrixwert letzte Runde:
dezimal: 9195241660778193
Basis 16: 20AB0608D086D1
Basis 62:      g75NvdnSD
-----
  
```

```

Finaler Hashwert für: HESSE.TXT
dezimal: 118761913108424347
Basis 16: 1A5ED54ABAC669B
Basis 62:      8lvi4wlbzP
=====
  
```

## In einem Bit veränderte Textdatei

CMhashLC / 64 / 62 / 77 / 1  
Serielle Matrixwerte in **Hesse1.txt**

-----  
dezimal: 109508360357658314  
Basis 16: 1850D46070DBECA  
Basis 62: 85Y45qshsQ  
-----

Matrixwerte letzte Runde:  
dezimal: 9483331550850177  
Basis 16: 21B10A30602C81  
Basis 62: hQtMwPJ1V  
-----

Finaler **Hashwert** für: **HESSE1.TXT**  
dezimal: 118991691908508491  
Basis 16: 1A6BE50376DEB4B  
Basis 62: **8myxSnI0tv**  
=====

Der durch Änderung auch nur eines Bits verursachte Unterschied in den Hashwerten beträgt:

Basis 62: **13FNqzOuW**  
dezimal: **229778806995656**

## E. Vergleich der CypherMatrix Hashfunktion mit aktuellen Verfahren

Im Verlauf der Forschung digitaler Sachverhalte wurden viele Hashfunktionen entwickelt, die ihre Aufgabe auch mehr oder weniger gut erfüllen. Als Problem bleibt jedoch, dass die Verfahren auch sicher genug sein müssen (eindeutig und unangreifbar). Die bisher verwendeten Verfahren der SHA-Familie sind in letzter Zeit durch spezielle Angriffe gefährdet. NIST [#9] führt daher zur Zeit einen internationalen Wettbewerb durch, in dem am Ende das am besten geeignete Verfahren gekürt werden soll.

Die bereits bekannt gewordenen Kandidaten [#10] sind im Wesentlichen nach einem Schema gestaltet, das weitgehend dem Aufbau bisheriger Hashfunktionen entspricht. Um mit den **CypherMtrix Hashwerten** zu vergleichen, soll im Folgenden eine Gegenüberstellung der beiden Bereiche vorgenommen werden.

Der grundsätzliche Unterschied zeigt sich bereits in den Grundelementen: Sowohl die aktuellen Verfahren als auch die Kandidaten im NIST Wettbewerb verwenden **“Bits”** als kleinste Arbeitseinheit, während CypherMatrix nur **“Bytes” (8 Bits)** als Zahlen verarbeitet (Zahlensysteme von Basis 2 bis zur Basis 256). Dementsprechend ist CypherMatrix im Grunde genommen ein rein **mathematisches Verfahren**.

<b>Hashfunktionen</b> (allgem.)	<b>CypherMatrix Hashfunktion</b>
<b>Grundelemente</b>	
<b>Bits</b>	<b>Bytes</b>
<b>Zusatzfunktionen</b>	
IVs, SALT, padding	keine
<b>Arbeitsschritte, Sequenzen</b> (message digest)	
224, 256, 384, 512, 1024 bits (teilweise: variabel)	kontinuierlich, unbegrenzt (optimal: 16 bis 256 Bytes)
<b>Interne Beschaffenheit</b> (internal states)	
Feistel network keys, S-boxes, constants, shifting, rotation, mixing, XOR swapping, permutations	positionsgewichtet Hash-Konstante C Expansion, Kontraktion (Einwegfunktionen)
<b>Kompressionsfunktion</b>	
„Merkle-Damgård“ block ciphers, AES-based Threefish based	keine
<b>Ausgabefunktion</b>	
output function truncated to output fixed length	CypherMatrix: $GF(16^2)$ dreifache Permutation Zahlensystem zur Basis 62 (9 bis 11 Ziffern)
<b>Anwendungen</b> (applications)	
hashing, MAC, randomizing, PRNG, digital signatures, authentication, encryptions,	Hashfunktion, Zufallsfolgen digitale Signaturen, RNG, Authentifizierung, Verschlüsselungen

## F. Referenzen

- [#1] Patent des Autors, DPMA 19811593 vom 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] analog zu: Descartes, René, (ohne weiteren Hinweise)
- [#5] Kollisionsfrei: [telectypher.net/Kollfrei.pdf](http://telectypher.net/Kollfrei.pdf)
- [#9] NIST: National Institute of Standards and Technology, USA
- [#10] siehe: [//ehash.iak.tugraz.at/wiki/The\\_SHA-3\\_Zoo](http://ehash.iak.tugraz.at/wiki/The_SHA-3_Zoo)

München, im Mai 2010