Analyse Kryptographischer Algorithmen: KRYPTON & TWOFISH

Martin Lötzsch

loetzsch@informatik.hu-berlin.de

Einleitung. Das Seminar „Analyse Kryptographischer Algorithmen“ beschäftigte sich mit interessanten, aber weniger bekannten Verschlüsselungsmethoden. Unter anderem wurde ein großer Teil der Kanditaten für den *Advanced Encryption Standard (AES)* untersucht. Das Thema dieser Arbeit sollte eigentlich der Algorithmus KRYPTON sein. Aufgrund der zum Zeitpunkt des Referates schlechten Quellenlage wurde das Thema jedoch um den Algorithmus TWOFISH erweitert. Dieser stellt eine Optimierung und Kombination von anderen in diesem Seminar schon vorgestellten Algorithmen dar.

1 KRYPTON

KRYPTON wurde von Chae Hoon Lim, einem Angestellten von Future Systems, Inc. in Soul als AES Kandidat eingereicht. Leider ist der einzigen im Internet zu findenden Quelle [2] nicht sehr viel über diesen Algorithmus zu erfahren.
Es handelt sich bei KRYPTON um einen 128 Bit Block-Chiffre. KRYPTON ist lediglich eine leichte Modifikation von SQUARE, was auch ein Grund dafür sein könnte, dass er in der Literatur relativ gering vertreten ist.

2 TWOFISH

TWOFISH wurde von Bruce Schneider, John Kelsey, Doug Whiting, Chris Hall und Niels Ferguson bei der Firma Counterpane Systems in Minneapolis sowie an University of California in Berkeley entwickelt. Es war ein AES Kandidat, der die Endrunde erreicht hat.

Es handelt sich auch hierbei um einen 128 Bit Block Chiffre. Die Schlüssel können 128, 192 und 256 Bit lang sein. Seine Besonderheit ist, dass er aufgrund der Verwendung von ausschließlich sehr einfacher Operationen leicht in Software und Hardware realisieren lässt. Es gibt verschiedene Implementierungsmöglichkeiten hinsichtlich des Tradeoffs Speicher – Geschwindigkeit.

TWOFISH ist sehr konservativ. Er enthält ausgehend von den anderen in diesem Seminar vorgestellten Algorithmen keine wesentlichen neuen Ideen oder Designs. Vielmehr wurden vorhandene gut verstandene Verfahren aus anderen Algorithmen optimiert und kombiniert.

2.1 Algorithmus

Abb. 1 zeigt die generelle Funktionsweise von TWOFISH. Zunächst wird Eingabeblock in vier 32 Bit Worte X_0 .. X_3 unterteilt.

Beim *Input Whitening* werden die Wörter mit jeweils einem Schlüssel XOR-verknüpft. Dadurch sieht der potentielle Angreifer die Eingabe des eigentlichen Eingaben des Algorithmus nicht. Analog funktioniert das Output Whitening am Ende des Algorithmus.

