>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 S**truktur** (z.B.: Matrix) sein. Mit der **Basisfunktion** werden dynamische Hash-verfahren begründet, die in vielen Bereichen der Kryptographie eingesetzt werden können. Eine ausführliche Darstellung der Basisfunktion lesen Sie im Artikel: Kryptographische Basisfunktion in Byte-Technik.

1 Der Hashwert Generator

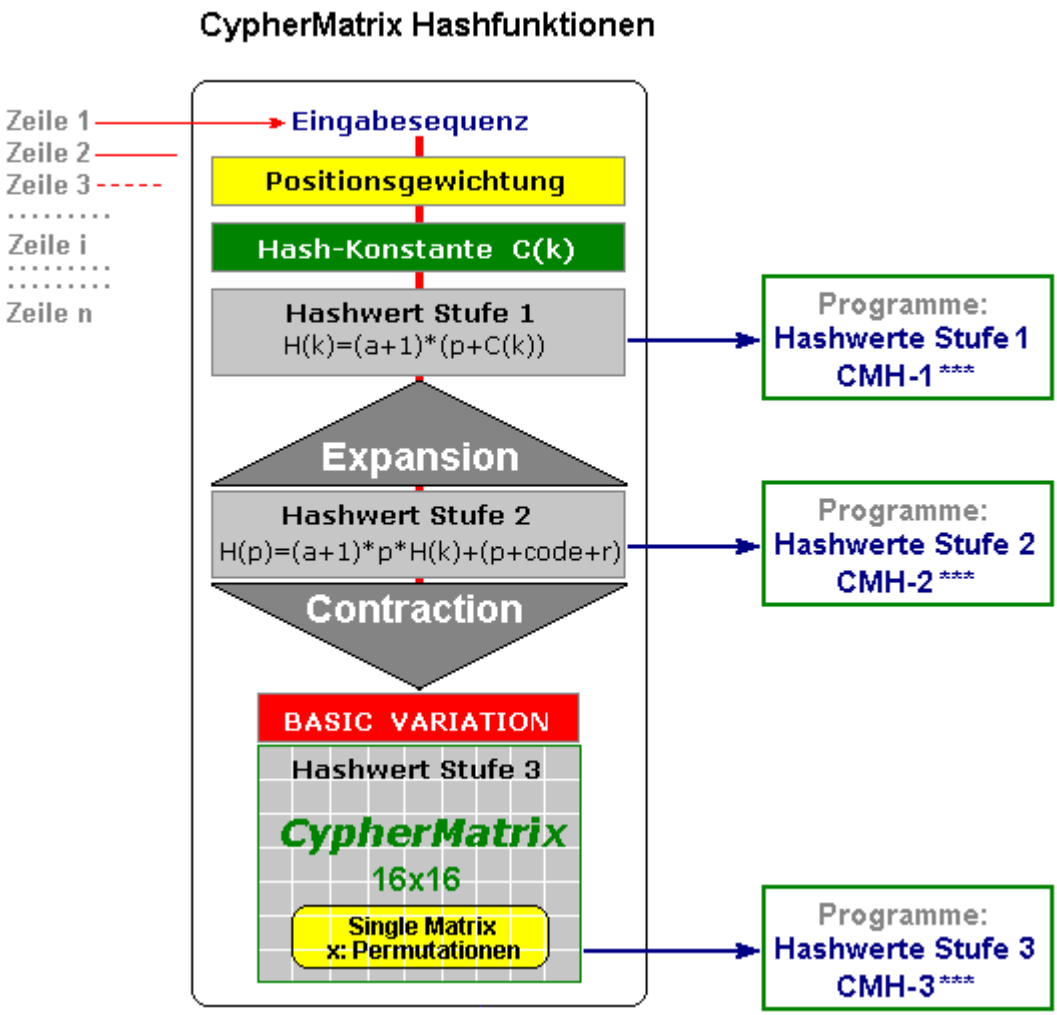
Die **Basisfunktion** bildet den Kern des Hashverfahrens [#1]. Digitale Zeichenfolgen (Dateien) werden in Klartextblöcke aufgeteilt (z.B. 64 Bytes), die in jeder Runde positionsgewichtet, mit der Hashkonstante **Ck** multipliziert, zu einer Hashfunktionsfolge erweitert, wieder zur BASIS VARIATION verdichtet und abschließend wahlweise einer dreifachen Permutation unterworfen werden. Ein Schlüssel ist nicht erforderlich. Der erste Textblock wird als Startsequenz verwendet.

Im Verlauf der Funktion stellen sich drei Stationen heraus, die eine eindeutige und kollisionsfreie Abbildung der eingelesenen Zeichenfolgen generieren und damit alle Voraussetzungen als Hashwert erfüllen:

1. Positionsgewichtung und Kollisionsfreiheit (Stufe 1),
2. Erweiterung zur Hashfunktionsfolge (Stufe 2) und
3. BASIS VARIATION mit Ableitung zur CypherMatrix (Stufe 3).

