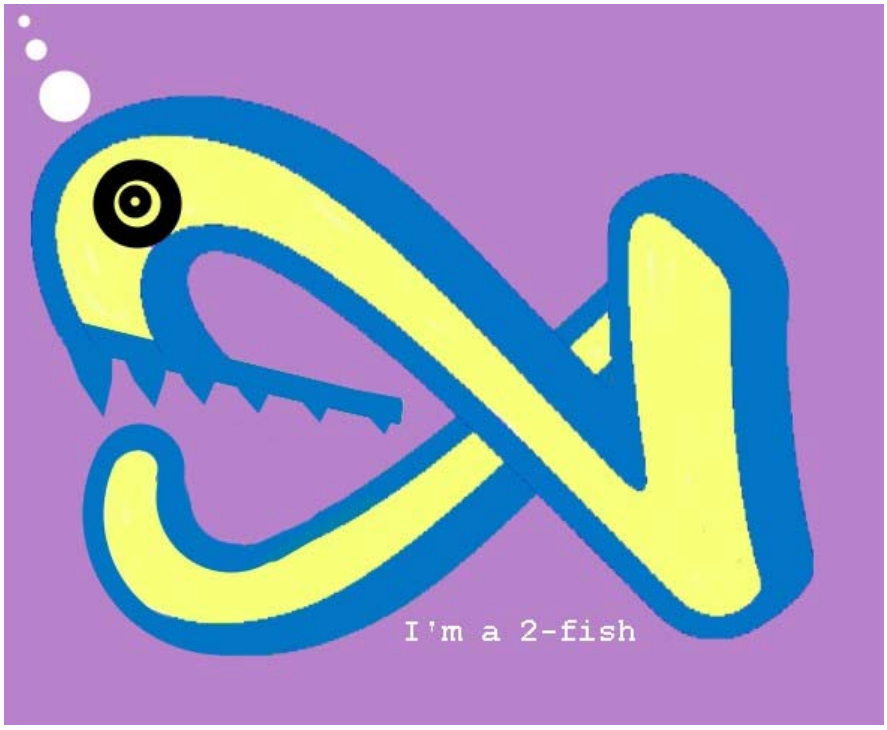
Twofish

a 128-Bit Block cipher



zusammengefasst von:

G. Rohner, M. Aregger, S. Schmid, R. Meier, J. Weber

Inhaltsverzeichnis

Inhaltsverzeichnis **2**

1. TwoFish abstract **3**

1.1. Entwicklung 3

1.2. Block- und Schlüssellänge 3

1.3. Typ und Anwendung 3

2. Key Scheduling Algorithmus **3**

2.1. Generierung von 40 Teilschlüsseln & Definition der S-Boxen 3

2.2. Funktionsweise Verschlüsselung 3

2.3. Funktionsweise Entschlüsselung 4

3. Geschwindigkeit in Hard- und Software **5**

3.1. Geschwindigkeit in Hardware 5

3.2. Synthetische Benchmarks auf Pentium Architektur 5

3.3. Benchmarks in Software 5

4. Schwache Schlüssel in Twofish **5**

5. Durchgeführte und mögliche Angriffe **5**

5.1. Durchgeführte Angriffe 5

5.2. Mögliche Angriffe 6

6. Kryptografische Stärken **6**

7. Links zum Thema **6**

8. Quellenverzeichnis **6**

9. Anhang A; Tabellenverzeichnis **7**

10. Anhang B; Abbildungsverzeichnis **7**

11. Anhang 1; Blockschema Twofish **8**

12. Anhang 2; S-Boxen **9**

13. Anhang 3; Berechnung der Rundenschlüssel **10**

1. TwoFish abstract

1.1. Entwicklung

Twofish ist ein Block-Cipher der Firma Counterpane Internet Security und war 1997 einer der fünf „Advanced Encryption Standard“ (AES) Finalisten (andere MARS, RC6, Rijndael und Serpent). Twofish ist nicht patentiert und der Source-Code kann lizenzfrei und in zahlreichen Implementierungen (Delphi, Java, C, Perl, VB, etc.) bezogen werden. Als Väter dieses Algorithmus seien Bruce Schneider, John Kesley, Doug Whiting, David Wagner, Chris Hall und Niels Ferguson genannt.

