MAGENTA

Ein Algorithmus der Deutschen Telekom AG

21. Mai 2002

Inhaltsverzeichnis

1 Einleitung

1.1 Name

Der Name MAGENTA steht als Abkürzung für:

Multifunctional
Algorithm for
General-purpose
Encryption and
Network
Telecommunication
Applications

1.2 Entwicklung

Im Sommer 1990 hat die Entwicklung des MAGENTA Algorithmus begonnen.
Ursprünglich sollte die Butterfly-Struktur verwendet werden, die dann jedoch wegen Hardware-Gründen durch die Shuffle-Struktur ersetzt wurde.
In den folgenden Jahren wurde auch die Nutzung des MAGENTA Algorithmus als Hashfunktion untersucht. Dabei waren noch weitere Punkte zu beachten, die als potentielle Schwachstellen angesehen werden konnten.

Der MAGENTA Algorithmus sollte zunächst für Sicherheitsmodule verwendet werden. Deswegen wurde ein externes Gutachten an die Firma SIT in Auftrag gegeben. In Folge dessen wurden einige kleine Änderungen am Algorithmus vogenommen, die deren Sicherheit verbesserte.
Im Abschlußbericht der Untersuchung des MAGENTA Algorithmus kommt man zu folgendem Schluß (Zitat):

> "Auswertbare Schwachstellen der neuen Algorithmusvariante wurden nicht gefunden. Nach dem bisherigen Erkenntnisstand kann man davon ausgehen, daß der MAGENTA Algorithmus bei fachgerechter Anwendung kryptanalytischen Angriffsversuchen der verschiedensten Art hohen Widerstand entgegensetzt."

In dem Gutachten der Firma SIT (Gesellschaft für Systeme der Informationstechnik mbH) wurden folgende Eigenschaften des Algorithmus untersucht:

- Algebraische Eigenschaften
- Avalanche Eigenschaften
- Statistische Eigenschaften
- Verhalten gegenüber der Differentialkryptanalysis
- Verhalten gegenüber der Linearen Kryptanalysis
- Charakterisierung von schwachen Schlüsseln
- Charakterisieung der Designkriterien

1.3 AES

Der Advanced Encryption Standard (AES) sollte eine symmetrsiche Blockchiffre sein, die mindestens eine Blockgröße von 128 Bit und Schlüsselgrößen von 128 Bit, 192 Bit und 256 Bit zuläßt. Das National Institute of Standards and Technology (NIST) rief dazu auf, bis zum 12. September 1997 verschiedene Algorithmen und Vorschläge einzusenden.

Am 20. August 1998 fand dann die erste AES Konferenz statt. NIST kündigte den Erhalt von 15 Algorithmen an, die von Kryptographen aus aller Welt eingesandt wurden. Somit begann die öffentliche Presentation und Auswertung der Algorithmen.

Am 22./23. März 1999 fand die zweite AES Konferenz statt. Die Ziele der Konferenz waren es, die besten fünf Kandidaten auszuwählen, die sich für die zweite Runde qualifizieren sollten. Dabei wurden die Algorithmen zwecks Sicherheit, Effizienz und Flexibilität untersucht.

1. Sicherheit:

   Wenn ein Algorithmus einen k-Bit Schlüssel benutzt, wird die Sicherheit gemessen, indem man untersucht, ob der Algorithmus 2^k-sicher ist, d.h. ob es Methoden gibt, die wesentlich besser sind als ein Brute-Force-Angriff.
   Manchmal ist die Schwäche eines Algorithmus sehr offensichlich, so wie bei MAGENTA. Die Schlüsselaufstellung (d.h. die Reihenfolge, in der die Schlüssel-Bits an den Algorithmus übergeben werden) war schlecht designed und wurde von den anderen Kryptographen schon während der ersten öffentlichen Vorstellung des Algorithmus entdeckt.

2. Effizienz/Performanz:

   Alle eingereichten Algorithmen wurden auf unterschiedlichen Plattformen mit 8 Bit, 32 Bit und 64 Bit getestet. MAGENTA zählte in allen drei Untersuchungen mit zu den fünf langsamsten Algorithmen.