Die letzte CypherMatrix mit 16x16 Elementen ergibt in ihrer Struktur eine eindeutige und kollisionsfreie Abbildung (term). Die Umformung der CypherMatrix in prägnante Größen ergibt den gesuchten **Hashwert**. Nach den Gesetzen der Wahrscheinlichkeit entsteht eine Wiederholung identischer Strukturen erst in **256!**(Fakultät) = **8E+506** Fällen.

Das folgende Schema zeigt die Zusammenhänge:

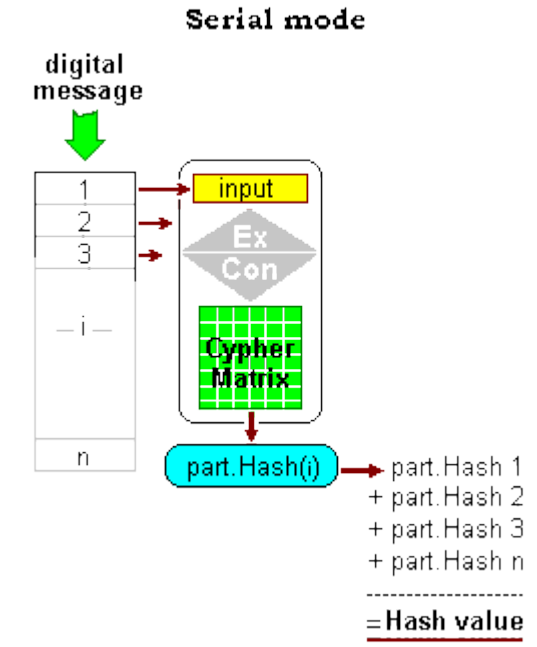


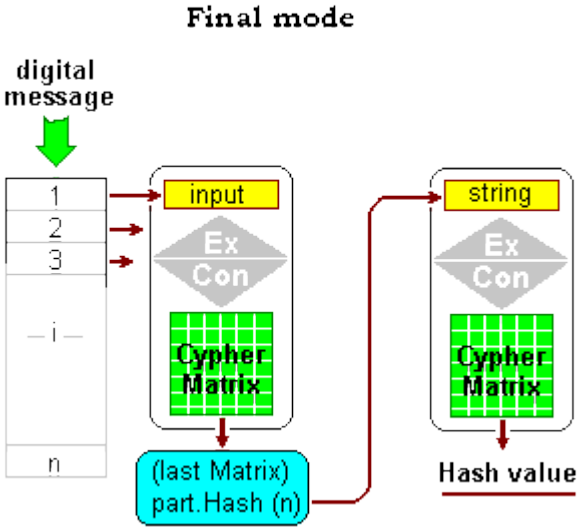
Die dynamische Hashfunktion arbeitet in vier Schritten:

1. Berechnung eines positionsgewichteten Werts **H(k)** der digitalen Eingangssequenz mit Einbindung der Hashkonstanten **C(k)** für kollisionsfreien Verlauf und Ableitung der Hashwerte Stufe 1,
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)** und Ableitung der Hashwerte Stufe 2,
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** (Stufe 3)**.**

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).**





1.2 Bestimmungsbasis für die Hashwertberechnung

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 $\mathbf{H(r_1)}$ wird berechnet.

Als Siddhartha den Hain verließ, in welchem der Buddha, der Voll (64 Bytes)

Das Verfahren durchläuft die folgenden Schritte:

1.2.1 Positionsgewichtung

Es gilt eine eindeutige Abbildung der Eingabesequenz als Bestimmungsbasis für eine Hashwertberechnung zu finden. Als Einstieg wird eine Verknüpfung der Eingabe durch Addition der Zeichen (Bytes) vorgenommen:

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

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

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

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

Der für **H(k)** errechnete Wert ist jedoch weit davon entfernt, als Bestimmungsbasis für die Hashwertberechnung zu dienen. Es müssen noch weitere Merkmale hinzukommen, insbesondere **Positionsgewichtung** und **Kollisionsfreiheit,** um eindeutige Ergebnisse zu erzielen.

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

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

1.2.2 Kollisionsfreiheit

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).