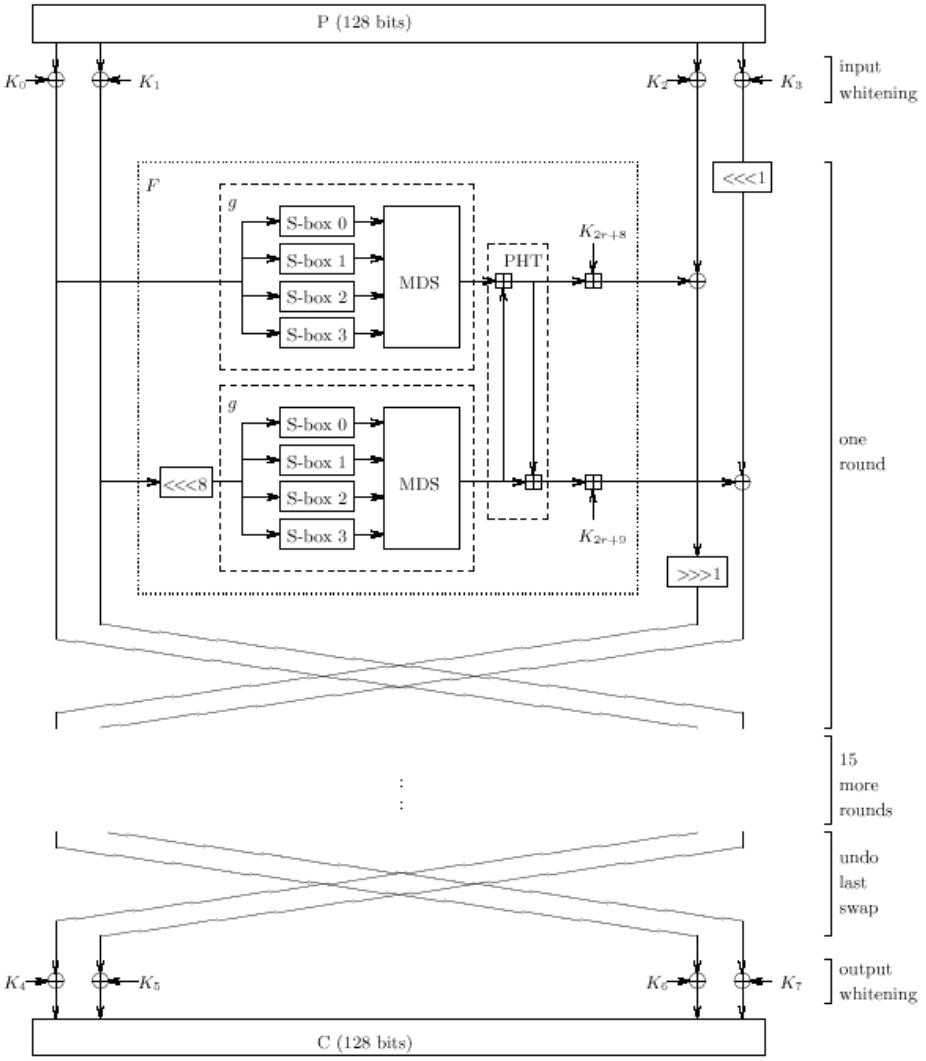

Abb. 1: Generelle Funktionsweise des TWOFISH Algorithmus.

Kernstück des Verfahrens sind 16 Wiederholungen einer „Runde“, deren Funktionsweise in Abb. 2 gezeigt wird. Nach der Rotation des ersten und vierten Eingabewortes wird die Funktion *g*, eine Kombination von vier verschiedenen S-Boxen und einer MDS-Matrix aus den ersten beiden Teilworten sowie den Schlüsseln K_{r+8} und K_{r+9}, angewandt.

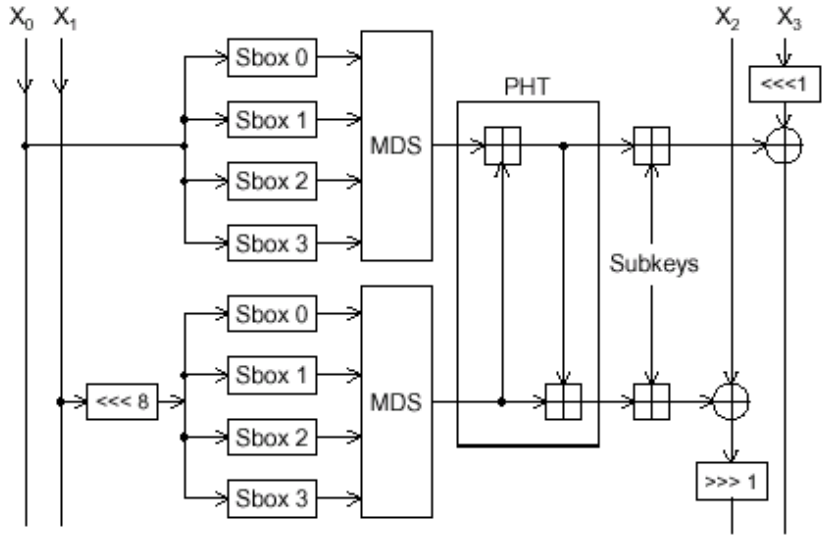

Abb. 2: Eine Runde des TWOFISH Algorithmus.

Eine S-Box ist eine bijektive Funktion, die in Abhängigkeit von 2 Teilschlüsseln ein 8-Bit-Eingabewort auf ein 8-Bit-Ausgabewort abbildet:

$s_0(x) = q_1[q_0[q_0[x] \wedge k_0] \wedge k_1]$
$s_1(x) = q_0[q_0[q_1[x] \wedge k_2] \wedge k_3]$
$s_2(x) = q_1[q_1[q_0[x] \wedge k_4] \wedge k_5]$
$s_3(x) = q_0[q_1[q_1[x] \wedge k_6] \wedge k_7]$

Dabei sind $k_0..k_7$ acht verschiedene Teilschlüssel, q_0 und q_1 sind zwei feste Permutationen.

q_0:
t0 = [8 1 7 D 6 F 3 2 0 B 5 9 E C A 4]
t1 = [E C B 8 1 2 3 5 F 4 A 6 7 0 9 D]
t2 = [B A 5 E 6 D 9 0 C 8 F 3 2 4 7 1]
t3 = [D 7 F 4 1 2 6 E 9 B 3 0 8 5 C A]

q_1:
t0 = [2 8 B D F 7 6 E 3 1 9 4 0 A C 5]
t1 = [1 E 2 B 4 C 3 7 6 D A 5 F 9 0 8]
t2 = [4 C 7 5 1 6 9 A 0 E D 8 2 B 3 F]
t3 = [B 9 5 1 C 3 D E 6 4 7 F 2 0 8 A]