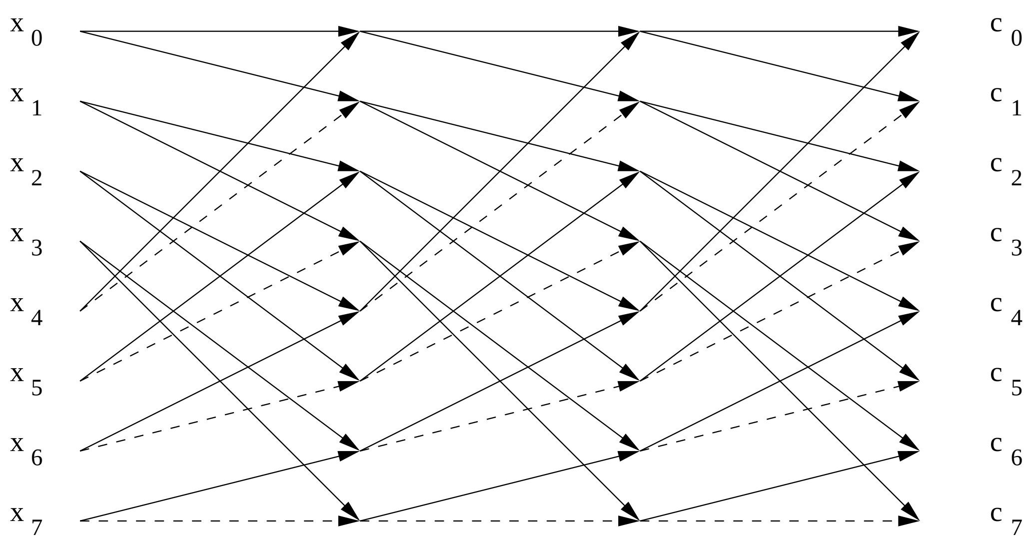
Eli Biham präsentierte dann in der Konferenz die Ergebnisse, die er und andere Kryptographen während der ersten AES Konferenz bei MAGENTA erzielt hatten. Sie fuhren einen einfachen Angriff, indem sie die Symmetrie der Schlüsselaufstellung ausnutzten. Für einen 128 Bit Schlüssel benötigte der Angriff 2^{64} Klartexte und 2^{64} Analyseschritte.

Zum Ende der zweiten AES Konferenz sprach Klaus Huber noch einmal zugunsten von MAGTENTA. Er behauptete, daß die Kritik an MAGENTA übertreiben und teilweise falsch sei, z.B. fechtete er die Behauptung an, daß es schwer sein würde MAGNENTA in Hardware schneller als 128 MBit/s zu implementieren. Er verteidigte weitere Aspekte von MAGENTA ... die Schwäche der Schlüsselaufstellung könnte einfach beseitigt werden. Ausserdem zählte MAGENTA mit zu den besten Kandidaten zwecks Speichernutzung, Widerstand gegen "implementation attacks" und Schlüsselgenerierung. MAGENTA war sehr schnell in Hardware und die Software Performanz könnte verbessert werden, indem man 16 Bit Tabellen benutzt.
Einen weiteren Pluspunkt sammelte MAGENTA, da im Algorithmus keinerlei arithmetische Operationen oder Rotationen verwendet wurden.

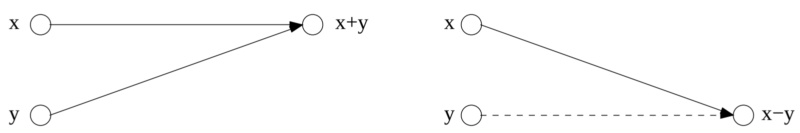
2 Der Algorithmus

2.1 Shuffle-Struktur

Wie schon eingangs erwähnt, basiert der MAGENTA Algorithmus auf der Shuffle-Struktur, d.h. einer Signalfluß-Struktur für die schnelle Walsh-Transformation (Hadamard-Transformation), bei der es sich um eine Abwandlung der schnellen Fourier-Tranformation handelt. Die Verwendung der l-stufigen Shuffle-Struktur für 2^l Eingangsvariablen garantiert, daß jede Eingangsvariable jede Ausgangsvariable beeinflußt. Die bei MAGENTA verwendete Shuffle-Struktur wird in folgendem Bild für $2^l = 8$ Eingangsvariablen $x_0, \ldots, x_7$ dargestellt.