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

$$\mathbf{C(k) = 64*62 + 1 = 3969}$$

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

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

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

$$\mathbf{H_k = 23733423}$$

2 Hashwerte Stufe 1

Der Bestimmungswert **H**(**k**) ist zwar noch relativ klein, weist aber Eigenschaften eines Hashwertes auf: das Ergebnis ist eindeutig und kollisionsfrei. Die Bestimmungsformel ist Grundlage für die Hashberechnung der ersten Stufe: Programm: **CMH-1 fmx.exe**

CMH-1 fmx / 64 / 62 / 77 / 1
Serielle Rundenwerte in Hesse.txt

dezimal	Basis 62	Runde
23733423	1ba99	1
23993236	1cfjg	2
23262327	1Zbap	3
22604177	1WqNV	4
23336732	1Zuwu	5
23289806	1Zik2	6
23602377	1b23V	7
22785118	1XbRu	8
24658487	1fSnX	9
24858336	1gImu	10
22240069	1VJen	11
4111517	HFan	12

---
dezimal: 262475605
Basis 16: FA50F55
Basis 62: **HlJtV**

Mit den Summen der bisherigen **seriellen Ergebnisse** (alle drei Notationen) und der Anzahl der Runden wird eine neue Eingabesequenz (**R)** gebildet, die zusätzlich als letzte Runde dem Hashwert Generator (Abschnitt 1, 2) unterworfen wird,

Eingabestring **(R)** = *Basis 62 + dezimal + Basis 16 + Runden*
Eingabestring **(R)** = *HlJtV262475605FA50F5512*

Die Berechnung führt zu folgendem Hashwert:

Hashwert letzte Runde:
dezimal: 11524233584
Basis 16: 2AEE5D970
Basis 62: CZuVbE

Finaler Hashwert für: HESSE.TXT
dezimal: 11786709189
Basis 16: 2BE8AE8C5
Basis 62: **CrfpUj**
==========

3 Hashwerte Stufe 2

Zur Erhöhung der Intensität wird die **Hashkonstante H(k)** eingebunden und zur Bestimmungsformel **H(p)** erweitert:

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

$$\mathbf{Hp} = 4540749690837$$

Die für **Hp** berechneten Werte bewegen sich in einem Bereich, der für die Berechnung von Hashwerten (zweite Stufe) geeignet ist. Die Ergebnisse der Bestimmungsformel sind eindeutig und kollisionsfrei. Das Programm **CMH-2 fmx.exe** berechnet Hashwerte der Stufe 2, wie folgt:

CMH-2 fmx / 64 / 62 / 77 / 1
Serielle Rundenwerte in Hesse.txt

dezimal	Basis 62	Runde
4540749690837	1HwQkxDJ	1
4459384543806	1GVcJ5aA	2
4488489259313	1H1NzVdR	3
3798334685211	14s3FFYh	4
4298812730720	1DgLU7pA	5
4357662443964	1EiaBWaS	6
4524316047345	1HeUb3pJ	7
3987829260808	18CtRB5E	8
4974677145561	1Pa58KgD	9
5082336514656	1RTb44dE	10
3979393308949	183gWmdZ	11
194203395213	3Pyrec1	12

dezimal: 48686189026383
Base 16: 2C47A3009C4F
Basis 62: **Dp99bXsl**

Mit der Summe der bisherigen **seriellen Ergebnisse** (alle drei Notationen) und der Anzahl der Runden wird eine neue Eingabesequenz (**R**) gebildet, die als letzte Runde zusätzlich dem Hashwert Generator (Abschnitt 1,2,3) unterworfen wird,

Eingabestring **(R)** = *Basis 62 + dezimal + Basis 16 + Runden*
Eingabestring **(R)** = *Dp99bXsl486861890263832C47A3009C4F12*

Die Berechnung führt zu folgendem Hashwert:

Hashwert letzte Runde:
dezimal: 108437729871
Basis 16: 193F64724F
Basis 62: 1uMbf95

Finaler Hashwert für: HESSE.TXT
dezimal: 48794626756254
Basis 16: 2C60E2650E9E
Basis 62: **Dr3WDD1q**
==========

4 Hashwert Stufe 3

Zur Verdichtung der Bestimmungsbasis wird die **Hashfunktionsfolge H(p)** eingeführt., die die Eingangsdaten zu einer umfangreichen Folge in einem höherwertigen Zahlensystem expandiert und zum Bestimmungswert **H(p)** addiert.

4.1 Expansion

Das Zahlensystem der Expansion ist wählbar zwischen 64 bis 96. Hier wird die **Basis 77** festgelegt. Für jedes Zeichen der Eingabe-Seqzenz errechnet das Verfahrenden den dezimalen Wert ($\mathbf{s_i}$), der zu ($\mathbf{d_i}$) (Ziffern im Zahlensystem zur Basis 77) umgewandelt wird. Gleichzeitig ermittelt das Verfahren die Summe aller einzelnen Ergebnisse ($\mathbf{s_i}$) als zusätzlichen Wert **H(p)** zur Bestimmung verschiedener Steuerungsparameter.

$$\mathbf{s_i} = \mathbf{(a_i + 1) * p_i * H_k + p_i + code + r_j}$$

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

$$\mathbf{Hp} = d_1 + \mathbf{d_2 + d_3 + \ldots + d_i + \ldots d_m}$$

(m = Anzahl der Zahlen im System zur Basis 77)

