Analyse kryptographischer Algorithmen: CAST-128 / 256

Roland Stigge
stigge@informatik.hu-berlin.de
http://www.informatik.hu-berlin.de/~stigge/
Humboldt Universität zu Berlin

8. Mai 2002

CAST: Überblick

- Einführung
- Geschichte
- Eigenschaften
- Algorithmus
- Analyse

Einführung

- Blockchiffre
- “CAST Design Procedure”
- CAST-256: AES-Kandidat
- *nicht* in Endauswahl

Geschichte

- Carlisle Adams ("CA"), Entrust Inc., Kanada
- Stafford Tavares ("ST"), Queens University Kingston, Kanada
- 1992: CAST-1
- ...
- 1995: CAST-5 = CAST-128
- 1997: CAST-6 = CAST-256 ($\rightarrow$ AES, 1998)

Geschichte (2)

- Beide: international freigegeben, offengelegt
- CAST-128: RFC2144, CAST-256: RFC2612
- kommerziell / nicht kommerziell
- CAST-128 in PGP 5+
- Bruce Schneier (1998): “I give a big yuk to CAST-128.”

Eigenschaften: CAST-128

- Ähnlichkeit: DES, Blowfish
- SPN (Substitution-Permutation-Network)
- Produktalgorithmus (Runden)
- Feistel-Netzwerk
- Resistenz gegen:
  - lineare
  - differentielle
  - related-key

  Kryptoanalysis

Eigenschaften: CAST-128 (2)

- Blockgröße: 64 bits
- Schlüsselgröße: 40 bis 128 bits (8-bit Schritte)
- Runden: 16 (kleine Schlüssel: 12)
- SAC (Strict Avalanche Criterion)
- BIC (Bit Independence Criterion)
- "No-Complementation"-Eigenschaft (DES: *nicht*)

Eigenschaften: CAST-128 (3)

- keine schwachen Schlüssel
- keine semi-schwachen Schlüssel
- Entschlüsselung wie Verschlüsselung (bis auf Key-Schedule)
- Performanz: 3,3 MByte/sec (150 MHz Pentium)

Eigenschaften: CAST-128 (4)

- 8 Substitutions-Boxen (S-Boxen):
  - Nicht-Linearität: 74
  - Maximum bei Differenzen-Verteilung von 2
- Operationsmix:
  - XOR, Addition, Subtraktion
  - verschiedene Reihenfolge
  - mehrere Rundenfunktionen
- $\rightarrow$ gegen lin./diff./HOD Angriffe

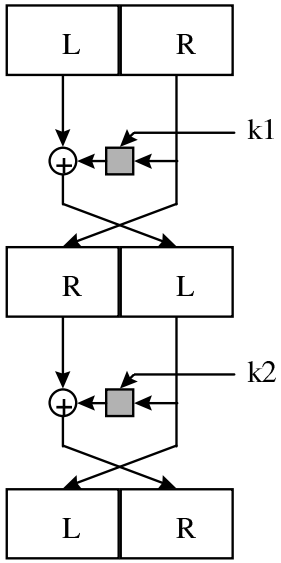
Eigenschaften: CAST-256

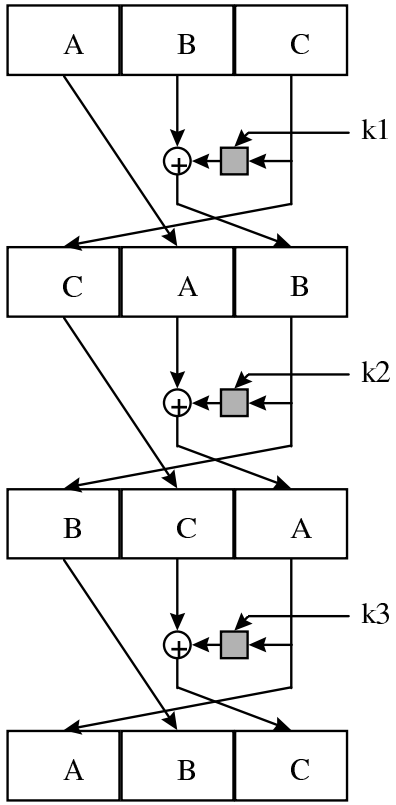
- Basiert auf CAST-128:
  - Rundenfunktionen
  - S-Boxen S_1 bis S_4
  - Schlüsselmix
  - Operationsmix