Der Vorteil der Verwendung von Teilschlüsseln in den S-Boxen ist, dass der Angreifer so die Funktion der S-Boxen nicht kennt. Feste S-Boxen können hingegen leicht studiert werden. Ausserdem dient dieser Mechanismus Schutz vor bisher unbekannten Angriffen. Nachteil ist, dass die S-Boxen für jeden neuen Schlüssel neu berechnet werden müssen.

Anschließend werden die Ergebnisse der 4 S-Boxen mit der MDS-Matrix multipliziert. Die Ergebnisse werden anschließend wieder zu einem 32-Bit-Wort zusammengefügt. Die MDS-Matrix hat die Maximum Distance Seperable (MDS) Eigenschaft, d.h. es gibt mindestens 5 Bytes, die in der Eingabe und der Ausgabe ungleich von *0* sind. Die in TWOFISH verwendete Matrix ist:

$$\mathrm{MDS} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix}$$

Die Ergebnisse der beiden MDS-Matrizes werden dann mit Hilfe der Pseudo Hadamart Transformation kombiniert:

$a' = a + b \bmod 2^{32}$
$b' = a + 2b \bmod 2^{32}$

Dies ist ein sehr häufig verwendeter Algorithmus zum Mischen zweier Wörter. Der Vorteil dieser Methode ist, dass es sehr schnell ist.

Abschließend werden die beiden Ergebnisse mit den Schlüsseln K_{2r+8} und K_{2r+9} addiert und dann jeweils mit den letzten beiden Teilwörtern XOR-verknüpft. Das 3. Teilwort wird dann noch einmal rotiert.

2.2 Performance

Die Implementierung von TWOFISH ist sehr schnell sowohl in Software auf verschiedenen Plattformen als auch vielen verschiedenen Hardwarearchitekturen. Dabei lassen sich bei der Implementierung sehr gut Tradeoffs zwischen Verschlüsselungs- und Entschlüsselungszeit, Schlüsselerstellung, Speicherverbrauch und der Anzahl benötigter Schaltungsgatter einstellen.

Unter anderem gibt es vier verschiedene Möglichkeiten der Schlüsselerstellung:

- *Full Keying*: Vollständiges Vorberechnen der Schlüssel. Die dabei entstehende Lookup Table realisiert sowohl die S-Box Berechnung als auch die Multiplikation mit der MDS-Matrix.
- *Partial Keying*: Für die S-Boxen werden Lookup Tables berechnet. Vier weitere Tabellen werden für die MDS-Matrix-Multiplikation berechnet.
- *Minimal Keying*: Nur ein Teil der Q-Boxen wird vorberechnet.
- *Zero Keying*: Keine Vorberechnung der S-Boxes.

Abb. 3 veranschaulicht den Tradeoff zwischen Speichernutzung, Schlüsselgenerierung und Verschlüsselungszeit bei den vier vorgestellten Varianten.

Keying Option	RAM bytes	Key Setup Clocks			Clocks to Encrypt
		128-bit	192-bit	256-bit	
Full	4500	8000	11200	15700	600
Partial	1400	7100	9700	14100	800
Minimal	1400	3000	7800	12200	1130
Zero	200	2450	3200	4000	1750*

Abb. 3: Performance von verschiedenen TWOFISH Implementierungen [1].

2.3 Kryptoanalyse

Die Autoren haben zwei mögliche Angriffe auf abgeschwächte Versionen von TWOFISH beschrieben.

Mit differenzieller Kryptoanalyse ist ein Angriff auf eine 5 Runden- Version von TWOFISH möglich, wenn das Pre- und Postwhitening nicht durchgeführt wird und der Angreifer $2^{22.5}$ Quelltexte aussuchen kann. Da der Angriff 2^{51} Durchgänge erfordert, haben die Autoren den Angriff nicht vollständig durchgeführt.

Weiterhin wurde ein Related Key Angriff auf eine 10-Runden- Version von TWOFISH ausgeführt. Wieder 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.

Literatur

[1] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson (Counterpane Labs). *Twofish: A New Block Cipher. 1998.* http://www.counterpane.com/twofish.html

[2] Chae Hoon Lim, Future Systems, Inc., Soul (Südkorea). *CRYPTON: A new 128-Bit Block Cipher.* 1998. http://csrc.nist.gov/encryption/aes/round1/conf1/crypton-slides.ps