1.2. Block- und Schlüssellänge

Um der Kontroverse über die offensichtlich kurze Schlüssellänge von DES (56 Bit) Einhalt zu gebieten, lancierte das National Institute of Standard and Technology (NIST) 1997 das Advanced Encryption Standard-Program. Anforderungen an den neuen Algorithmus waren u.a. ein längerer Schlüssel und grössere Blocksize. Twofisch erfüllte diese Designkriterien mit:

- **Blockgrösse: 128 Bit**
- **Schlüssellänge: 128-, 192- und 256 Bit (variabel)**

1.3. Typ und Anwendung

Der Cipher ist ein 16-Runden Feistel-Netzwerk, dass in allen Standard-Betriebsmoden arbeitet. Die Verfügbarkeit und charakteristischen Eigenschaften des Algorithmus führen zu einer breiten Anwendungspalette (siehe http://www.schneier.com/twofish-products.html).

2. Key Scheduling Algorithmus

Dieses Kapitel beschreibt die einzelnen Schritte des Twofish Algorithmus (siehe auch Anhang 1; Blockschema Twofish).

2.1. Generierung von 40 Teilschlüsseln & Definition der S-Boxen

Der Twofish Algorithmus benötigt für die Funktion F 40 Teilschlüssel. Diese werden aus dem geheimen Schlüssel generiert. Ebenfalls wird der Schlüssel zur Definition der S-Boxen verwendet. Diese sind also schlüsselabhängig aber nicht zufällig.

Die Generierung der S-Boxen wird in Anhang 2; S-Boxen genauer beschrieben.

Die Generierung der Teilschlüssel wird in Anhang 3; Berechnung der Rundenschlüssel genauer beschrieben.

2.2. Funktionsweise Verschlüsselung

- Input in 4 x 32 bit Worte aufteilen
- XOR-Verknüpfung mit den 32 bit Teilschlüsseln
- $R_{r,1}$ wird um 8 Positionen nach links rotiert
- $R_{r,0}$ und $R_{r,1}$ werden als Input für die Funktion F verwendet
- Die 32 bit Worte werden in 4 Bytes unterteilt und in den S-Boxen Substituiert. Die S-Boxen sind schlüsselabhängig und arbeiten auf 8-Bit-Blöcken. Siehe Anhang 2; S-Boxen.
- Das Resultat der Substitution wird zu einem Vektor zusammengefasst und mit der Matrix MDS multipliziert. Die Ausgabe wird T_0 und T_1 genannt. Die MDS-Matrix ist genauer beschrieben im Anhang 2; S-Boxen.
- PHT (Pseudo-Hadamart-Transformation): T_0 wird zu T_1 modulo 2^{32} addiert. Das Ergebnis wird dann zu T_1 modulo 2^{32} addiert

- Das Ergebnis wird mit den Teilschlüsseln K_{2r+8} bzw. K_{2r+9} modulo 2^{32} addiert. Das Ergebnis wird $F_{r,0}$ bzw. $F_{r,1}$ genannt
- $F_{r,0}$ und $R_{r,2}$ werden XOR verknüpft und um eine Position nach rechts rotiert. $R_{r,3}$ wird um eine Position nach links rotiert und mit $F_{r,1}$ XOR verknüpft
- Die Ergebnisse werden zu $R_{r+1,0}$ bzw. $R_{r+1,1}$. $R_{r,0}$ und $R_{r,1}$ werden zu $R_{r+1,2}$ und $R_{r+1,3}$
- Dies wird 16 mal ausgeführt
- Am Schluss wird die letzte Vertauschung rückgängig gemacht und die Worte $R_{16,0}$ bis $R_{16,3}$ mit den Teilschlüsseln K_4 bis K_7 XOR verknüpft

2.3. Funktionsweise Entschlüsselung

Zur Entschlüsselung wird der selbe Algorithmus leicht modifiziert (siehe Abbildung 3). Die Teilschlüssel werden in umgekehrter Reihenfolge benutzt.