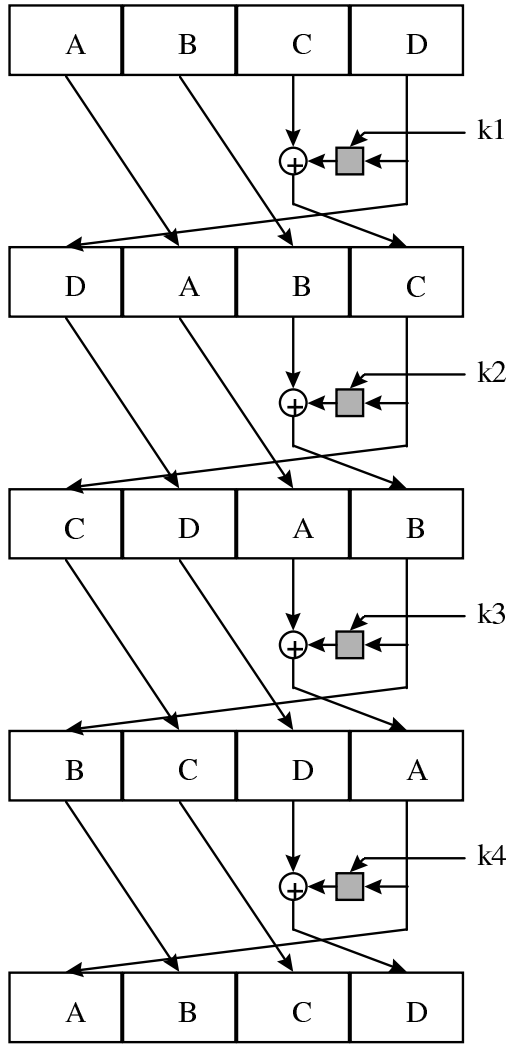
Eigenschaften: CAST-256 (2)

- Schlüsselgrößen: 128, 160, 192, 224, 256 Bits
- Blockgröße: 128 Bits
- Rundenschlüssel durch Verschlüsselung
- 48 Runden: 12 “Vierer-Runden”
- erweitertes (“unvollständiges”) Feistel-Netzwerk:

Feistel-Netzwerk (Eigenschaften: CAST-256)







Algorithmus: CAST-128

- Def.: Rundenfunktion(en) f: $\mathbb{Z}_{2^{32}} \rightarrow \mathbb{Z}_{2^{32}}$:

$$
\begin{aligned}
\text{Typ 1:} \quad I &= ((k_{m_i} + D) \circlearrowleft k_{r_i}) \\
f_1 &= ((S_1[I_a] \oplus S_2[I_b]) - S_3[I_c]) + S_4[I_d] \\
\text{Typ 2:} \quad I &= ((k_{m_i} \oplus D) \circlearrowleft k_{r_i}) \\
f_2 &= ((S_1[I_a] - S_2[I_b]) + S_3[I_c]) \oplus S_4[I_d] \\
\text{Typ 3:} \quad I &= ((k_{m_i} - D) \circlearrowleft k_{r_i}) \\
f_3 &= ((S_1[I_a] + S_2[I_b]) \oplus S_3[I_c]) - S_4[I_d]
\end{aligned}
$$

- Typ n: Bei Runde r mit $r \equiv n \pmod 3$

Algorithmus: CAST-128 (2)

1. Eingabe: Klartext $m_1 \cdots m_{64}$, Schlüssel $k = k_1 \cdots k_{128}$

2. Rundenschlüsselgenerierung: k_{m_i} (Maske, 32 bits), k_{r_i} (Rotation, 5 bits) für jede Runde i (abh. von k)

3. $(L_0, R_0) \leftarrow (m_1 \cdots m_{64})$
   d.h. $L_0 \leftarrow (m_1 \cdots m_{32})$ und $R_0 \leftarrow (m_{33} \cdots m_{64})$

4. 16 Runden: $L_i = R_{i-1}$ und $R_i = L_{i-1} \oplus f(R_{i-1}, k_{m_i}, k_{r_i})$ für alle $i \in \{1, \ldots, 16\}$ (Typ von f hängt von i ab)

5. Ausgabe: Kryptotext $c_1 \cdots c_{64} \leftarrow (R_{16}, L_{16})$
   d.h. noch eine Vertauschung

Algorithmus: CAST-128 (3)

- Entschlüsselung: Gleicher Algorithmus, nur Schlüssel in umgekehrter Reihenfolge ($(R_{16}, L_{16}) \rightarrow (L_0, R_0)$)

