Der AES Kandidat MARS

-

Seminar Analyse kryptographischer Systeme

Martin Stiel[1]

Humboldt Universität zu Berlin - Institut für Informatik
August 2002

[1]stiel@informatik.hu-berlin.de

Zusammenfassung

Im Zuge der Entwicklung eines Nachfolgers, genannt *Advanced Encryption Standard* (AES), für den *Default Encryption Standard* (DES) des *National Institute of Standards* (NIST) der USA haben sich eine Reihe von Neu- und Weiterentwicklungen auf dem Gebiet kryptographischer Algorithmen ergeben. Diese Arbeit soll den bis in die finale Runde des AES Auswahlverfahrens gelangten Algorithmus MARS, eingereicht von der Firma IBM, vorstellen. Der Algorithmus beruht auf einem verhältnismäßig konservativem Design um Sicherheitslücken durch weitestgehend ungetestete neue Methoden zu begegnen. Im Vergleich zum neuen AES, dem Sieger des Verfahrens, Rjindael ist vor allem die schlechtere Performanz in den Abschlußberichten hervorgehoben worden. MARS ist ein symmetrisches Blockchiffre mit S-Boxen und einem Typ 3 Feistelnetzwerk.

Inhaltsverzeichnis

1 Einführung

In der Zusammenfassung wurden bereits einige wichtige Eigenschaften von MARS erwähnt. Es handelt sich um ein symmetrisches Blockchiffre mit einer Blockgröße von 128 Bit und einer variablen Schlüssellänge von 128 Bit - 448 Bit. MARS verwendet im Verlaufe der Verschlüsselung S-Boxen und ein Feistelnetzwerk des Typs 3. Der Algorithmus ist Smart-Card fähig, d.h. es wurde bei der Entwicklung auf eine speicherschonenden Ablauf und einfache Operationen geachtet. Das unter anderem durch S-Boxen (siehe auch DES und CAST) sowie Feistelnetzwerk (siehe auch DES, RC5, CAST und Blowfish) verdeutlichte konservative Design erlaubt es lt. [4] MARS hinsichtlich der Sicherheitsreserven besser als den mit einer Reihe unkonventioneller Designentscheidungen erstellten Sieger Rjindael einzustufen. Schematische Darstellungen der im folgenden vorgestellten Bestandteile des Algorithmus können den Vortragsfolien entnommen werden.

2 Design

Der 128 Bit Eingabeblock (Klartext) wird in vier Teilblöcke a 32 Bit aufgeteilt. Diese durchlaufen nacheinander die drei im folgenden näher beschriebenen Phasen des Algorithmus:

1. *Forward Mixing* (8 Runden)
2. *Cryptographic Core* (16 Runden)
3. *Backward Mixing* (8 Runden).

Im gesamten Algorithmus werden die Addition, Substraktion, Multiplikation, bitweises Exklusiv-Oder ($\wedge$) und Rotation ($\lll$ respektive $\ggg$) verwendet. Letztere sowohl um feste als auch durch variable Argumente gegebene Werte. Die aus 512 (32 Bit -) Wörtern bestehende S - Box wird vor allem zur Abwehr linearer und differentieller Kryptoanalyse, sowie zur Erzeugung eines Lawineneffekts[1] eingesetzt. Die Entschlüsselung ist wie bei symmetrischen Verfahren üblich invers.

2.1 Schlüsselexpansion

In der Einführung wurde bereits erwähnt, daß die Schlüssellänge in Längen von 4 bis 40 Wörtern variiert. Da im *Cryptographic Core* mit einer konstanten Schlüssellänge gearbeitet wird findet die Erweiterung des durch den den Nutzer gegebenen Schlüssels auf einen Array aus 40 Wörtern statt. Ein Exploit des Algorithmus

[1] Die Beeinflussung aller Eingabewörter und völlige Veränderung bereits nach wenigen Runden.

der Schlüsselexpansion kann [1] entnommen werden. Er erhält als Eingabe den Nutzerschlüssel und gibt nach den drei Phasen

1. lineare Schlüsselerweiterung
2. Permutation mittels S-Box
3. Modifikation der Multiplikationsschlüssel

ein aus 40 Wörtern bestehenden Schlüsselarray aus. Das eigentliche 'Aufblähen' des Schlüssels erfolgt bereits in der ersten Phase mittels Exklusiv-Oder und Rotation. Die letzte Phase behandelt die in der Rundenfunktion des *Cryptographic Core* zur Multiplikation verwendeten Schlüssel nochmal extra auf Grundlage einer Art 'kleinen S - Box'.