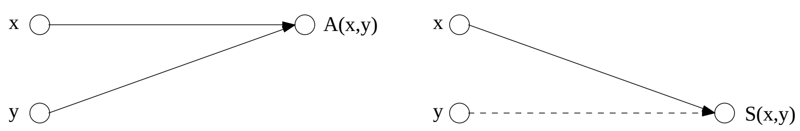


Wie man leicht sieht, gehen von jedem Knoten zwei Pfeile aus, ausgenommen die Endknoten $c_0, \ldots, c_7$, und es treffen in jedem Knoten zwei Pfeile zusammen, ausgenommen die Anfangsknoten $x_0, \ldots, x_7$. Die entsprechenden Eingangswerte werden dann im Knoten wie folgt berechnet:

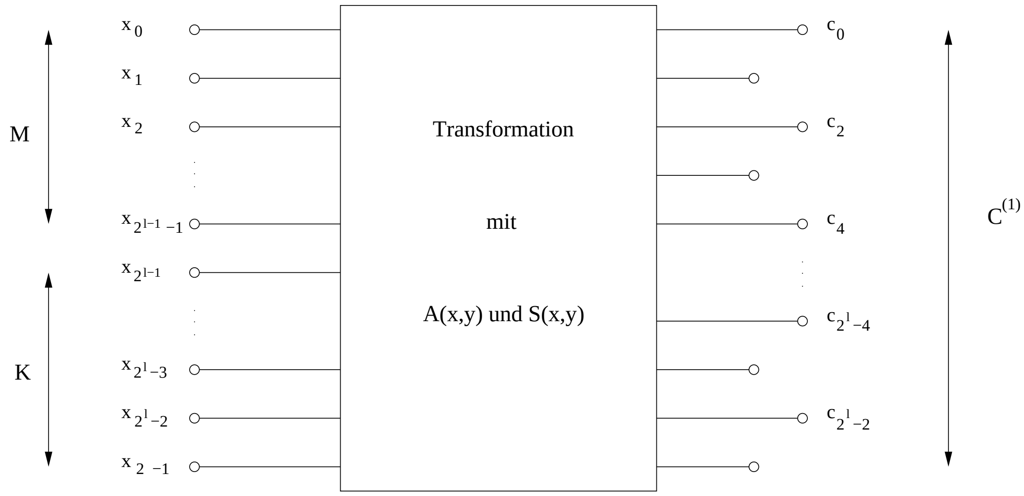


Treffen zwei Pfeile in einem Knoten zusammen, so werden die ursprünglichen Werte x und y addiert. Bei den gestrichelten Pfeilen wird zuvor noch das Vorzeichen des Eingangswertes invertiert, so daß man $x - y$ gemäß obiger Abbildung erhält.

Für das Kryptomodul in MAGENTA werden nun die arithmetischen Operationen Addition und Subtraktion durch nicht-lineare Operationen $A(x, y)$ und $S(x, y)$ ersetzt.



Auf diese Weise erhält man dann das folgende Kryptomodul:



Sei $X = (x_0, \ldots, x_{2^l-1})$ ein Vektor der Länge 2^l, wobei $x_i, i = 1, \ldots, 2^l$ Variablen sind, die je m Bit enthalten. X ist der Eingangsvektor für die Transformation, die mit den Operationen $A(x,y)$ und $S(x,y)$ der schnellen Walsh-Transformation beschrieben wird. Der Ausgangsvektor $C^{(1)}$ besteht aus dem gerade indizierten Teil $C_g^{(1)} = (c_0, \ldots c_{2^l-2})$ und dem ungerade indizierten Teil $C_u^{(1)} = (c_1, \ldots, c_{2^l-1})$. Dies führt dazu, daß die Kenntnis von $C_g^{(1)}$ (bzw. $C_u^{(1)}$) alleine nicht ausreicht, um die Transformation zu invertieren. Das sieht man auch im ersten Bild der Shuffle-Struktur. Denn wenn man nur die geraden (ungeraden) Komponenten des Ausgangsvektors kennt, so hat man nicht gleichzeitig beide Werte $A(x,y)$ und $S(x,y)$ eines Paares (x,y), was somit die Invertierung der Kryptotransformation erleichtern würde.