Das Zahlensystem zur Basis 77 umfasst die folgenden Ziffern:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz&#@àáâãäåæçèéêë
(definiert vom Autor, nicht standardisiert)

Als Bestimmungsdaten ergeben sich die folgenden Werte:

	dezimal	Basis 62
Hashkonstante C(k):	**3969**	**121**
positionsgewichteter Wert (H_k):	**23733423**	**1ba99**
partieller Zwischenwert (H_p):	**4540749690837**	**1HwQkxDJ**

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

Variante	(H_k MOD 11) +1	=	**11**	Beginn Rückumwandlung
Alpha	((H_k + H_p) MOD 255)+1	=	**241**	Offset BASIS VARIATION
Theta	(H_k MOD 32) +1	=	**16**	Variation Rückumwandlung

Bei der Generierung der Hashfunktionsfolge ergibt sich beispielsweise für die Teilsequenz "**Buddha**" an den Positionen 49 bis 54 der Eingabesequenz folgende Berechnung:

char	pi		Hk	(ai+1)*pi*Hk		Si	Basis 77
	(ai+1)	(ai+1)*pi				pi+code+r	
B	67	49	3283	23733423	77916827709	51	77916827760	SycvZ2
u	118	50	5900	23733423	140027195700	52	140027195752	puRpyx
d	101	51	5151	23733423	122250861873	53	122250861926	jCpëxd
d	101	52	5252	23733423	124647937596	54	124647937650	k3áki5
h	105	53	5565	23733423	132076498995	55	132076499050	mzEMRY
a	98	54	5292	23733423	125597274516	56	125597274572	kUâ490
						Summe:	4540749690837	LyfãvYq

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

ih6è31åDël**d33èFmê**1C9Dä63qhGGn5iUHèh6FPJjç76dáUc8M0éêâ8jnb@XB74hQDCNän&
7BéiEFaC2N6xU4QFDlHED2tyéFFs7gCHdçZQD5cORdLCzsâSaI3YmoiKYXà@OMTn&Kk6çt
6TQQ6gëRtNJbYEnRHPFTFQwfæmwQèUmLxQ#ç#AézX5WãCCmGBkD9gHëBZVkHOvTY4çYc
2A W5m5ccænzWWXêái3lbL3V1äZ5iáXZbvcosobhOçëçfaXzljCuNêákdàfQêCfAyOèKlUCcD
uDãXCwoSycvZ2puRpyxjCpëxdk3áki5mzEMRYkUâ490Lrêã4vGFpegwoa&nhápâCY&Ixbá
aZJHRysY&keëàFEyãGLroyGHQWpzC&@uV

4.2 Kontraktion

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

Die ersten fünf Rückrechnungen ab **Variante** = **11** zeigen sich wie folgt:

3 Ziffern Basis 78	dezimal	Modulo 256	- Theta		Element
d33	237513	201	16	185	185
33è	18559	127	16	111	111
3èF	23961	153	16	137	137
èFm	445350	166	16	150	150
Fmê	95079	103	16	87	087
. . .	. . . . .	. . .	. .	. . .	. . . .

Die BASIS VARIATION ergibt die folgende Struktur:

BASIS VARIATION (256 Elemente)

185 111 137 150 087 139 182 101 235 015 240 155 166 063 234 204
065 035 038 074 213 095 093 244 191 051 029 011 122 091 056 071
172 006 194 132 142 196 018 199 181 206 217 247 119 250 229 009
149 130 175 052 214 218 163 103 195 241 096 151 242 020 014 243
094 112 042 211 128 156 160 068 058 251 120 145 072 040 069 161

183 118 158 123 057 053 231 148 036 209 117 030 223 212 184 215
144 113 115 177 147 062 233 236 092 207 232 106 255 197 200 202
024 180 219 189 073 066 076 086 033 208 080 108 174 075 070 176
216 253 121 043 203 220 162 097 146 124 034 125 228 178 109 179
192 227 164 198 081 210 152 173 031 039 041 126 141 054 044 157
082 025 077 083 098 078 131 099 079 186 221 201 055 187 248 222
159 188 110 045 205 190 153 046 165 037 100 193 114 127 224 140
133 129 019 154 143 225 167 102 047 105 104 134 230 226 135 048
136 084 237 085 168 238 239 245 088 246 013 249 064 252 116 138
169 254 170 003 059 000 090 171 001 002 004 005 007 008 010 060
012 016 017 021 049 022 089 107 032 023 026 027 028 050 061 067

Die Zahlen der BASIS VARIATION (16x16) bilden die Grundlage für die folgenden Ableitungen:

4.3 Generierung der CypherMatrix (CM)

Die dezimalen Zahlen der BASIS VARIATION (16x16) werden direkt auf die 256 Werte des erweiterten ASCII-Zeichensatzes bezogen und ergeben so die erste **CypherMatrix (CM1)** mit 16x16 Elementen.