- 8 S-Boxen:
  - 4 für Rundenfunktionen ($S_1, \ldots, S_4$)
  - 4 für (einmalige) Schlüsselgenerierung ($S_5, \ldots, S_8$)
  - jew. "8x32": Lookup-Table von 2^8 x 32bits

Algorithmus: CAST-128 (4) - Beispiel: S-Box S_1

```
30fb40d4 9fa0ff0b 6beccd2f 3f258c7a 1e213f2f 9c004dd3 6003e540 cf9fc949
bfd4af27 88bbbdb5 e2034090 98d09675 6e63a0e0 15c361d2 c2e7661d 22d4ff8e
28683b6f c07fd059 ff2379c8 775f50e2 43c340d3 df2f8656 887ca41a a2d2bd2d
a1c9e0d6 346c4819 61b76d87 22540f2f 2abe32e1 aa54166b 22568e3a a2d341d0
66db40c8 a784392f 004dff2f 2db9d2de 97943fac 4a97c1d8 527644b7 b5f437a7
b82cbaef d751d159 6ff7f0ed 5a097a1f 827b68d0 90ecf52e 22b0c054 bc8e5935
4b6d2f7f 50bb64a2 d2664910 bee5812d b7332290 e93b159f b48ee411 4bff345d
fd45c240 ad31973f c4f6d02e 55fc8165 d5b1caad a1ac2dae a2d4b76d c19b0c50
882240f2 0c6e4f38 a4e4bfd7 4f5ba272 564c1d2f c59c5319 b949e354 b04669fe
b1b6ab8a c71358dd 6385c545 110f935d 57538ad5 6a390493 e63d37e0 2a54f6b3
3a787d5f 6276a0b5 19a6fcdf 7a42206a 29f9d4d5 f61b1891 bb72275e aa508167
38901091 c6b505eb 84c7cb8c 2ad75a0f 874a1427 a2d1936b 2ad286af aa56d291
d7894360 425c750d 93b39e26 187184c9 6c00b32d 73e2bb14 a0bebc3c 54623779
64459eab 3f328b82 7718cf82 59a2cea6 04ee002e 89fe78e6 3fab0950 325ff6c2
81383f05 6963c5c8 76cb5ad6 d49974c9 ca180dcf 380782d5 c7fa5cf6 8ac31511
35e79e13 47da91d0 f40f9086 a7e2419e 31366241 051ef495 aa573b04 4a805d8d
548300d0 00322a3c bf64cddf ba57a68e 75c6372b 50afd341 a7c13275 915a0bf5
6b54bfab 2b0b1426 ab4cc9d7 449ccd82 f7fbf265 ab85c5f3 1b55db94 aad4e324
...
```

Algorithmus: CAST-128 (5)

- bei $\|k\| \leq 80$ nur 12 statt 16 Runden
- falls $\|k\| < 128$, so $k_{neu} := 2^{128-\|k\|}k$
- Für Schlüsselgenerierung (k_{m_i} und k_{r_i}): Seien im folgenden
  - $x_0, \ldots, x_{15}$ (Bytes) Kopie von k
  - $z_0, \ldots, z_{15}$ temporäre Bytes
- $k_{m_i} =$ 32-Bit-Werte
- $k_{r_i} =$ 32-Bit berechnet, aber nur 5 Bit verwendet

Algorithmus: CAST-128 (6)

Schlüsselgenerierung:

$$
\begin{aligned}
z_0 z_1 z_2 z_3 &= x_0 x_1 x_2 x_3 \oplus S_5[x_{13}] \oplus S_6[x_{15}] \oplus S_7[x_{12}] \oplus S_8[x_{14}] \oplus S_7[x_8] \\
z_4 z_5 z_6 z_7 &= x_8 x_9 x_{10} x_{11} \oplus S_5[z_0] \oplus S_6[z_2] \oplus S_7[z_1] \oplus S_8[z_3] \oplus S_8[x_{10}] \\
z_8 z_9 z_{10} z_{11} &= x_{12} x_{13} x_{14} x_{15} \oplus S_5[z_7] \oplus S_6[z_6] \oplus S_7[z_5] \oplus S_8[z_4] \oplus S_5[x_9] \\
z_{12} z_{13} z_{14} z_{15} &= x_4 x_5 x_6 x_7 \oplus S_5[z_{10}] \oplus S_6[z_9] \oplus S_7[z_{11}] \oplus S_8[z_8] \oplus S_6[x_{11}] \\
k_{m_1} &= S_5[z_8] \oplus S_6[z_9] \oplus S_7[z_7] \oplus S_8[z_6] \oplus S_5[z_2] \\
k_{m_2} &= S_5[z_{10}] \oplus S_6[z_{11}] \oplus S_7[z_5] \oplus S_8[z_4] \oplus S_6[z_6] \\
k_{m_3} &= S_5[z_{12}] \oplus S_6[z_{13}] \oplus S_7[z_3] \oplus S_8[z_2] \oplus S_7[z_9] \\
k_{m_4} &= S_5[z_{14}] \oplus S_6[z_{15}] \oplus S_7[z_1] \oplus S_8[z_0] \oplus S_8[z_{12}] \\
&\vdots
\end{aligned}
$$