2.2 Forward und Backward Mixing

Forward und *Backward Mixing* sind schlüsselunabhägige Permutationen der Eingabewörter unter Verwendung der S-Box. Der folgende Pseudocode Exploit ([1]) veranschaulicht die jeweils acht Runden umfassenden Permutationen:

```
// forward mixing
for (i = 0; i<8; i++) {
  b = (b ^ s0[a]) + s1[a >> 8]
  c = c + s0[a >> 16]
  d = d ^ s1[a >> 24]
  a = (a >> 24)
  if ((i == 1) || (i == 5)) a += b
  if ((i == 0) || (i == 4)) a += d
  (a,b,c,d) = (b,c,d,a)
}

// backward mixing
for (i = 0; i<8; i++) {
  if ((i == 3) || (i == 7)) a -= b
  if ((i == 2) || (i == 6)) a -= d
  b = b ^ s1[a]
  c = c - s0[a << 8]
  d = (d - s1[a << 16]) xo s0[a << 24]
  (a,b,c,d) = (b,c,d,a << 24)
}
```

Hierbei ist `(a,b,c,d)` der aus vier Wörtern bestehende 128 Bit Eingabeblock und `s0` respektive `s1` sind S - Box Zugriffe. Bei beiden werden jeweils nur die unteren 8 Bit zur Indizierung verwendet.

3 Algorithmus - Cryptographic Core

Die eigentliche schlüsselbassierte Ent- und Verschlüsselung findet in den 16 Runden der Phase 2, genannt *Cryptographic Core*, statt. Es werden zwei Modi (*Forward* und *Backward Mode*) unterschieden um die Sicherheit bei Ver- und Entschlüsselung entsprechend der Symmetrie zu gewährleisten. In beiden Modi wird innerhalb der Rundenfunktion eine Funktion namens E eingesetzt, die ein Datenwort sowie zwei Schlüsselwörter als Eingabe erhält und eine Ausgabe von drei Datenwörtern damit erzeugt. Nach dem Durchlauf durch die Operationen einer Runde werden die vier Eingabewörter entsprechend einem Typ 3 Feistelnetzwerk vertauscht. Die nicht von der *E - Funktion* verwendeten Schlüssel werden am Beginn (vor allen Runden) auf die Eingabewörter addiert respektive am Ende (nach allen Runden) subtrahiert. Nachfolgend ein weiteres Pseudocode Exploit aus [1] zur Veranschaulichung:

```
// cryptographic core
for (i = 0; i<16; i++) {
  // e - function
  r = ((a << 13) * k[2*i+5]) << 10
  m = (a + k[2*i+4]) << (low 5 bits of: r >> 5)
  l = (s[m] ^(r >> 5) ^ r) << (low 5 bits of: r)
  // forward mode
  if (i < 8) {
    b += l
    d = d ^ R
  }
  // backward mode
  if (i >= 8) {
    b = b ^ r
    d += l
  }
  c = c + m
  (a,b,c,d) = (b,c,d,a << 13)
}
```

Wiederum ist `(a,b,c,d)` der aus vier Wörtern bestehende Eingabeblock. Der Ausdruck `s[m]` bezeichnet einen S-Box Zugriff mit den 9 unteren Bits von `m` als

Index. Die Variablen `r`, `l` und `m` entsprechen den Ausgängen der *E - Funktion* und `k` ist der Schlüsselarray.

3.1 Forward Mode

Im *Forward Mode* - den ersten acht Runden - wird das erste Eingabewort a als Eingabe der *E - Funktion* verwendet. Das erste Ausgabewort dieser Funktion wird auf das zweite Eingabewort b addiert, ebenso das zweite Ausgabewort auf das dritte Eingabewort c. Das letzte Ausgabewort der *E - Funktion* und das vierte Eingabewort d werden durch ein bitweises Exklusiv-Oder verknüpft. Zusätzlich werden von der *E - Funktion* die Schlüsselwörter *k[2i+4]* und *k[2i+5]* (i ist die Nummer der Runde) verwendet. Nach dem Anwenden der *E - Funktion* werden die Eingabewörter noch vertauscht um die endgültigen Rundenausgabewörter zu erzeugen: das zuvor um 13 Bit rotierte a wird zu d, b wird zu a, c wird zu b und d wird zu c.