Ableitung der ersten CypherMatrix (CM1) aus der BASIS VARIATION

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

Die Elemente der ersten CypherMatrix **(CM1)** können direkt als Bestimmungsbasis zur Hashwertberechnung verwendet werden. Zur Erhöhung der Sicherheit wird eine zusätzliche **dreifache Permutation** eingeführt mit dem Ergebnis der **CypherMatrix (CM3)**. Für die Permutation sind drei Varianten (A, B und C) möglich:

In Pseudo-Code geschieht folgendes:

4.3.1 Einfacher Ersatz des Index: i (Variante A)

```
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
                 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 Versetzter Austausch der Indizes: i,j (Variante 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 Dynamische Generierung der Indizes: i,j (Variante C)

Alle Indexwerte (16x16) werden in einem gesonderten Index-Array **Index$(16,16)** neu generiert (Anwendung der CypherMatrix Funktion):

```
           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
```

Die mit **Variante B** permutierte **CypherMatrix** (CM3) zeigt sich wie folgt:

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

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

4.4 Hashwertberechnung H(r)

Die Umformung geschieht in der Form, dass der Inhalt der letzten Cypher Matrix (CM3) in seiner Gesamtheit noch einmal der dynamischen **Hash Funktion** unterworfen wird. Alle Elemente ($\mathbf{e_i}$) der CypherMatrix (256 ASCII-Zeichen) werden seriell in einem **CypherString** mit der Länge **n = 256** geordnet und der Funktion als weitere Eingabesequenz übergeben.

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

Im vorliegenden Fall wird folgender CypherString der Hash Funktion unterworfen:

CüæÁÝ"Vé5€4Â#¹Aì=#@†dº#aL>9Ó¯#,¬Gê2#ùh%O¢B"{*žp•
8?##i¥c˜Ül±½sv^óå[¦##ö/.ƒÒËQ+Ûq·¡#úz›##Xf™N¾bÆy´□×
E#w#ð##õ§ïáÍS¤ý#Ê¸(ò÷## «kZî□-MãØ°ÈÔH—Ù3ë¿eY¨šn#À³
FÅß‘`Îñµô¶#;U#¼R□mK□#xuûÃÇ]‹1#í□ŸÞ,²®jlèÑ:g#_W#ªT…
Œø6ä□}PÏ$D£ÄÕ–#þˆ0à»□7~"Ð\" ÚŽJ‰#©Š‡târÉ)|!ìçœÖ„&o<

Die Hashkonstante **C(k)** errechnet sich wie folgt:

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

Zur Vermeidung von Kollisionen wird der CypherString **positionsgewichtet.**

Der CypherString wird dem Hashwert Generator (Abschnitte 1,2,3,4: Positionsgewichtung, Kollisionsfrei, Expansion und Kontraktion) unterworfen. Dabei werden die Reihen ($\mathbf{r_j}$) einbezogen, damit bei gleichem Reiheninhalt ungleiche Hashwerte errechnet werden.

$$\mathbf{CM(k)} = \sum_{i=1}^{256} (e_i + 1) * (p_i + C_{k,j} + r_j)$$

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

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

Zur besseren Handhabung wird der Hashwert im Zahlensystem zur Basis 62 ausgewiesen, das die folgenden Ziffern umfasst:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
(definiert vom Autor, nicht standardisiert)

Der endgültige Hashwert $\mathbf{H(r_1)}$ der ersten Eingabesequenz errechnet sich abschließend wie folgt:

		dezimal	Basis 62
CM(k)	=	**2143440436**	**2L3elw**
$H(r_1)$	=	**9313548776114448**	**gegFRR4ZE**

In den weiteren Entwicklungen wird die **Variante B** (versetzter Austausch der Indizes: i,j) angewendet. Für Hashwertberechnungen digitaler Folgen bestehen weitere Lösungs-möglichkeiten.

5 Alternative Hashwertberechnung

Sind digitale 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 durch-laufen. 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 (Version: ***serial mode SM***)
   2. mit **dreifacher** Permutation

B. Berechnung im „**final mode**“
   1. auf Basis aller seriellen Ergebnisse (Version: ***final mode FM***)
      a) **ohne** Permutation
      b) mit **dreifacher** Permutation
   2. auf Basis der **letzten** CypherMatrix (Version: ***last cycle LC***)
      a) **ohne** Permutation
      b) mit **dreifacher** Permutation
   3. auf Basis **aller** Runden (Version: ***all cycles AC***)
      a) **ohne** Permutation
      b) mit **dreifacher** Permutation

Die einzelnen Fälle sind in der folgenden Übersicht zusammengestellt:

Dynamische Hashfunktionen

Programm	Basis	mode		Permutation		Hashwert
		serial	final	ja	nein	
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