2.2 Operationen

In diesem Abschnitt werden die gewählten Operationen für den MAGENTA Algorithmus erläutert, die auf endlichen Körpern basieren.

Die Eingangsvariablen x_i zu je m Bit werden nun je nach Kontext entweder als m-Bit Integer oder als Elemente des endlichen Körpers $GF(q)$ mit $q = 2^m$ angesehen. Beim MAGENTA Algortihmus ist $m = 8$ gewählt, also $GF(q) = GF(256)$. Das j-te Bit von x_i für $j = 1, \ldots, 8$ wird als Koeffizient eines Polynoms mit dem Grad 7 betrachtet. x_i wird also beschrieben durch:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0.$$

Um sicherzustellen, daß jedes Ergebnis nur acht Bit benötigt, werden alle Berechnungen modulo dem folgenden Polynom $p(x)$ genommen:

$$p(x) = x^8 + x^6 + x^5 + x^2 + 1.$$

Nun ersetzt man die gewöhnliche Addition und Subtraktion von zwei Zahlen x und y in der Walsh-Transformation durch Operationen, die die wie folgt definierte Funktion $F(x, y, \circ)$ mit der Rundenfunktion f benutzen:

$$\begin{aligned} F(x,y,\circ) &:= f(x \circ y) \\ f(x) &:= \begin{cases} 2^x & , x \neq q-1\,(=255) \\ 0 & , x = q-1\,(=0) \end{cases} \end{aligned}$$

Die Operation $\circ$ ist eine Operation, die auf unterschiedliche Art vom Entwerfer frei gewählt werden kann. Mögliche Operationen sind zum Beispiel die Addition modulo $q = 2^m$ oder die bitweise

Exklusiv-Oder Operation $\oplus$, die für den MAGENTA Algorithmus gewählt wurde.
Bei kleinen endlichenKörpern $GF(2^m)$, d.h. etwa $4 \leq m \leq 12$, können die Operationen mittels Tabellen realisiert werden. Damit folgt dann, daß mit der Funktion F eine recht schnelle Operation definiert wird.
Weitere Beispiele für die Auswahl von $F(x, y, \circ)$ mit $A(x, y)$ und $S(x, y)$ werden hier angegeben:

1. $A(x,y) := F(x,y,\oplus)$ und $S(x,y) := A(y,x) = F(y,x,\oplus)$
2. $A(x,y) := F(x,y,+)$ und $S(x,y) := A(y,x) = F(y,x,+)$
3. $A(x,y,) := F(x,y,\oplus)$ und $S(x,y) := F(y,x,+)$

Für den MAGENTA Algorithmus hat man die erste Variante gewählt.

2.3 Schlüsselexpansion

Der Benutzerschlüssel K wird in Teilschlüssel gleicher Länge aufgespalten. Die Amzahl der Teilschlüssel ist abhängig von der Größe des Benutzerschlüssels:

128 Bit: $K = K_1 || K_2$

192 Bit: $K = K_1 || K_2 || K_3$

256 Bit: $K = K_1 || K_2 || K_3 || K_4$

Damit besitzen alle Teilschlüssel eine Länge von 64 Bit.

2.4 Verschlüsselung

Sei $C^{(1)} = T(K, M)$ die im letzten Bild dargestellte Transformation. Durch $C^{(1)}$ wird schon die wesentliche Kryptofunktion des universellen Kryptomoduls beschrieben.