Algorithmus: CAST-256

- Def.: $\beta = (ABCD)$ – 128-Bit-Block; $A, \ldots, D$ – jew. 32 Bits
- Def.: $\beta \leftarrow Q_i(\beta)$ gdw.
$$\begin{cases} C &= C \oplus f_1(D, k_{r_0}^{(i)}, k_{m_0}^{(i)}) \\ B &= B \oplus f_2(C, k_{r_1}^{(i)}, k_{m_1}^{(i)}) \\ A &= A \oplus f_3(B, k_{r_2}^{(i)}, k_{m_2}^{(i)}) \\ D &= D \oplus f_1(A, k_{r_3}^{(i)}, k_{m_3}^{(i)}) \end{cases}$$
- Def.: $\beta \leftarrow \overline{Q_i}(\beta)$ gdw.
$$\begin{cases} D &= D \oplus f_1(A, k_{r_3}^{(i)}, k_{m_3}^{(i)}) \\ A &= A \oplus f_3(B, k_{r_2}^{(i)}, k_{m_2}^{(i)}) \\ B &= B \oplus f_2(C, k_{r_1}^{(i)}, k_{m_1}^{(i)}) \\ C &= C \oplus f_1(D, k_{r_0}^{(i)}, k_{m_0}^{(i)}) \end{cases}$$

Algorithmus: CAST-256 (2)

- $Q =$ "Vorwärts-Vierer-Runde"
- $\overline{Q} =$ "Rückwärts-Vierer-Runde"
- Def.: $k_r^{(i)} = \{k_{r_0}^{(i)}, k_{r_1}^{(i)}, k_{r_2}^{(i)}, k_{r_3}^{(i)}\}$
- Def.: $k_m^{(i)} = \{k_{m_0}^{(i)}, k_{m_1}^{(i)}, k_{m_2}^{(i)}, k_{m_3}^{(i)}\}$

Algorithmus: CAST-256 (3)

- Def.: $\kappa = (ABCDEFGH)$ – 256-Bit-Schlüssel; $A, \ldots, H$ – jew. 32 Bits
- Def.: $\kappa \leftarrow \omega_i(\kappa)$ gdw.

$$\begin{cases} G &= G \oplus f_1(H, t_{r_0}^{(i)}, t_{m_0}^{(i)}) \\ F &= F \oplus f_2(G, t_{r_1}^{(i)}, t_{m_1}^{(i)}) \\ E &= E \oplus f_3(F, t_{r_2}^{(i)}, t_{m_2}^{(i)}) \\ D &= D \oplus f_1(E, t_{r_3}^{(i)}, t_{m_3}^{(i)}) \\ C &= C \oplus f_2(D, t_{r_4}^{(i)}, t_{m_4}^{(i)}) \\ B &= B \oplus f_3(C, t_{r_5}^{(i)}, t_{m_5}^{(i)}) \\ A &= A \oplus f_1(B, t_{r_6}^{(i)}, t_{m_6}^{(i)}) \\ H &= H \oplus f_2(A, t_{r_7}^{(i)}, t_{m_7}^{(i)}) \end{cases}$$

- $\omega =$ “Vorwärts-Oktave”

Algorithmus: CAST-256 (4)

- Def.: $k_r^{(i)} \leftarrow \kappa$ gdw.
$$\begin{cases} k_{r_0}^{(i)} &= LSB(5, A) \\ k_{r_1}^{(i)} &= LSB(5, C) \\ k_{r_2}^{(i)} &= LSB(5, E) \\ k_{r_3}^{(i)} &= LSB(5, G) \end{cases}$$

- Def.: $k_m^{(i)} \leftarrow \kappa$ gdw.
$$\begin{cases} k_{m_0}^{(i)} &= H \\ k_{m_1}^{(i)} &= F \\ k_{m_2}^{(i)} &= D \\ k_{m_3}^{(i)} &= B \end{cases}$$

Algorithmus: CAST-256 (5)