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

5.1 Hashwertberechnung im „serial mode“

Im **„serial mode“** addiert das Verfahren alle Hashwerte **H(r)** der einzelnen Runden (wobei: **r → 1...m**) zum seriellen Hashwert **H(s)** der Datei. Dabei kann der jeweilige Rundenwert entweder mit **dreifacher** Permutation der CypherMatrix oder **ohne** Permutation ermittelt werden.

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

Die Textdatei Hesse.txt hat 12 digitale Eingabesequenzen (Runden: m = 12). Der serielle Hashwert im „**serial mode**“ **H(s)** umfasst somit die Summe aller 12 vorhergehenden Rundenwerte. Dabei entstehen für die Hashwertberechnung unterschiedliche Ergebnisse abhängig vom Verlauf, entweder **ohne** oder mit dreifacher **Permutation B**:
Das Programm „**CMH-3 sm“** im „serial mode“ ohne Permutation errechnet für die Datei **Hesse.txt** den folgenden Hashwert:

CMH-3sm / 64 / 62 / 77 / 1
Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde	CM(k)
9234032681436729	gI6K43Huz	1	2L3V8n
9007738084237732	fFqHNf7Ke	2	2L3CGo
8878922871117668	efGPoCQ7g	3	2L35Cs
8649552282617616	dc8DXiDz6	4	2L2lxc
9172130389671616	g0WUwRVDM	5	2L3vo0
8905669342207940	emrImNNo4	6	2L3Y4Q
9145007650996929	fsozG46KH	7	2L49bN
8806614557192305	eKjO2y7zF	8	2L3d9b
9838304196752868	j3grTYGEW	9	2L5mdm
8853716103107001	eY6dUcfW5	10	2L3zwT
9178963550374288	g2SndtRZI	11	2L4lsS
9327964900341473	gim3HpBNB	12	2L5CSl

Serieller Hashwert für: HESSE.TXT
dezimal: 108998616610054165
Basis 16: 1833DAA100D2015
Basis 62: **83DJIjNjOP** **H(s)**

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

CMH-3 smx / 64 / 62 / 77 / 1
Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde	CM(k)
9313548776114448	gegFRR4ZE	1	2L3elw
8987598602496448	fA7iEc92m	2	2L39pY
9327853720046505	gik5vbzlB	3	2L3xaN
8778179221223313	eCelbYFKT	4	2L31Xl
8959578203998848	f2AOiPbhA	5	2L3W3k
8706123664320369	dsCBjqf45	6	2L39th
8820884940989529	eOmcoGFLd	7	2L3WLD
8997912724259569	fD3lYnx1l	8	2L40KH
9110905154598641	fj8arAb9l	9	2L4MYL
8818252193917712	eO2H33puy	10	2L3ve8
9110518931446473	fj1nHDbzd	11	2L4daV
9038039600705732	fORkqlBrl	12	2L4dLm

Serieller Hashwert für: HESSE.TXT
dezimal: 107969395734117587
Basis 16: 17F9597E7587CD3
Basis 62: **7yV3l4lAlv** **H(s)**
==========

5.2 Hashwertberechnung im „final mode“

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

a) mit den Summen der bisherigen **seriellen Ergebnisse** (alle drei Notationen) und der Anzahl der Runden wird eine neue Eingabesequenz gebildet, die als letzte Runde zusätzlich dem Hashwert Generator (Abschnitte 1,2,3,4) unterworfen wird,
b) alle **Elemente** der letzten CypherMatrix bilden als **CypherString (CM)** eine neue Eingabesequenz und werden als abschließende Runde dem Hashwert Generator (Abschnitte 1,2,3,4) unterworfen (**Version: letzte Runde**) oder
c) die seriellen **Hashwerte H(s)** der einzelnen Runden **(ganz oder teilweise)** werden zusammengefasst und als abschließende Runde dem Hashwert Generator (Abschnitte 1,2,3,4) unterworfen (**Version: alle Runden**).

5.2.1 Serielle Ergebnisse der bisherigen Runden (Version: final mode)

Die Summen der bisherigen Ergebnisse, und zwar in allen drei Notationen, und die Anzahl der Runden bilden einen gemeinsamen Eingabestring für eine zusätzliche Runde. Dabei werden die Reihen $\mathbf{r}_j$ nicht einzeln, sondern nur im Eingabestring als Anzahl der Runden berücksichtigt.
Eingabestring **(R)** = *Basis 62 + dezimal + Basis 16 + Runden*

Summe aller Runden für: HESSE.TXT
dezimal: 107969395734117587
Basis 16: 17F9597E7587CD3
Basis 62: **7yV3l4lAlv**

Eingabestring **(R)** = *7yV3l4lAlv10796939573411758717F9597E7587CD312*
Die Berechnung führt zu folgendem Hashwert:

CMH-3fmx-B / 64 / 62 / 77 / 1
Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde	CM(k)
9313548776114448	gegFRR4ZE	1	2L3elw
8987598602496448	fA7iEc92m	2	2L39pY
9327853720046505	gik5vbzlB	3	2L3xaN
8778179221223313	eCelbYFKT	4	2L31Xl
8959578203998848	f2AOiPbhA	5	2L3W3k
8706123664320369	dsCBjqf45	6	2L39th
8820884940989529	eOmcoGFLd	7	2L3WLD
8997912724259569	fD3lYnx1l	8	2L40KH
9110905154598641	fj8arAb9l	9	2L4MYL