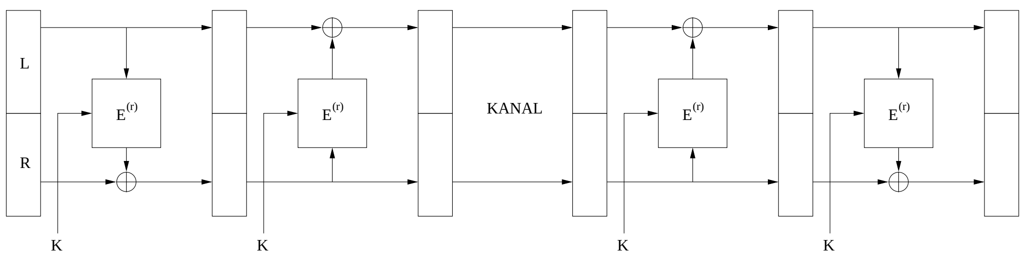
Nun seien darauf aufbauend die weiteren Kryptofunktionen wie folgt definiert:

$$\begin{aligned} C^{(1)} &= T(K, M) \\ C^{(j+1)} &= T\left(K \oplus C_g^{(j)}, M \oplus C_u^{(j)}\right) \end{aligned}$$

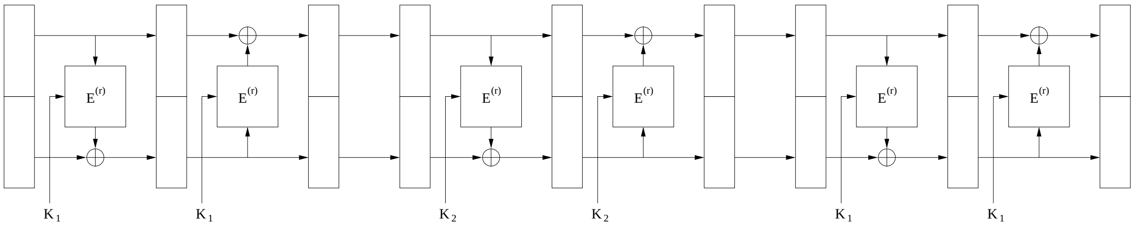
für $j = 1, 2, \ldots, r$. Bei MAGENTA wurde $r = 3$ gewählt. Die verschlüsselte Nachricht $C = E^{(r)}$ erhält man dann wie folgt:

$$C = E^{(r)}(K, M) = C_g^{(r)}.$$

Um nun einen vollständigen Verschlüsselungsalgorithmus zu erhalten, wurde die Feistel-Struktur verwendet, da es für eine Feistelchiffre nicht notwendig ist, die benutzte Kryptoabbildung $E^{(r)}$ invertieren zu können. Das Prinzip eines Feistel-Netzwerks wird im folgenden Bild dargestellt.



Der gesamte MAGENTA Algorithmus besteht nun bei einer Schlüssellänge von 128 Bit aus drei hintereinander ausgeführten Feistel-Konstruktionen wie in der folgenden Abbildung dargestellt.



Die Anzahl und die Reihenfolge der eingesetzten Teilschlüssel variiert mit der gegebenen Schlüssellänge. Die folgende Tabelle gibt darüber Aufschluß:

Runde	128 Bit	192 Bit	256 Bit
1	K_1	K_1	K_1
2	K_1	K_2	K_2
3	K_2	K_3	K_3
4	K_2	K_3	K_4
5	K_1	K_2	K_4
6	K_1	K_1	K_3
7			K_2
8			K_1

Bei MAGENTA operiert also damit die Verschlüsselungfunktion $E^{(r)}$ auf je 64 Bit. Die benutzten Schlüssel haben ebenfalls 64 Bit. Die Eingangs -und Ausgangsdaten haben immer $2 * 64 = 128$ Bit.

2.5 Entschlüsselung

Zur Entschlüsselung der Daten durch eine Entschlüsselungsfunktion D kann die Verschlüsselungsfunktion E eingesetzt werden, wenn man die Bytes vor und nach dem Entschlüsseln permutiert:

$$D_K(C) = \pi(E_K(C))$$

mit

$$\pi(b_0||b_1 \ldots ||b_{15}) = b_8||b_9||\ldots||b_{15}||b_0||\ldots||b_7$$

Dabei sind die $b_i, i = 0, \ldots 15$ die 16 Bytes eines 128 Bit großen Datenblockes.