3.2 Backward Mode

Analog zum Forward Mode ergibt sich der Backward Mode - die letzten acht Runden: Die Eingabewörter der *E - Funktion* und die Vertauschung am Ende einer jeden Runde bleiben gleich. Es wird lediglich das erste Ausgabewort der *E - Funktion* auf das letzte Eingabewort d addiert, das zweite Ausgabewort auf das dritte Eingabewort c und das letzte Ausgabewort wird mit einem bitweisen Exklusiv-Oder mit dem zweiten Eingabewort b verknüpft.

3.3 E - Funktion

Wie bereits erwähnt wird die *E - Funktion* in jeder Runde der eigentlichen Ver- und Entschlüsselung verwendet, stellt also das zentrale Element des eigentlichen Algorithmus dar. Das Eingabewort der Funktion wird auf 'drei verschiedenen Wegen' benutzt. Die erste Ausgabe bestimmt sich aus der Eingabe durch Rotation um 13 Bits nach links, anschließende Multiplikation mit den ebenfalls eingehenden ungeraden[2] Schlüsselwort und zwei weiteren Rotationen nach links um jeweils 5 Bits. Das zweite Ausgabewort ergibt sich aus der Eingabe durch Addition des geraden eingehenden Schlüsselwortes und anschließender Rotation nach links um einen Wert, der sich aus den 5 niedrigsten Bits nach der ersten Rotation bei der Erzeugung des ersten Ausgabewortes ergibt. Auch auf das dritte Ausgabewort wird anfangs das gerade Schlüsselwort addiert, anschließend werden dessen unteren 9 Bits als Index in der verwendeten S - Box betrachtet und der entsprechende

[2] Ungerade meint den Index im Schlüsselarray.

Eintrag wird mit den bei der Erzeugung des ersten Eingabewortes nach der ersten und zweiten Rotation entstehenden Werten jeweils durch ein Exklusiv-Oder verknüpft. Zuletzt wird das dritte Ausgabewort wieder um die 5 niedrigsten Bits des ersten Ausgabewortes nach links rotiert.

Der Wert (13) der ersten festen Rotation soll lt. [1] einen höheren Widerstand gegenüber differentiellen Attacken ermöglichen. Die Verwendung der jeweiligen Bits für die *data dependant rotation* begründet die gleiche Quelle mit der größten Beeinflussung dieser Bits durch den ungeraden Schlüssel nach der Multiplikation und den entsprechenden Rotationen. Hinsichtlich der Implementation wäre noch anzumerken, daß Multiplikation und *S - Box Lookup* in der *E - Funktion* mit entsprechenden Performanzvorteilen auf geeigneten Architekturen parallel abgearbeitet werden können.

4 Performanz

4.1 Software

Die im folgenden angegebenen Werte für MARS beziehen sich auf eine C - Implementation. Ver- und Entschlüsselung benötigen entsprechend einem symmetrischen Verfahren die gleiche Zeit. Bei der Schlüsselerweiterung spielt die Größe des vorgegebenen (Nutzer-)schlüssels (siehe Abschnitt Schlüsselexpansion) eine Rolle.

Performanz in Software

	Power PC 200 MHz	Pentium Pro 200 MHz
DES (MBit/s)	-	7.3
Triple DES (MBit/s)	-	16.7
MARS - gesamt (MBit/s)	85	65
MARS - Core (MBit/s)	160	104
Schlüsselexpansion (10^3 keys/s)	121	52

Quelle: [1]

Wie erwartet ist MARS deutlich performanter als DES und Triple DES. Der Vergleich mit den anderen AES Kandidaten erfolgt unten.

4.2 Hardware

Die Angabe der Komplexität bezieht sich auf die Anzahl der notwendigen Zellen in den Schaltkreisen - zum Vergleich: DES: ca. $28 \cdot 10^3$ Zellen. In der Spalte *Pipelining* sind parallel verarbeitete Kopien der einzelnen Phasen (*Forward*, *Backward*

Mixing und Cryptographic Core) gemeint. Diese Form der Abarbeitung ist nur im ECB und CBC Modus möglich.