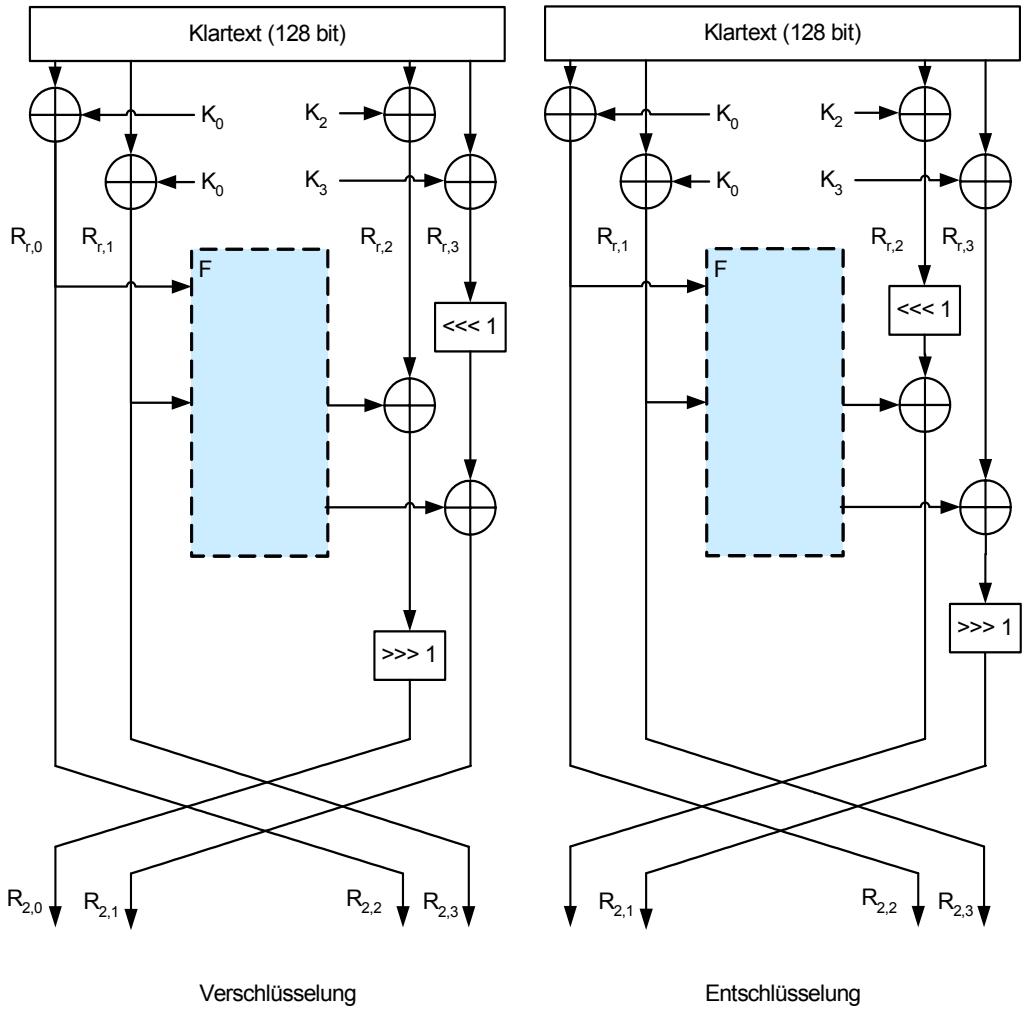


Abbildung 1: Funktionsweise Ver- und Entschlüsselung

3. Geschwindigkeit in Hard- und Software

3.1. Geschwindigkeit in Hardware

Eine genauere Analyse findet sich auf der offiziellen Homepage: http://www.schneier.com/twofish-hardware.html

3.2. Synthetische Benchmarks auf Pentium Architektur

Algorithmus	Schlüssellänge	Länge (bits)	Runden	Durchläufe	Takte/Byte
Twofish	variabel	128	16	8	18.1
Blowfish	variable	64	16	8	19.8
DES	56	64	16	8	43

Tabelle 1: Performance auf Pentium Architektur

3.3. Benchmarks in Software

System	Compiler	Bytes/s
Pentium 90, MS-DOS 6.22	bcc –O2	~ 494’361
PII-233 , Linux	gcc -O2 -ansi	~ 7’818’226

Tabelle 2: Benchmarks mit konkreten Implementierungen

4. Schwache Schlüssel in Twofish

Zur Zeit sind keine schwachen oder semischwachen Schlüssel bekannt. Dies zeigt sich auch dadurch, dass keine effizienteren Angriffe als Brute-Force Attacken möglich sind (vgl. 5.2).

5. Durchgeführte und mögliche Angriffe

Bis jetzt konnte man nur auf abgeschwächten Twofish Verschlüsselungen eine erfolgreiche Attacke durchführen. Hierzu zwei Beispiele:

5.1. Durchgeführte Angriffe

5 (von 16) Runden des Feistel-Netzwerkes mit $2^{22,5}$ = 6 Mio. bekannten Eingaben bei einem Aufwand von 2^{51} (= $2{,}3 * 10^{15}$). Bedarf an Rechenzeit steigt exponentiell mit jedem weiteren Feistel-Netzwerk!