```
          8818252193917712     eO2H33puy     10    2L3ve8
          9110518931446473     fj1nHDbzd     11    2L4daV
          9038039600705732     fORkqIBrl     12    2L4dLm
-------------------------------------------------------------------------------
dezimal: 107969395734117587
Basis 16:   17F9597E7587CD3
Basis 62:              7yV3l4IAlv
----------------------------------------
Matrixwert letzte Runde:
dezimal:          406440880763
Basis 16:          5EA1C3AA7B
Basis 62:                79eCVwp
----------------------------------------
Finaler Hashwert für HESSE.TXT
dezimal: 107969802174998350
Basis 16:   17F95F6891C274E
Basis 62:             7yVAuiUgik
========================
```

Die Einbindung der Summen der bisherigen Rundenwerte und die Anzahl der Runden erfassen jede Abweichung in der bisherigen Analyse der digitalen Daten.

5.2.2 Hashwert der letzten CypherMatrix (Version: letzte Runde)

Ungeachtet der gewählten Blocklänge (z.B. 64 Bytes) wird aus allen 256 Elementen ($\mathbf{e_i}$) der letzten laufenden CypherMatrix eine neue Eingangssequenz zur Berechnung eines zusätzlichen **Matrixwert H(l)** generiert.

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

Der **CypherString** von 256 Bytes (in Textzeichen) wird dem Hashwert Generator (Abschnitte 1,2,3,4) unterworfen. Der so errechnete zusätzliche **Matrixwert H(l)** und die Summe aller Rundenwerte **H(s)** werden addiert. Das Ergebnis stellt den **finalen Hashwert H** dar.
Die folgende Aufstellung (Programm:**CMH-3 lcx.exe**) zeigt die Reihenfolge der einzelnen Vorgänge für die Datei HESSE.TXT:

CMH-3 lcx-B / 64 / 62 / 77 / 1
Serielle Matrixwerte in Hesse.txt

decimal	Basis 62	Runde	CM(k)
9313548776114448	gegFRR4ZE	1	2L3elw
8987598602496448	fA7iEc92m	2	2L39pY
9327853720046505	gik5vbzlB	3	2L3xaN
8778179221223313	eCelbYFKT	4	2L31Xl
8959578203998848	f2AOiPbhA	5	2L3W3k
8706123664320369	dsCBjqf45	6	2L39th
8820884940989529	eOmcoGFLd	7	2L3WLD
8997912724259569	fD3lYnx1l	8	2L40KH
9110905154598641	fj8arAb9l	9	2L4MYL
8818252193917712	eO2H33puy	10	2L3ve8
9110518931446473	fj1nHDbzd	11	2L4daV
9038039600705732	fORkqIBrl	12	2L4dLm

dezimal: 107969395734117587
Basis 16: 17F9597E7587CD3
Basis 62: 7yV3l4lAlv

Matrixwert letzte Runde:
dezimal: 9036375268636868
Basis 16: 201A8911F1B8C4
Basis 62: fNyS9KRQS

Finaler Hashwert für: HESSE.TXT
dezimal: 117005771002754455
Basis 16: 19FB020F94A3597
Basis 62: **8dt2DDccCN**
==========

5.2.3 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_i)** die jeweils letzten drei Ziffern (Zahlensystem 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. Auf diese Weise werden die Ergebnisse aller vorhergehenden Runden in die finale Hashwertberechnung eingebunden. Der Matrixwert des zusätzlichen Durchgangs **H(l)** und die Summe der seriellen Hashwerte **H(s)** werden addiert. Das Ergebnis stellt den **finalen Hashwert H** dar.

Für die Datei **Hesse.txt** errechnet das Programm **CMH-3 acx.exe** folgende Hashwerte:

CMH-3 acx / 64 / 62 / 77 / 1
Serielle Matrixwerte in Hesse.txt

dezimal	Basis 62	Runde	CM(k)
9313548776114448	gegFRR4ZE	1	2L3elw
8987598602496448	fA7iEc92m	2	2L39pY
9327853720046505	gik5vbzlB	3	2L3xaN
8778179221223313	eCelbYFKT	4	2L31Xl
8959578203998848	f2AOiPbhA	5	2L3W3k
8706123664320369	dsCBjqf45	6	2L39th
8820884940989529	eOmcoGFLd	7	2L3WLD
8997912724259569	fD3lYnx1l	8	2L40KH
9110905154598641	fj8arAb9l	9	2L4MYL
8818252193917712	eO2H33puy	10	2L3ve8
9110518931446473	fj1nHDbzd	11	2L4daV
9038039600705732	fORkqlBrl	12	2L4dLm

---
dezimal: 107969395734117587
Basis 16: 17F9597E7587CD3
Basis 62: **7yV3l4lAlv** **H(s)**

Die Eingabesequenz (**R**) für den letzten Durchgang ergibt sich wie folgt:

Hashsequenz (**R**) = *4ZE92mzlBFKTbhAf45FLdx1lb9lpuybzdBrl*

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:          243386817606
Basis 16:          38AAFC1C46
Basis 62:               4HfOHYM                  H(l)
-------------------------------------------------------------
Finaler Hashwert für: HESSE.TXT
dezimal: 107969639120935193
Basis 16:    17F95D092549919
Basis 62:           7yV82jgSKH                  H
=====================================
```