3 Angriffe

Im folgenden werden Algorithmen angegeben, mit denen man MAGENTA attackieren kann. Dabei sei die Schlüssellänge von K auf 128 Bit festgelegt. Der Schlüssel K wird in zwei Teilschlüssel K_1 und K_2 zerlegt. Diese Teilschlüssel werden gemäß obiger Tabelle benutzt: K_1 in den Runden 1,2,5,6 und K_2 in den Runden 3,4.
X_0 sei der Klartext, X_1 das Ergebnis nach der ersten Runde ... X_i das Ergebnis nach der i-ten Runde und X_6 der Kryptotext. Die Daten X_i werden jeweils in zwei Hälften unterteilt: X_i^T der obere Teil und X_i^B der untere Teil.

3.1 Angriff bei frei wählbarem Klartext

Der Angriff bei frei wählbarem Klartext benutzt 2^{64} Klartexte und 2^{64} Stufen der Analyse.

1. Wähle einen willkürlichen Klartext X_0.
2. Man beschaffe sich den Kryptotext X_6 von X_0 mit dem unbekannten Schlüssel K.
3. Versuche alle 2^{64} Möglichkeiten für $K_1^{'}$ und berechne dafür folgendes:
   (a) Verschlüssele X_0 teilweise (für die ersten zwei Runden), so daß man einen Kandidat für X_2 erhält.
   (b) Wähle einen beliebeigen Text $X_2^{'}$, so daß $X_2^{'T} = X_2^T$ gilt.
   (c) Entschlüssele $X_2^{'}$ mit dem angenommenen Teilschlüssel $K_1^{'}$, um $X_0^{'}$ zu erhalten.
   (d) Man besorge sich den Kryptotext $X_6^{'}$ von $X_0^{'}$ mit dem unbekannten Schlüssel K.
   (e) Entschlüssele teilweise X_6 und $X_6^{'}$ mit dem angenommenen Teilschlüssel $K_1^{'}$, um X_4 und $X_4^{'}$ zu berechnen.
   (f) Berechne von X_2 und X_4 das Ergebnis der $E^{(r)}$-Funktion in Runde 3 und analog für $X_2^{'}, X_4^{'}$
   (g) Sind die Ergebnisse unterschiedlich, so entferne $K_1^{'}$
4. Fertige eine Liste aller Schlüssel an, deren Werte bei (f) das gleiche Ergebnis lieferten.

Der richtige Schlüssel muß jetzt in dieser Liste vorhanden sein, da die Gleichheit von zwei Eingaben an die Funktion $E^{(r)}$ auch eine Gleichheit der Ergebnisse bewirkt. Es ist zu erwarten, daß die Liste nur wenige Kandidaten enthält, so daß die falschen Schlüssel durch eine zusätzliche einfache Verschlüsselung herausgefiltert werden können.

3.2 Angriff bei bekanntem Klartext

Den vorigen Angriff kann man in einen Angriff mit bekanntem Klartext konvertieren. Dabei werden 2^{33} bekannte Klartexte und 2^{97} Phasen der Analyse benötigt.
Bei diesem Angriff erhält der Angreifer 2^{33} Klartexte mit den dazugehörigen Kryptotexten und sucht dann nach Kollisionen von X_2^T:

1. Versuche alle 2^{64} möglichen Werte $K_1^{'}$ von K_1 und berechne folgendes:
   (a) Verschlüssele alle 2^{33} X_0's für die ersten beiden Runden, um ein Ergebnis für X_2 zu erhalten.
   (b) Suche nach Kollisionen von X_2^T in den erhaltenen Ergebnissen.
   (c) Entschlüssele alle Paare von Kryptotexten X_6 die mit X_2^T kollidiert sind für die letzten zwei Runden, um einen Kandidat für X_4 zu erhalten.
   (d) Berechne die Ergebnisse der $E^{(r)}$-Funktion für X_2 und X_4.
   (e) Entferne den angenommenen Teilschlüssel $K_1^{'}$, wenn die Ergebnisse unterschiedlich sind.
2. Fertige eine Liste aller Schlüssel an, deren Ergebnis bei (d) gleich war.