Weiterhin wurde ein Related Key Angriff auf eine 10-Runden-Version von TWOFISH ausgeführt. Es darf kein Pre- und Postwhitening durchgeführt werden. Der Angreifer muss sich 232 Quelltexte aussuchen dürfen sowie später 211 Quelltexte adaptiv aussuchen dürfen. Außerdem muss er die Kontrolle über 20 ausgesuchte Bytes des Schlüssels haben.

Wie wir an hand dieser zwei Beispielen sehen, wurde die Twofish Verschlüsselung sehr eingeschränkt, damit eine Entschlüsselung überhaupt durchgeführt werden konnte.

5.2. Mögliche Angriffe

Twofish wurde bisher sehr intensiv geprüft, ohne dass Schwachstellen gefunden worden wären.

Zur Zeit ist man der Meinung, dass es keine effektive Angriffsmöglichkeit auf den Twofish Algorithmus gibt, ausser eine Brute Force Attacke auf den Schlüssel. Ob dies bei einer Schlüssellänge zwischen 128 und 256 Bit jemals zum Ziel führt sei an dieser Stelle in Frage gestellt (Mathematische Sicherheit/Aufwand zum Knacken bei 256 Bit Schlüssellänge ca. 10^{10} Jahre).

6. Kryptografische Stärken

Durch die Verwendung von relativ kleinen 8x8 S-Boxen ist der Algorithmus sehr Smartcard-freundlich. Twofish ist sehr schnell und flexibel an die verwendeten Ressourcen anpassbar. Das Design ist ausführlich dokumentiert und bietet Sicherheit gegen alle bekannten Angriffe. Konkret:

a) Unverträgliche Opretaionen:
   - Bitweises XOR
   - MDS- bzw. RS-Matritzen
   - Addition modulo 2^{32}, Pseudo-Hadamard-Transformationen
   - Nichtlineare Substitutionen

b) Chosen-Plaintext-Angriff auf einen Twofish mit 5 Runden (ohne Output-Whitening) erfordert $2^{22,5}$ ausgewählte Klartexte und 2^{51} Berechnungen der Funktion g

c) Chosen-Key-Angriff auf einen Twofish mit 10 Runden (ohne Whitening) erfordert 2^{32} ausgewählte Klartexte und 2^{32} Berechnungen von g

d) Widerstandsfähig gegen Related-Key-Attacken (Schneier)

7. Links zum Thema

Source Code:

http://www.schneier.com/twofish-download.html

Literatur:

http://www.schneier.com/paper-twofish-paper.pdf

Applet:

http://russ.hn.org/twofish/twofish.html

8. Quellenverzeichnis

Twofish Home, http://www.schneier.com/twofish.html

Der Algorithmus Twofish, http://www.repges.net/AES-Kandidaten/Twofish/twofish.htm

Twofish Wikipedia, http://de.wikipedia.org/wiki/Twofish

Seccommerce - Symmetrische Verfahren Symmetrische Verfahren

Bruce Schneier Wikipedia, http://de.wikipedia.org/wiki/Bruce_Schneier

AES Wikipedia, http://de.wikipedia.org/wiki/Advanced_Encryption_Standard

9. Anhang A; Tabellenverzeichnis

Tabelle 1: Performance auf Pentium Architektur....5
Tabelle 2: Benchmarks mit konkreten Implementierungen....5

10. Anhang B; Abbildungsverzeichnis

Abbildung 1: Funktionsweise Ver- und Entschlüsselung....4
Abbildung 3; Blockschema Twofish....8
Abbildung 4: S-Boxen....9
Abbildung 5: Die Funktion q....9
Abbildung 6: Matrix MDS....9
Abbildung 7: Erzeugung von S[i]....9
Abbildung 8: Rundenschlüssel-Berechnung....10