- Die Verschlüsselung ansich:
  1: Eingabe: $\beta = 128$-Bit-Klartextblock
  2: **for** $i = 0$ to 5 **do**
  3: $\quad \beta \leftarrow Q_i(\beta)$
  4: **end for**
  5: **for** $i = 6$ to 11 **do**
  6: $\quad \beta \leftarrow \overline{Q_i}(\beta)$
  7: **end for**
  8: Ausgabe: 128-Bit-Kryptotextblock $= \beta$

- Entschlüsselung: s.o. mit umgekehrter Reihenfolge der Schlüssel $k_r^{(i)}$ und $k_m^{(i)}$

Algorithmus: CAST-256 (6)

Rundenschlüsselberechnung: Initialisierung

1: $c_m = 2^{30}\sqrt{2} = 5A827999_{16}$
2: $m_m = 2^{30}\sqrt{3} = 6ED9EBA1_{16}$
3: $c_r = 19$
4: $m_r = 17$
5: **for** $i = 0$ to 23 **do**
6: **for** $j = 0$ to 7 **do**
7: $t_{m_j}^{(i)} = c_m$
8: $c_m = (c_m + m_m) \mod 2^{32}$
9: $t_{r_j}^{(i)} = c_r$
10: $c_r = (c_r + m_r) \mod 32$
11: **end for**
12: **end for**

Algorithmus: CAST-256 (7)

- falls $\|k\| < 256$, so $k_{neu} := 2^{256-\|k\|}k$

- Rundenschlüsselberechnung:

  1: $\kappa = ABCDEFGH = 256\text{-Bit-Primär-Schlüssel } k$
  2: **for** $i = 0$ to 11 **do**
  3: $\quad \kappa \leftarrow \omega_{2i}(\kappa)$
  4: $\quad \kappa \leftarrow \omega_{2i+1}(\kappa)$
  5: $\quad k_r^{(i)} \leftarrow \kappa$
  6: $\quad k_m^{(i)} \leftarrow \kappa$
  7: **end for**

Analyse: Allgemein

- 10 Jahre Entwicklung (1998)
- keine praxisrelevanten (veröff.) Angriffe
- bis 6 Runden, vereinfachte Bedingungen (keine Rotation, nur XOR):
  - non-surjective attacks
  - HOD attacks
- denn: erst nach 7 Runden ist jedes Ausgabebit von jedem Eingabebit abhängig ("non-degeneracy property")
- alle Schlüssel verwendbar, da keine schwachen oder semi-schwachen Schlüssel bekannt

Analyse: Allgemein (2)

Mögliche, bisher erfolglose, Angriffe:

- Ciphertext Only Attack
- Known Plaintext Attack, Lineare Kryptoanalysis
- Chosen Plaintext Attack, Differentielle Kryptoanalysis
- Chosen Key Attack

Analyse: Allgemein (3)

- Related Key Attack
- Higher Order Differential (HOD) Attack
- Non Surjective Attack
- Combination Attack

Analyse: CAST-128

- Gute Speicher- / Zeitkomplexität
- gleiche Performanz bei allen Schlüsselgrößen

Analyse: CAST-256

- "Erbt" Sicherheit des Vorgängers
- spart Speicherplatz (4KBytes von 4 S-Boxen)
- 2/3 der Geschwindigkeit des Vorgängers
- kaum halb so schnell wie Rijndael
- Transparenz: Feistel-Netzwerk gut verständlich

Literatur

[1] Carlisle Adams: "The CAST-256 Encryption Algorithm" (Teil der AES-Bewerbung), Juni 1998
http://www.entrust.com/resources/pdf/cast-256.pdf

[2] C. Adams, Entrust Technologies: RFC2144 - "The CAST-128 Encryption Algorithm", Mai 1997
http://www.ietf.org/rfc/rfc2144.txt

[3] C. Adams, J. Gilchrist: RFC2612 - "The CAST-256 Encryption Algorithm", Juni 1999
http://www.ietf.org/rfc/rfc2612.txt

[4] C. Adams: Constructing Symmetric Ciphers using the CAST Design Procedure
http://www.entrust.com/resources/pdf/cast.pdf

[5] C. Adams: "CAST-256 - A Submission for the Advanced Encryption Standard" (Folien), August 1998
http://csrc.nist.gov/encryption/aes/round1/conf1/cast-slides.pdf

[6] Adams, Heys, Tavares, Wiener: "An Analysis of the CAST-256 Cipher"
http://www.engr.mun.ca/~howard/PAPERS/cast256.ps

Schluß

Vielen Dank für die Aufmerksamkeit!

Fragen?