Diese Attacken lassen sich auch auf die anderen Schlüsselgrößen anwenden. Ein 192-Bit Schlüssel kann gefunden werden, indem man 2^{128} Klartexte wählt und 2^{128} Phasen der Analyse durchführt, bzw. 2^{33} bekannte Klartexte verwendet und 2^{161} Stufen der Analyse. Einen 256-Bit Schlüssel findet man, indem man 2^{128} Klartexte wählt und 2^{192} Phasen der Analyse durchführt, bzw. 2^{33} bekannte Klartexte verwendet und 2^{225} Stufen der Analyse.

4 Performanz

4.1 Performanz in Abhängigkeit der Schlüssellänge

Unter den gesamten AES Kandidaten gibt es vier Algorithmen, darunter MAGENTA, die mit unterschiedlicher Geschwindigkeit verschlüsseln bzw. entschlüsseln in Abhängigkeit der Schlüssellänge.
Weiterhin sei angemerkt, daß MAGENTA etwa 33% langsamer verschlüsselt, wenn ein 256-Bit Schlüssel anstatt eines 128-Bit Schlüssels verwendet wird.
Dieses wird anhand der folgenden Tabellen nochmal im Vergleich zu den anderen Kandidaten dargestellt. Geschwindigkeit der AES Kandidaten für unterschiedliche Schlüssellängen:

Algorithmus	Schlüsselgenerierung	Verschlüsselung
CAST-256	konstant	konstant
Crypton	konstant	konstant
DEAL	wachsend	128,192: 6 Runden 256: 8 Runden
DFC	konstant	konstant
E2	konstant	konstant
Frog	konstant	konstant
HPC	konstant	konstant
Loki97	fallend	konstant
MAGENTA	**wachsend**	**128,192: 6 Runden** **256: 8 Runden**
Mars	konstant	konstant
RC6	konstant	konstant
Rijndel	wachsend	128: 10 Runden 192: 12 Runden 256: 14 Runden
SAFER+	wachsend	128: 8 Runden 192: 12 Runden 256: 16 Runden
Serpent	konstant	konstant
Twofish	wachsend	konstant

Performanz der Kandidaten mit einem 128-Bit Schlüssel auf einer Pentium Pro CPU (es wird jeweils die Zeit in Takten für die Schlüsselgenerierung bzw. Verschlüsselung angegeben):

Algorithmus	Schlüsselgenerierung	Verschlüsselung
CAST-256	4300	660
Crypton	955	476
DEAL	4000	2600
DFC	7200	1700
E2	2100	720
Frog	1386000	2600
HPC	120000	1600
Loki97	7500	2150
MAGENTA	**50**	**6600**
Mars	4400	390
RC6	1700	260
Rijndel	850	440
SAFER+	4000	1400
Serpent	2500	1030
Twofish	8600	400

4.2 32-Bit Performanz

MAGENTA ist hier bei weitem der langsamste Algorithmus. Das beruht darauf, daß der Kern des Algorithmus eine byteweise Feistelstruktur benutzt und deswegen jegliche 32-Bit Implementation auf 8-Bit manipuliert wird, was natürlich sehr ineffizient ist. MAGENTA besitzt ausserdem eine hohe Anzahl von Operationen pro Runde, so daß dadurch die Geschwindigkeit weiter beeinträchtigt wird.

5 Quellen

1. Der MAGENTA Algorithmus; Klaus Huber; Deutsche Telekom; 8.5.95
2. Cryptanalysis of MAGENTA; Eli Biham, Niels Ferguson, Lars R. Knudsen, Bruce Schneier, Adi Shamir; 20.8.1998
3. Performance Comparison of the AES Submissions; Version 2.0; 1.2.1999
4. Conference Report (2. AES Conference); Morris Dworkin; 22./23.3.1999