Performanz in Hardware

Komplexität (10^3 Zellen)	Pipelining (Kopien)	Durchsatz (MBit/s)
70	2	640
70	4 + 4	4000
393	4 + 8	8000

Quelle: [1]

Aus der Tabelle ist ersichtlich, daß zum einen der Datendurchsatz bei Hardwarelösungen wie erwartet deutlich höher gegenüber der Implementation in Software ist und zum anderen die Erhöhung der Komplexität der Parallelverarbeitung mittels *Pipelining* ab 12 Kopien nicht mehr proportional zur Steigerungsrate des Durchsatzes ist.

4.3 AES Vergleich

Eingangs wurde bereits erwähnt, daß der AES Sieger Rjindael unter anderem wegen der besseren Performanz präferiert wurde. Die folgende Tabelle zeigt die Performanzwerte einiger AES Kandidaten im Vergleich.

MARS im Vergleich zu den anderen Kandidaten

	Key Setup (Takte)	Verschlüsselung (Takte)	SC RAM (Bytes)
MARS	4400	390	195
Serpent	2500	1030	50
Rjindael	850	440	52
RC6	1700	260	210
Twofish	8600	400	60

Quelle: [3]

Man kann aus der Tabelle ersehen, daß MARS hinsichtlich der Performanz der eigentlich Verschlüsselung zu den besten Algorithmen gehört, was sicherlich auf die durchweg verwendeten einfachen Grundoperationen zurückzuführen ist[3]. Eher im hinteren Bereich sind hingegen die Werte für *Key Setup* und *Smart Card RAM*. Ersterer hat jedoch aufgrund der Möglichkeit des *Key Caching*[4] als Wert eine geringere Bedeutung als die für die eigentliche Verschlüsselung benötigten Takte. Zusammenfassend sprechen [3] und [5] von einer 'durchschnittlichen' Performanz

[3]Einschränkend muß gesagt werden, daß [3] beim Wert für MARS von einem speziell optimierten Compiler (und entsprechendem C - Code) spricht. Dies sollte an der eigentlichen Aussage jedoch nichts ändern.

[4]D.h. das *Key Setup* muß für jeden Nutzerschlüssel nur einmal ausgeführt werden.

des Algorithmus. Eine Stellungnahme von IBM [4] zu den Kritikpunkten sieht hingegen MARS mit guter Performanz als Rjindael durchaus ebendbürtig an.

5 Sicherheit

Der bei jedem Algorithmus denkbare *Brute Force* Angriff[5] benötigt entsprechend der Schlüsselraumgröße 2^n Schritte, wobei n die maximale Länge der verwendbaren Schlüssel ist. Untersuchungen von IBM und der AES Jury ([1, 5]) haben keine Möglichkeit zur differentiellen oder linearen Kryptoanalyse für MARS mit 128 Bit Blöcken ergeben.

Die nachfolgenden Angaben zur Sicherheit von MARS gegnüber differentieller und linearer Kryptoanalyse beziehen sich auf einen auf 16 Runden (nur *Cryptographic Core*) reduzierten Algorithmus. Es gibt keinen linearen Angriff der mit eine Wahrscheinlichkeit größer 2^{-69} Abhängigkeiten zwischen den Daten entdeckt. Für differentielle Angriffe beträgt diese Wahrscheinlichkeit sogar 2^{-156}. [1]

Erprobte Angriffe im Rahmen von AES Analysen [5] waren erst ab 5 von 16 *Core Rounds* respektive 21 von 32 Gesamtrunden erfolgreich und bescheinigten dem Algorithmus keine verwertbaren Lücken sowie einen hohen Sicherheitsabstand.

[5] D.h. das Durchprobieren aller Schlüssel.

Literatur

[1] *The MARS Encryption Algorithm.*
URL: http://www.research.ibm.com/security/mars.pdf.

[2] Reinhard Wobst. *Abenteuer Kryptologie.*
Addison Wesley, 1998.

[3] Schneier, Kelsey, Whiting, Wagner und Hall. *Performance Comparison of the AES Submissions.*
URL: http://csrc.nist.gov/encryption/aes/round2/perfomance.pdf.

[4] *MARS and the AES Selection Criteria.*
URL: http://www.research.ibm.com/security/final-comments.pdf.

[5] Nechvatal, Barker, Bassham, Burr, Dworkin, Foti, Roback. *Report on the Development of the Advanced Encryption Standard.*
URL: http://csrc.nist.gov/encryption/aes/round2/r2report.pdf.

[6] *Third AES Candidate Conference - Proceedings.*
URL: http://csrc.nist.gov/encryption/aes/round2/AES3proceedings.pdf.