Von den vorstehend erläuterten Verfahren erscheint das Programm „**CMH-3 fmx**“ für den praktischen Einsatz am besten geeignet zu sein (Programm: DynaHash.exe). Die Länge der Hashwerte bewegt sich zwischen **9** bis **12** Ziffern im Zahlensystem zur Basis 62. Die Spanne der Hashwerte reicht somit von **62^9** bis **62^12,** bzw. in dezimalen Werten:

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

6 Sensibilität der Hashfunktion

Zur Darstellung der Sensibilität der Hashfunktionen wird in der Datei **Hesse.txt** die letzte Ziffer der letzten Zeile von ..5**3** auf ..5**2** geändert**.**

Hermann Hesse, Siddhartha, Eine indische Dichtung, Montagnola 195**3** **(Hesse.txt)**
. . . . 01100001 00100000 00110001 00111001 00110101 0011001**1**
Hermann Hesse, Siddhartha, Eine indische Dichtung, Montagnola 195**2** **(Hesse1.txt)**
. . . . 01100001 00100000 00110001 00111001 00110101 0011001**0**

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 **CMH-3 lcx.exe** führt zu folgenden Ergebnissen:

Unveränderte Textdatei

```
dezimal: 107969395734117587
Basis 16:   17F9597E7587CD3
Basis 62:              7yV3l4lAlv
-----------------------------------------
```

Matrixwert letzte Runde:
dezimal: 406440880763
Basis 16: 5EA1C3AA7B
Basis 62: 79eCVwp

Finaler Hashwert für HESSE.TXT
dezimal: 107969802174998350
Basis 16: 17F95F6891C274E
Basis 62: **7yVAuiUgik**
==========

Im letzten Bit veränderte Textdatei

dezimal: 107953898763835352
Basis 16: 17F877FBBEFD7D8
Basis 62: 7yQevQoAFU

Matrixwert letzte Runde:
dezimal: 439237882880
Basis 16: 66449E3400
Basis 62: 7jRlOPw

Finaler Hashwert für: HESSE1.TXT
dezimal: 107954338001718232
Basis 16: 17F87E6008E0BD8
Basis 62: **7yQmesZYfQ**
==========

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

	Original	Änderung	Differenz
Basis 62:	**7yVAuiUgik**	**7yQmesZYfQ**	**4OFpv83K**
dezimal:	107969802174998350	107954338001718232	15464173280118

7 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.

Um mit den **CypherMatrix 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“ als Zeichen und Zahlen verarbeitet (Zahlensysteme von zur Basis 2 bis zur Basis 256). Dementsprechend ist CypherMatrix im Grunde genommen ein rein mathematisches Verfahren.

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

8 Abschließende Bemerkungen

Ausführliche Erläuterungen und weitere Beispiele zum CypherMatrix Verfahren sind im Internet unter http://www.telecypher.net/ zusammengestellt.

Einige der im Abschnitt C. Dynamische Hashfunktionen aufgeführten Programme können von http://www.telecypher.net/ZUSENDEN.HTM herunter geladen und ausführlich getestet werden.

9 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) Kartesisches Koordinatensystem: Sachverhalt = Gegenstand * Ort * Zeit
[#5] Kollisionsfrei: telecypher.net/Kollfrei.pdf
[#9] NIST: National Institute of Standards and Technology, USA
[#10] siehe: //ehash.iak.tugraz.qt/wiki/The_SHA-3_Zoo

München, im Juli 2011