11. Anhang 1; Blockschema Twofish

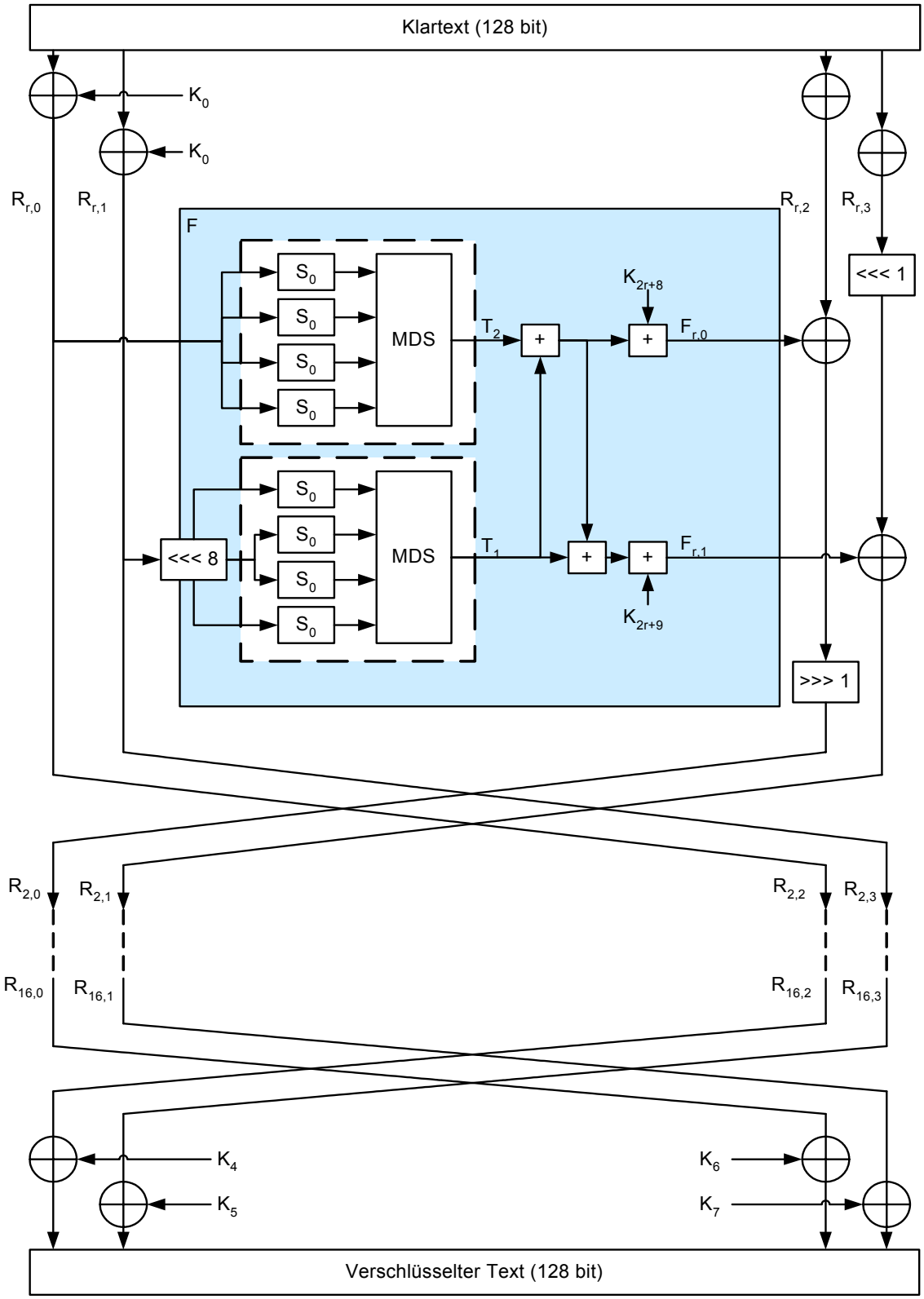


Abbildung 2; Blockschema Twofish

12. Anhang 2; S-Boxen

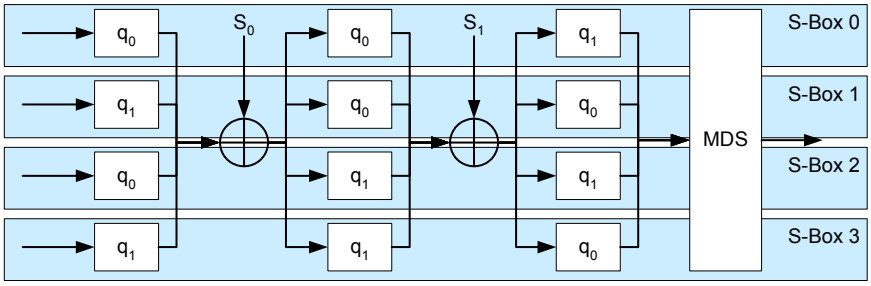
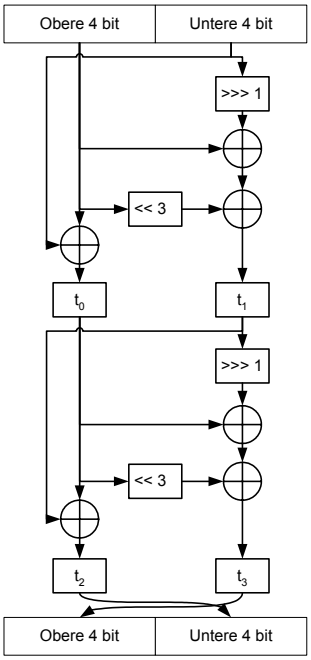


Abbildung 3: S-Boxen

Die dazu benötigten Funktionen sind:

Funktion	**Hinweise**
 Abbildung 4: Die Funktion q	Für q_0 gilt: $t_0 = [81 \quad 7D \quad 6F \quad 32 \quad 0B \quad 59 \quad EC \quad A4]$ $t_1 = [EC \quad B8 \quad 12 \quad 35 \quad F4 \quad A6 \quad 70 \quad 9D]$ $t_2 = [BA \quad 5E \quad 6D \quad 90 \quad C8 \quad F3 \quad 24 \quad 71]$ $t_3 = [D7 \quad F4 \quad 12 \quad 6E \quad 9B \quad 30 \quad 85 \quad CA]$ Für q_1 gilt: $t_0 = [28 \quad BD \quad F7 \quad 6E \quad 31 \quad 94 \quad 0A \quad C5]$ $t_1 = [1E \quad 2B \quad 4C \quad 37 \quad 6D \quad A5 \quad F9 \quad 08]$ $t_2 = [4C \quad 75 \quad 16 \quad 9A \quad 0E \quad D8 \quad 2B \quad 3F]$ $t_3 = [B9 \quad 51 \quad C3 \quad DE \quad 64 \quad 7F \quad 20 \quad 8A]$
$MDS = \begin{bmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{bmatrix}$ Abbildung 5: Matrix MDS	
$\begin{bmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{bmatrix} = \begin{bmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{bmatrix} \cdot \begin{bmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{bmatrix}$ *Abbildung 6: Erzeugung von S[i]*	$i = 0,...,k-1$ $S_i = \sum_{j=0}^{3} s_{i,j} \cdot 2^{8j}$

13. Anhang 3; Berechnung der Rundenschlüssel

Zur Berechnung der Rundenschlüssel werden leicht modifizierte S-Boxen benutzt. Hier der Einfachheit halber nur das Blockschema für einen 128bit Schlüssel.

Funktion

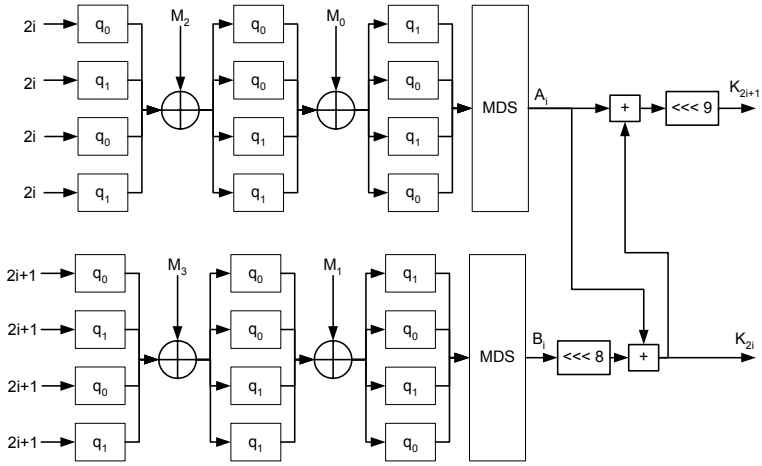


Abbildung 7: Rundenschlüssel-Berechnung

Hinweise

M: Schlüssel der Länge N. Besteht aus 8*k Bytes:

$M=(m_0, m_1,..., m_{8k-1})$

bzw. 2*k Wörtern der Länge 32 bit:

$M=(M_0, M_1, ..., M_{2k-1})$

N: Schlüssellänge in bit (128, 192, 256)

k: {2, 3, 4} so dass k=N/64

i: 0, ..., 19